NEC

Preliminary User's Manual

V850E/CA2™ JUPITER

32-/16-bit Romless Microcontroller

Hardware

µPD703128, µPD703129

Document No. U15839EE1V0UM00
Date Published August 2003

NOTES FOR CMOS DEVICES

① PRECAUTION AGAINST ESD FOR SEMICONDUCTORS

Note:

Strong electric field, when exposed to a MOS device, can cause destruction of the gate oxide and ultimately degrade the device operation. Steps must be taken to stop generation of static electricity as much as possible, and quickly dissipate it once, when it has occurred. Environmental control must be adequate. When it is dry, humidifier should be used. It is recommended to avoid using insulators that easily build static electricity. Semiconductor devices must be stored and transported in an anti-static container, static shielding bag or conductive material. All test and measurement tools including work bench and floor should be grounded. The operator should be grounded using wrist strap. Semiconductor devices must not be touched with bare hands. Similar precautions need to be taken for PW boards with semiconductor devices on it.

② HANDLING OF UNUSED INPUT PINS FOR CMOS

Note:

No connection for CMOS device inputs can be cause of malfunction. If no connection is provided to the input pins, it is possible that an internal input level may be generated due to noise, etc., hence causing malfunction. CMOS devices behave differently than Bipolar or NMOS devices. Input levels of CMOS devices must be fixed high or low by using a pull-up or pull-down circuitry. Each unused pin should be connected to V_{DD} or GND with a resistor, if it is considered to have a possibility of being an output pin. All handling related to the unused pins must be judged device by device and related specifications governing the devices.

③ STATUS BEFORE INITIALIZATION OF MOS DEVICES

Note:

Power-on does not necessarily define initial status of MOS device. Production process of MOS does not define the initial operation status of the device. Immediately after the power source is turned ON, the devices with reset function have not yet been initialized. Hence, power-on does not guarantee out-pin levels, I/O settings or contents of registers. Device is not initialized until the reset signal is received. Reset operation must be executed immediately after power-on for devices having reset function.

Regional Information

Some information contained in this document may vary from country to country. Before using any NEC product in your application, please contact the NEC office in your country to obtain a list of authorized representatives and distributors. They will verify:

- Device availability
- Ordering information
- Product release schedule
- Availability of related technical literature
- Development environment specifications (for example, specifications for third-party tools and components, host computers, power plugs, AC supply voltages, and so forth)
- Network requirements

In addition, trademarks, registered trademarks, export restrictions, and other legal issues may also vary from country to country.

NEC Electronics Inc. (U.S.)
Santa Clara, California
Tel: 408-588-6000
800-366-9782
Fax: 408-588-6130
800-729-9288

NEC Electronics (Europe) GmbH
Duesseldorf, Germany
Tel: 0211-65 03 01
Fax: 0211-65 03 327

Sucursal en España
Madrid, Spain
Tel: 091- 504 27 87
Fax: 091- 504 28 60

Succursale Française
Vélizy-Villacoublay, France
Tel: 01-30-67 58 00
Fax: 01-30-67 58 99

Filiale Italiana
Milano, Italy
Tel: 02-66 75 41
Fax: 02-66 75 42 99

Branch The Netherlands
Eindhoven, The Netherlands
Tel: 040-244 58 45
Fax: 040-244 45 80

Branch Sweden
Taeby, Sweden
Tel: 08-63 80 820
Fax: 08-63 80 388

United Kingdom Branch
Milton Keynes, UK
Tel: 01908-691-133
Fax: 01908-670-290

NEC Electronics Hong Kong Ltd.
Hong Kong
Tel: 2886-9318
Fax: 2886-9022/9044

NEC Electronics Hong Kong Ltd.
Seoul Branch
Seoul, Korea
Tel: 02-528-0303
Fax: 02-528-4411

NEC Electronics Singapore Pte. Ltd.
Singapore
Tel: 65-6253-8311
Fax: 65-6250-3583

NEC Electronics Taiwan Ltd.
Taipei, Taiwan
Tel: 02-2719-2377
Fax: 02-2719-5951

NEC do Brasil S.A.
Electron Devices Division
Guarulhos, Brasil
Tel: 55-11-6465-6810
Fax: 55-11-6465-6829

Preface

Readers — This manual is intended for users who want to understand the functions of the V850E/CA2 (nickname Jupiter).

Purpose — This manual presents the hardware manual of V850E/CA2.

Organization — This system specification describes the following sections:

- Pin function
- CPU function
- Internal peripheral function

Legend — Symbols and notation are used as follows:

Weight in data notation	:	Left is high-order column, right is low order column
Active low notation	:	$\overline{\text{xxx}}$ (pin or signal name is over-scored) or /xxx (slash before signal name)
Memory map address:	:	High order at high stage and low order at low stage
Note	:	Explanation of (Note) in the text
Caution	:	Item deserving extra attention
Remark	:	Supplementary explanation to the text
Numeric notation	:	Binary . . . xxxx or xxxB Decimal . . . xxxx Hexadecimal . . . xxxxH or 0x xxxx

Prefixes representing powers of 2 (address space, memory capacity)

K (kilo) : 2^{10} = 1024
M (mega) : 2^{20} = 1024^2 = 1,048,576
G (giga) : 2^{30} = 1024^3 = 1,073,741,824

Table of Contents

Preface 5

Chapter 1 Introduction 23
- **1.1 General** 23
- **1.2 Device Features** 24
- **1.3 Application Fields** 26
- **1.4 Ordering Information** 26
- **1.5 Pin Configuration (Top View)** 27
- **1.6 Function Block Diagram** 29
  - 1.6.1 On-chip units 30

Chapter 2 Pin Functions 33
- **2.1 List of Pin Functions** 33
- **2.2 Description of Pin Functions** 40
- **2.3 Types of Pin I/O Circuit and Connection of Unused Pins** 51

Chapter 3 CPU Function 55
- **3.1 Features** 55
- **3.2 CPU Register Set** 56
  - 3.2.1 Program register set 57
  - 3.2.2 System register set 58
- **3.3 Operation Modes** 62
  - 3.3.1 Operation modes 62
  - 3.3.2 Operation mode specification 63
- **3.4 Address Space** 65
  - 3.4.1 CPU address space 65
  - 3.4.2 Image 66
  - 3.4.3 Wrap-around of CPU address space 67
- **3.5 Memory Map** 68
  - 3.5.1 Area 70
  - 3.5.2 Recommended use of address space 75
  - 3.5.3 Peripheral I/O Registers 76
  - 3.5.4 Programmable peripheral I/O registers 83
- **3.6 Specific Registers** 103
  - 3.6.1 Command Register (PRCMD) 104
  - 3.6.2 Peripheral Command Register (PHCMD) 105
  - 3.6.3 Peripheral Status Register (PHS) 106
  - 3.6.4 Internal peripheral function wait control register (VSWC) 107

Chapter 4 Bus Control Function 109
- **4.1 Features** 109
- **4.2 Bus Control Pins** 109
- **4.3 Memory Block Function** 110
  - 4.3.1 Chip Select Control Function 111
- **4.4 Programmable peripheral I/O registers** 113
- **4.5 Bus Cycle Type Control Function** 115
  - 4.5.1 Bus cycle type configuration 115
- **4.6 Bus Access** 116
  - 4.6.1 Number of access clocks 116
  - 4.6.2 Bus sizing function 116
  - 4.6.3 Endian control function 117
- **4.7 Cache Configuration** 118
  - 4.7.1 Bus width 119
- **4.8 Wait Function** 131
  - 4.8.1 Programmable wait function 131

4.8.2 External wait function . . . 133
4.8.3 Relationship between programmable wait and external wait . . . 133
4.9 Idle State Insertion Function . . . 134
4.10 Bus Priority Order . . . 135
4.11 Boundary Operation Conditions . . . 136
4.11.1 Program space . . . 136
4.11.2 Data space . . . 136

Chapter 5 Memory Access Control Function . . . 137
5.1 SRAM, External ROM, External I/O Interface . . . 137
5.1.1 Features . . . 137
5.1.2 SRAM connections . . . 138
5.1.3 SRAM, external ROM, external I/O access . . . 139
5.2 Page ROM Controller (ROMC) . . . 145
5.2.1 Features . . . 145
5.2.2 Page ROM connections . . . 146
5.2.3 On-page/off-page judgment . . . 147
5.2.4 Page ROM configuration register (PRC) . . . 149
5.2.5 Page ROM access . . . 150

Chapter 6 Instruction Cache . . . 155
6.1 Features . . . 155
6.2 Configuration . . . 156
6.2.1 Four Kbytes 2-way set-associative Instruction Cache . . . 157
6.3 Control Registers . . . 158
6.4 Instruction Cache Operation . . . 160
6.5 Instruction Cache Initialisation . . . 166
6.6 Operating Precautions . . . 167

Chapter 7 DMA Functions (DMA Controller) . . . 169
7.1 Features . . . 169
7.2 Control Registers . . . 170
7.2.1 DMA source address registers H0 to H3 (DSAH0 to DSAH3) . . . 170
7.2.2 DMA destination address registers H0 to H3 (DDAH0 to DDAH3) . . . 172
7.2.3 DMA transfer count registers 0 to 3 (DBC0 to DBC3) . . . 174
7.2.4 DMA addressing control registers 0 to 3 (DADC0 to DADC3) . . . 175
7.2.5 DMA channel control registers 0 to 3 (DCHC0 to DCHC3) . . . 177
7.2.6 DMA disable status register (DDIS) . . . 178
7.2.7 DMA restart register (DRST) . . . 178
7.2.8 DMA trigger factor register 0 (DTFR0) . . . 179
7.2.9 DMA trigger factor register 1 (DTFR1) . . . 180
7.2.10 DMA trigger factor register 2 (DTFR2) . . . 181
7.2.11 DMA trigger factor register 3 (DTFR3) . . . 182
7.3 Next Address Setting Function . . . 183
7.4 DMA Bus States . . . 184
7.4.1 Types of bus states . . . 184
7.4.2 DMAC bus cycle state transition . . . 186
7.5 Transfer Mode . . . 187
7.5.1 Single transfer mode . . . 187
7.5.2 Single-step transfer mode . . . 189
7.5.3 Line Transfer Mode . . . 190
7.5.4 Block transfer mode . . . 192
7.6 Transfer Types . . . 193
7.6.1 Two-cycle transfer . . . 193
7.7 Transfer Object . . . 193
7.7.1 Transfer type and transfer object . . . 193
7.8 DMA Channel Priorities . . . 194
7.9 DMA Transfer Start Factors . . . 194

7.10 Forcible Interruption . . . 195
7.11 Forcible Termination . . . 196
7.12 DMA Transfer Completion . . . 197
7.12.1 DMA transfer end interrupt . . . 197
7.12.2 Terminal count output upon DMA transfer end . . . 197
7.13 Precautions . . . 198

Chapter 8 Interrupt/Exception Processing Function . . . 199
8.1 Features . . . 199
8.2 Non-Maskable Interrupts . . . 203
8.2.1 Operation . . . 206
8.2.2 Restore . . . 207
8.2.3 Non-maskable interrupt status flag (NP) . . . 208
8.2.4 Edge Detection Function . . . 208
8.3 Maskable Interrupts . . . 209
8.3.1 Operation . . . 209
8.3.2 Restore . . . 211
8.3.3 Priorities of maskable interrupts . . . 212
8.3.4 Interrupt control register (xxIC) . . . 216
8.3.5 Interrupt mask registers 0 to 3 (IMR0 to IMR3) . . . 219
8.3.6 In-service priority register (ISPR) . . . 220
8.3.7 Maskable interrupt status flag (ID) . . . 220
8.4 Noise Elimination Circuit . . . 221
8.4.1 Analog Filter . . . 222
8.4.2 Interrupt Trigger Mode Selection . . . 222
8.4.3 Interrupt Edge Detection Control Registers . . . 223
8.5 Software Exception . . . 227
8.5.1 Operation . . . 227
8.5.2 Restore . . . 228
8.5.3 Exception status flag (EP) . . . 229
8.6 Exception Trap . . . 230
8.6.1 Illegal opcode definition . . . 230
8.6.2 Debug trap . . . 232
8.7 Multiple Interrupt Processing Control . . . 234
8.8 Interrupt Response Time . . . 236
8.9 Periods in Which Interrupts Are Not Acknowledged . . . 237

Chapter 9 Clock Generator . . . 239
9.1 Features . . . 239
9.2 Configuration . . . 240
9.3 Control Registers . . . 241
9.3.1 Clock Control Register (CKC) . . . 241
9.3.2 Clock Generator Status Register (CGSTAT) . . . 243
9.3.3 Watchdog Timer Clock Control Register (WCC) . . . 244
9.3.4 Processor Clock Control Register (PCC) . . . 245
9.3.5 Reset Source Monitor Register (RSM) . . . 247
9.3.6 SSCG Frequency Modulation Control Register (SCFMC) . . . 248
9.3.7 SSCG Frequency Control Register 0 (SCFC0) . . . 249
9.3.8 SSCG Frequency Control Register 1 (SCFC1) . . . 250
9.4 Power Saving Functions . . . 251
9.4.1 General . . . 251
9.4.2 Power Save Modes Outline . . . 253
9.4.3 Power Saving Mode Functions . . . 254
9.4.4 HALT mode . . . 256
9.4.5 IDLE Mode . . . 258
9.4.6 WATCH mode . . . 259
9.4.7 SUB WATCH mode . . . 261
9.4.8 Software STOP mode . . . 265

9.5 Register Description . . . 268
9.5.1 Power Save Control Register (PSC) . . . 268
9.5.2 Power Save Mode Register (PSM) . . . 270

Chapter 10 Timer . . . 271
10.1 Timer C . . . 271
10.1.1 Features (Timer C) . . . 271
10.1.2 Function overview (Timer C) . . . 272
10.1.3 Basic configuration . . . 274
10.1.4 Control registers . . . 278
10.1.5 Operation . . . 283
10.1.6 Sub Oscillator Calibration Function . . . 293
10.1.7 Precautions Timer C . . . 295
10.2 Timer D . . . 296
10.2.1 Features Timer D . . . 296
10.2.2 Function overview Timer Dn . . . 296
10.2.3 Basic configuration . . . 297
10.2.4 Control register . . . 301
10.2.5 Operation . . . 302
10.2.6 Application example . . . 304
10.2.7 Precautions for Timer Dn . . . 305
10.3 Timer G . . . 306
10.3.1 Features of Timer G . . . 306
10.3.2 Function overview of each Timer Gn . . . 307
10.3.3 Basic configuration . . . 309
10.3.4 Control registers . . . 313
10.3.5 Output delay operation . . . 321
10.3.6 Explanation of basic operation . . . 322
10.3.7 Operation in Free-run mode . . . 324
10.3.8 Match and clear mode . . . 335
10.3.9 Edge noise elimination . . . 346
10.3.10 Precautions Timer Gn . . . 347

Chapter 11 Watch Timer . . . 349
11.1 Function . . . 349
11.2 Configuration . . . 350
11.3 Watch Timer Control Register . . . 350
11.4 Operations . . . 352
11.4.1 Selection of the Watch Timer Clock . . . 352
11.4.2 Control of the watch timer . . . 353
11.4.3 Operation as interval timer . . . 354

Chapter 12 Watchdog Timer Function . . . 357
12.1 Functions . . . 357
12.2 Configuration . . . 358
12.3 Watchdog Timer Control Register . . . 359
12.4 Operation . . . 362
12.4.1 Operating as watchdog timer . . . 362

Chapter 13 Serial Interface Function . . . 363
13.1 Features . . . 363
13.2 Asynchronous Serial Interfaces UART5n (UART50, UART51) . . . 364
13.2.1 Features . . . 364
13.2.2 Configuration . . . 365
13.2.3 Control registers . . . 367
13.2.4 Interrupt requests . . . 374
13.2.5 Operation . . . 375
13.2.6 Dedicated baud rate generators (BRG) of UART5n (n = 0, 1) . . . 384

13.2.7 Precautions . . . 391
13.3 Clocked Serial Interfaces (CSI00 to CSI02) . . . 392
13.3.1 Features . . . 392
13.3.2 Configuration . . . 393
13.3.3 Control registers . . . 395
13.3.4 Operation . . . 408
13.3.5 Output pins . . . 422
13.3.6 Dedicated baud rate generators 0, 1 (BRG0, BRG1) . . . 423

Chapter 14 FCAN Interface Function . . . 427
14.1 Features . . . 427
14.2 Outline of the FCAN System . . . 428
14.2.1 General . . . 428
14.2.2 CAN memory and register layout . . . 429
14.2.3 Clock structure . . . 438
14.2.4 Interrupt handling . . . 439
14.2.5 Time stamp . . . 441
14.2.6 Message handling . . . 443
14.2.7 Mask handling . . . 448
14.2.8 Remote frame handling . . . 449
14.3 Control and Data Registers . . . 452
14.3.1 Bit set/clear function . . . 452
14.3.2 Common registers . . . 454
14.3.3 CAN interrupt pending registers . . . 467
14.3.4 CAN message buffer registers . . . 472
14.3.5 CAN Module Registers . . . 485
14.4 Operating Considerations . . . 509
14.4.1 Rules to be observed for correct baud rate settings . . . 509
14.4.2 Example for baudrate setting of CAN module . . . 510
14.4.3 Ensuring data consistency . . . 512
14.4.4 Operating states of the CAN modules . . . 514
14.4.5 Initialisation routines . . . 515

Chapter 15 A/D Converter . . . 523
15.1 Features . . . 523
15.2 Configuration . . . 524
15.3 Control Registers . . . 527
15.3.1 Register format of A/D Converter Control Register . . . 527
15.3.2 Input voltage and conversion results . . . 535
15.4 Interrupt Request . . . 536
15.5 A/D Converter Operation . . . 537
15.5.1 A/D converter basic operation . . . 537
15.5.2 Operation modes . . . 538
15.6 A/D Converter Precautions . . . 541
15.7 How to read the A/D Converter Characteristics Table . . . 542

Chapter 16 Port Functions . . . 545
16.1 Features . . . 545
16.2 Port Configuration . . . 546
16.3 Pin Functions of Each Port . . . 556
16.3.1 Port 1 . . . 556
16.3.2 Port 2 . . . 559
16.3.3 Port 3 . . . 562
16.3.4 Port 4 . . . 564
16.3.5 Port 5 . . . 566
16.3.6 Port 6 . . . 569
16.3.7 Port 7 . . . 572
16.3.8 Port 7/8 . . . 573

16.3.9 Port 9 574
16.3.10 Port AH 575
16.3.11 Port CS 577
16.3.12 Port CT 579
16.3.13 Port CM. 581

Chapter 17 RESET 583

17.1 Reset Overview 583
17.2 Features 583
17.3 Pin Functions 583
17.4 Reset by $\overline{\text{RESET}}$ Pin 585
17.5 Reset by Watchdog Timer 587
17.6 Reset Output 587
17.7 Initialization 588

Appendix A List of Instruction Sets 589

Appendix B Index 601

List of Figures

Figure 1-1: V850E/CA2 Jupiter Pin Configuration ... 27
Figure 1-2: V850E/CA2 Jupiter Block Diagram ... 29
Figure 2-1: Pin I/O Circuits ... 54
Figure 3-1: CPU Register Set ... 56
Figure 3-2: Program Counter (PC) ... 57
Figure 3-3: Interrupt Source Register (ECR) ... 59
Figure 3-4: Program Status Word (PSW) ... 60
Figure 3-5: CPU Address Space ... 65
Figure 3-6: Image on Address Space ... 66
Figure 3-7: Wrap-around of Program Space ... 67
Figure 3-8: Wrap-around of Data Space ... 67
Figure 3-9: Memory Map (µPD703128 (A)) ... 68
Figure 3-10: Memory Map (µPD703129 (A), µPD703129 (A1)) ... 69
Figure 3-11: Internal RAM Area of µPD703129 ... 73
Figure 3-12: Internal RAM Area of µPD703128 ... 73
Figure 3-13: Internal Peripheral I/O Area ... 74
Figure 3-14: Example Application of wrap-around (µPD703129) ... 75
Figure 3-15: Programmable Peripheral I/O Register (Outline) ... 83
Figure 3-16: Peripheral Area Selection Control Register (BPC) ... 84
Figure 3-17: Command Register (PRCMD) Format ... 104
Figure 3-18: Peripheral Command Register (PHCMD) Format ... 105
Figure 3-19: Peripheral Status Register (PHS) Format ... 106
Figure 3-20: Internal peripheral function wait control register (VSWC) Format ... 107
Figure 4-1: Memory Block Function ... 110
Figure 4-2: Chip Area Select Control Registers 0, 1 (1/2) ... 111
Figure 4-3: Programmable Peripheral I/O Register (Outline) ... 113
Figure 4-4: Peripheral Area Selection Control Register (BPC) ... 114
Figure 4-5: Big Endian Addresses within Word ... 117
Figure 4-6: Little Endian Addresses within Word ... 117
Figure 4-7: Example of Wait Insertion ... 133
Figure 5-1: Example of Connection to SRAM ... 138
Figure 5-2: SRAM, External ROM, External I/O Access Timing (1/6) ... 139
Figure 5-3: Example of Page ROM Connections ... 146
Figure 5-4: On-Page/Off-Page Judgment during Page ROM Connection (1/2) ... 147
Figure 5-5: Page ROM Configuration Register (PRC) ... 149
Figure 5-6: Page ROM Access Timing (1/4) ... 150
Figure 6-1: Instruction Cache Configuration ... 156
Figure 6-2: Configuration of 4 KB 2-Way Set-Associative Instruction Cache ... 157
Figure 6-3: Instruction Cache Control Register (ICC) ... 158
Figure 6-4: Instruction Cache Data Configuration Register (ICD) ... 159
Figure 6-5: Instruction Cache Initial Register (ICI) ... 159
Figure 6-6: Operation on Instruction Cache Hit ... 160
Figure 6-7: Operation on Instruction Cache Miss ... 161
Figure 6-8: Refill Sequence to Instruction Cache (16-bit Data Bus) ... 162
Figure 6-9: iCache Area Setting Example ... 168
Figure 7-1: DMA Source Address Registers DSAH0 to DSAH3 (DSAH0 to DSAH3) ... 170
Figure 7-2: DMA Source Address Registers DSAL0 to DSAL3 (DSAL0 to DSAL3) ... 171
Figure 7-3: DMA Destination Address Registers 0H to 3H (DDA0H to DDA3H) ... 172
Figure 7-4: DMA Destination Address Registers L0 to L3 (DDAL0 to DDAL3) ... 173
Figure 7-5: DMA Transfer Count Registers 0 to 3 (DBC0 to DBC3) ... 174
Figure 7-6: DMA Addressing Control Registers 0 to 3 (DADC0 to DADC3) (1/2) ... 175
Figure 7-7: DMA Channel Control Registers 0 to 3 (DCHC0 to DCHC3) ... 177
Figure 7-8: DMA Disable Status Register (DDIS) ... 178
Figure 7-9: DMA Restart Register (DRST) ... 178
Figure 7-10: DMA Trigger Factor Registers 0 (DTFR0) ... 179

Figure 7-11: DMA Trigger Factor Registers 1 (DTFR1) ... 180
Figure 7-12: DMA Trigger Factor Registers 2 (DTFR2) ... 181
Figure 7-13: DMA Trigger Factor Registers 3 (DTFR3) ... 182
Figure 7-14: Buffer Register Configuration ... 183
Figure 7-15: DMAC Bus Cycle State Transition Diagram ... 186
Figure 7-16: Single Transfer Example 1 ... 187
Figure 7-17: Single Transfer Example 2 ... 187
Figure 7-18: Single Transfer Example 3 ... 188
Figure 7-19: Single Transfer Example 4 ... 188
Figure 7-20: Single-Step Transfer Example 1 ... 189
Figure 7-21: Single-Step Transfer Example 2 ... 189
Figure 7-22: Line Transfer Example 1 ... 190
Figure 7-23: Line Transfer Example 2 ... 190
Figure 7-24: Line Transfer Example 3 ... 191
Figure 7-25: Line Transfer Example 4 ... 191
Figure 7-26: Block Transfer Example ... 192
Figure 7-27: Example of Forcible Interruption of DMA Transfer ... 195
Figure 7-28: DMA Transfer Forcible Termination Example 1 ... 196
Figure 7-29: DMA Transfer Forcible Termination Example 2 ... 197
Figure 8-1: Example of Non-Maskable Interrupt Request Acknowledgement Operation (1/2) ... 204
Figure 8-2: Processing Configuration of Non-Maskable Interrupt ... 206
Figure 8-3: RETI Instruction Processing ... 207
Figure 8-4: Non-maskable Interrupt Status Flag (NP) ... 208
Figure 8-5: Interrupt Mode Register 3 (INTM3) ... 208
Figure 8-6: Maskable Interrupt Processing ... 210
Figure 8-7: RETI Instruction Processing ... 211
Figure 8-8: Example of Processing in Which Another Interrupt Request Is Issued While an Interrupt Is Being Processed (1/2) ... 213
Figure 8-9: Example of Processing Interrupt Requests Simultaneously Generated ... 215
Figure 8-10: Interrupt Control Register (xxIC) ... 216
Figure 8-11: Interrupt Mask Registers 0 to 3 (IMR0 to IMR3) ... 219
Figure 8-12: In-Service Priority Register (ISPR) ... 220
Figure 8-13: Maskable Interrupt Status Flag (ID) ... 220
Figure 8-14: Port Interrupt Input Circuit (P52, P53, P61, P62, P63, P64) ... 221
Figure 8-15: Timer G Input Circuit (P30, P35, P40, P45, P54, P55) ... 221
Figure 8-16: NMI Input Circuit ... 221
Figure 8-17: Interrupt Mode Register 0 (IMTM0) ... 223
Figure 8-18: Interrupt Mode Register 1 (IMTM1) ... 224
Figure 8-19: Interrupt Mode Register 2 (IMTM2) ... 225
Figure 8-20: Interrupt Mode Register 3 (IMTM3) ... 226
Figure 8-21: Software Exception Processing ... 227
Figure 8-22: RETI Instruction Processing ... 228
Figure 8-23: Exception Status Flag (EP) ... 229
Figure 8-24: Exception Trap Processing ... 230
Figure 8-25: Restore Processing from Exception Trap ... 231
Figure 8-26: Debug Trap Processing ... 232
Figure 8-27: Restore Processing from Debug Trap ... 233
Figure 8-28: Pipeline Operation at Interrupt Request Acknowledgment (Outline) ... 236
Figure 9-1: Block Diagram of the Clock Generator ... 240
Figure 9-2: Clock Control Register (CKC) (1/2) ... 241
Figure 9-3: Clock Generator Status Register (CGSTAT) ... 243
Figure 9-4: Watchdog Timer Clock Control Register (WCC) ... 244
Figure 9-5: Processor Clock Control Register (PCC) (1/2) ... 245
Figure 9-6: Reset Source Monitor Register (RSM) ... 247
Figure 9-7: SSCG Frequency Modulation Control Register (SCFMC) ... 248
Figure 9-8: SSCG Frequency Control Register 0 (SCFC0) ... 249
Figure 9-9: SSCG Frequency Control Register 1 (SCFC1) ... 250
Figure 9-10: Power Save Mode State Transition Diagram ... 252

Figure 9-11: Sub Watch mode released by $\overline{\text{RESET}}$ input 263
Figure 9-12: Sub Watch mode release by Watchdog reset, NMI, INT........ 264
Figure 9-13: STOP mode released by $\overline{\text{RESET}}$ input 266
Figure 9-14: STOP mode release by Watchdog reset, NMI, INT........ 267
Figure 9-15: Power Save Control Register (PSC) 268
Figure 9-16: Power Save Mode Register (PSM) 270
Figure 10-1: Block Diagram of Timer C 273
Figure 10-2: Timer C counter (TMC0) 274
Figure 10-3: Capture/Compare Register 0 (CCC00) 276
Figure 10-4: Capture/Compare Register 1 (CCC01) 276
Figure 10-5: Timer C control Register 0 (TMCC00) (1/2) 278
Figure 10-6: Timer C control Register 1 (TMCC01) (1/2) 280
Figure 10-7: Valid Edge Selection Register (SESC0)........ 282
Figure 10-8: Timing of basic operation of Timer C 283
Figure 10-9: Timing of interrupt operation after overflow 284
Figure 10-10: Timing of capture for pulse cycle measurement (rising edge)........ 285
Figure 10-11: Timing of capture for pulse width measurement (both edges) 286
Figure 10-12: Timing of cycle measurement operation 288
Figure 10-13: Timing of compare operation........ 289
Figure 10-14: Timing of interval timer operation 290
Figure 10-15: Timing of PWM output operation (overview) 291
Figure 10-16: Timing of PWM output operation (detail)........ 292
Figure 10-17: Multiplexed Inputs for Timer C Sub Oscillator Calibration Function 294
Figure 10-18: Block Diagram of Timer Dn (n = 0, 1)........ 296
Figure 10-19: Timer Dn counter register (TMDn) (n = 0, 1) 298
Figure 10-20: Timer Dn Compare Register (CMDn) (n = 0, 1) 299
Figure 10-21: Timing of Timer Dn Operation 300
Figure 10-22: Timer Dn Control Register (TMCDn) (n = 0, 1) 301
Figure 10-23: Timing of Compare Operation (1/2)........ 302
Figure 10-24: Block Diagram of Timer Gn 308
Figure 10-25: Timer Gn Counter 0 Value Registers TMGn0 310
Figure 10-26: Timer Gn Counter 1 Value Registers TMGn1 310
Figure 10-27: Timer Gn counter TMGn0 assigned Capture/Compare Register (GCCn0)........ 311
Figure 10-28: Timer Gn counter TMGn1 assigned Capture/Compare Register (GCCn5)........ 311
Figure 10-29: Timer Gn free assignable Capture/Compare Registers (GCCnm) (m = 1 to 4) 312
Figure 10-30: Timer Gn Mode Register (TMGMn) (1/2) 313
Figure 10-31: Timer Gn Mode Register Low (TMGMnL) 315
Figure 10-32: Timer Gn Mode Register Low (TMGMnH) 315
Figure 10-33: Timer Gn Channel Mode Register (TMGCMn)........ 316
Figure 10-34: Timer Gn Channel Mode Register (TMGCMnL)........ 317
Figure 10-35: Timer Gn Channel Mode Register (TMGCMnH) 317
Figure 10-36: Timer Gn Output Control Register (OCTLGn)........ 318
Figure 10-37: Timer Gn Output Control Register Low (OCTLGnL) 319
Figure 10-38: Timer Gn Output Control Register High (OCTLGnH)........ 319
Figure 10-39: Timer Gn Status Register (TMGSTn)........ 320
Figure 10-40: Timing of Output delay operation 321
Figure 10-41: Timing when both edges of TIGn0 are valid (free run)........ 326
Figure 10-42: Timing of capture trigger edge detection (free run) 327
Figure 10-43: Timing of starting capture trigger edge detection 328
Figure 10-44: Timing of compare mode (free run)........ 329
Figure 10-45: Timing when GCCn1 is rewritten during operation (free run)........ 330
Figure 10-46: Timing of PWM operation (free run) 332
Figure 10-47: Timing when 0000H is set in GCCnm (free run)........ 333
Figure 10-48: Timing when FFFFH is set in GCCnm (free run)........ 333
Figure 10-49: Timing when GCCnm is rewritten during operation (free run)........ 334
Figure 10-50: Timing when both edges of TIGm are valid (match and clear)........ 336
Figure 10-51: Timing of compare operation (match and clear)........ 338
Figure 10-52: Timing when GCCnm is rewritten during operation (match and clear)........ 339

Figure 10-53: Timing of PWM operation (match and clear) 341
Figure 10-54: Timing when 0000H is set in GCCnm (match and clear) 342
Figure 10-55: Timing when the same value as set in GCCn0/GCCn5 is set in GCCnm (match and clear) 343
Figure 10-56: Timing when the value of GCCnm exceeding GCCn0 or GCCn5 (match and clear) 344
Figure 10-57: Timing when GCCnm is rewritten during operation (match and clear) 345
Figure 10-58: Timing of Edge detection noise elimination 346
Figure 11-1: Block Diagram of Watch Timer 349
Figure 11-2: Watch Timer Mode Control Register (WTM) (1/2) 350
Figure 11-3: Operation Timing of Watch Timer/Interval Timer 355
Figure 12-1: Block Diagram of Watchdog Timer 357
Figure 12-2: Watchdog Timer Clock Selection Register (WDCS) 359
Figure 12-3: Watchdog Timer Mode Register (WDTM) 360
Figure 12-4: Watchdog Timer Mode Register (WCMD) 361
Figure 12-5: Watchdog Timer Mode Register (WPHS) 361
Figure 13-1: Asynchronous Serial Interfaces Block Diagram 366
Figure 13-2: Asynchronous Serial Interface Mode Registers (ASIM0, ASIM1) (1/3) 367
Figure 13-3: Asynchronous Serial Interface Status Registers (ASIS0, ASIS1) 370
Figure 13-4: Asynchronous Serial Interface Transmit Status Registers (ASIF0, ASIF1) 371
Figure 13-5: Reception Buffer Registers (RXB0, RXB1) 372
Figure 13-6: Transmission Buffer Registers (TXB0, TXB1) 373
Figure 13-7: Asynchronous Serial Interface Transmit/Receive Data Format 375
Figure 13-8: Asynchronous Serial Interface Transmission Completion Interrupt Timing 376
Figure 13-9: Continuous Transmission Starting Procedure 378
Figure 13-10: Continuous Transmission End Procedure 379
Figure 13-11: Asynchronous Serial Interface Reception Completion Interrupt Timing 380
Figure 13-12: When Reception Error Interrupt Is Separated from INTSRn Interrupt (ISRM Bit = 0) 381
Figure 13-13: When Reception Error Interrupt Is Included in INTSRn Interrupt (ISRM Bit = 1) 381
Figure 13-14: Noise Filter Circuit 383
Figure 13-15: Timing of RXD5n Signal Judged as Noise 383
Figure 13-16: Baud Rate Generator (BRG) Configuration of UART5n (n = 0, 1) 384
Figure 13-17: Clock Select Registers (CHKSR0, CHKSR1) 385
Figure 13-18: Baud Rate Generator Control Registers (BRGC0, BRGC1) 386
Figure 13-19: Allowable Baud Rate Range During Reception 389
Figure 13-20: Transfer Rate During Continuous Transmission 391
Figure 13-21: Block Diagram of Clocked Serial Interfaces 394
Figure 13-22: Clocked Serial Interface Mode Registers (CSIM0 to CSIM2) 395
Figure 13-23: Clocked Serial Interface Clock Selection Registers (CSIC0 to CSIC2) (1/2) 396
Figure 13-24: Clocked Serial Interface Reception Buffer Registers (SIRB0 to SIRB2) 398
Figure 13-25: Clocked Serial Interface Reception Buffer Registers Low (SIRBL0 to SIRBL2) 399
Figure 13-26: Clocked Serial Interface Read-Only Reception Buffer Registers (SIRBE0 to SIRBE2) 400
Figure 13-27: Clocked Serial Interface Read-Only Reception Buffer Registers Low (SIRBEL0 to SIRBEL1) 401
Figure 13-28: Clocked Serial Interface Transmission Buffer Registers (SOTB0 to SOTB2) 402
Figure 13-29: Clocked Serial Interface Transmission Buffer Registers Low (SOTBL0 to SOTBL2) 403
Figure 13-30: Clocked Serial Interface Initial Transmission Buffer Registers (SOTBF0 to SOTBF2) 404
Figure 13-31: Clocked Serial Interface Initial Transmission Buffer Registers Low (SOTBFL0 to SOTBFL2) 405
Figure 13-32: Serial I/O Shift Registers (SIO0 to SIO2) 406
Figure 13-33: Serial I/O Shift Registers Low (SIOL0 to SIOL2) 407
Figure 13-34: Timing Chart in single Transfer Mode (1/2) 409
Figure 13-35: Timing Chart According to Clock Phase Selection (1/2) 411

Figure 13-36: Timing Chart of Interrupt Request Signal Output in Delay Mode (1/2) 413
Figure 13-37: Repeat Transfer (Receive-Only) Timing Chart 415
Figure 13-38: Repeat Transfer (Transmission/Reception) Timing Chart 417
Figure 13-39: Timing Chart of Next Transfer Reservation Period (1/2) 418
Figure 13-40: Transfer Request Clear and Register Access Contention 420
Figure 13-41: Interrupt Request and Register Access Contention 421
Figure 13-42: Baud Rate Generators 0, 1 (BRG0, BRG1) Block Diagram 423
Figure 13-43: Prescaler Mode Registers 0, 1 (PRSM0, PRSM1) 424
Figure 13-44: Prescaler Compare Registers 0, 1 (PRSCM0, PRSCM1) 425
Figure 14-1: Functional Blocks of the FCAN Interface 428
Figure 14-2: Memory Area of the FCAN System 429
Figure 14-3: Clock Structure of the FCAN System 438
Figure 14-4: FCAN Interrupt Bundling of V850E/CA2 439
Figure 14-5: Time Stamp Capturing at Message Reception 441
Figure 14-6: Time Stamp Capturing at Message Transmission 442
Figure 14-7: 16-Bit Data Write Operation for Specific Registers 453
Figure 14-8: CAN Stop Register (CSTOP) 454
Figure 14-9: CAN Main Clock Select Register (CGSC) (1/2) 455
Figure 14-10: Configuration of FCAN System Main Clock 456
Figure 14-11: Configuration of FCAN Global Time System Clock 456
Figure 14-12: CAN Global Status Register (CGST) (1/3) 457
Figure 14-13: CAN Global Interrupt Enable Register (CGIE) (1/2) 460
Figure 14-14: CAN Timer Event Enable Register (CGTEN) 462
Figure 14-15: CAN Global Time System Counter and event generation 462
Figure 14-16: CAN Global Time System Counter (CGTSC) 463
Figure 14-17: CAN Message Search Start Register (CGMSS) 464
Figure 14-18: CAN Message Search Result Register (CGMSR) 465
Figure 14-19: CAN Test Bus Register (CTBR) 466
Figure 14-20: Internal CAN Test Bus Structure 466
Figure 14-21: CAN Interrupt Pending Registers (CCINTPL, CCINTPH) 467
Figure 14-22: CAN Global Interrupt Pending Register (CGINTP) (1/2) 468
Figure 14-23: CAN 1 to 4 Interrupt Pending Registers (C1INTP to C4INTP) (1/2) 470
Figure 14-24: Message Identifier Registers L00 to L31 and H00 to H31
(M_IDL00 to M_IDL31, M_IDH00 to M_IDH31) 472
Figure 14-25: Message Configuration Registers 00 to 31 (M_CONF00 to M_CONF31) (1/2) 473
Figure 14-26: Message Status Registers 00 to 31 (M_STAT00 to M_STAT31) 475
Figure 14-27: Message Set/Clear Status Registers 00 to 31 (SC_STAT00 to SC_STAT31) 477
Figure 14-28: Message Data Registers m0 to m7
(M_DATAm0 to M_DATAm7) (m = 00 to 31) (1/2) 478
Figure 14-29: Message Data Length Code Registers 00 to 31 (M_DLC00 to M_DLC31) 480
Figure 14-30: Message Control Registers 00 to 31 (M_CTRL00 to M_CTRL31) (1/2) 481
Figure 14-31: Message Time Stamp Registers 00 to 31 (M_TIME00 to M_TIME31) 483
Figure 14-32: Message Event Registers m0, m1, and m3
(M_EVTm0, M_EVTm1, M_EVTm2, M_EVTm3) (m = 00 to 31) 484
Figure 14-33: CAN 1 to 4 Mask 0 to 3 Registers L, H
(CxMASKL0 to CxMASKL3, CxMASKH0 to CxMASKH3) (x = 1 to 4) 485
Figure 14-34: CAN 1 to 4 Control Registers (C1CTRL to C4CTRL) (1/5) 487
Figure 14-35: CAN 1 to 4 Definition Registers (C1DEF to C4DEF) (1/4) 492
Figure 14-36: CAN 1 to 4 Information Registers (C1LAST to C4LAST) 496
Figure 14-37: CAN 1 to 4 Error Counter Registers (C1ERC to C4ERC) 497
Figure 14-38: CAN 1 to 4 Interrupt Enable Registers (C1IE to C4IE) (1/3) 498
Figure 14-39: CAN 1 to 4 Bus Activity Registers (C1BA to C4BA) (1/2) 501
Figure 14-40: CAN 1 to 4 Bit Rate Prescaler Registers (C1BRP to C4BRP) (1/2) 503
Figure 14-41: CAN Bus Bit Timing 505
Figure 14-42: CAN 1 to 4 Synchronization Control Registers (C1SYNC to C4SYNC) (1/2)) 506
Figure 14-43: CAN 1 to 4 Bus Diagnostic Information Registers (C1DINF to C4DINF) 508
Figure 14-44: Sequential CAN Data Read by CPU 512
Figure 14-45: State Transition Diagram for CAN Modules 514

Figure 14-46: General Initialisation Sequence for the CAN Interface 515
Figure 14-47: Initialisation Sequence for a CAN module 517
Figure 14-48: Setting CAN Module into Initialisation State 519
Figure 15-1: Block Diagram of A/D Converter 526
Figure 15-2: A/D Converter Mode Register (ADM) 528
Figure 15-3: A/D Converter Register (ADS) 530
Figure 15-4: A/D Conversion Result Register (ADCR) 531
Figure 15-5: A/D Conversion Result Register (ADCRL) 531
Figure 15-6: A/D Conversion Result Register (ADCRH) 532
Figure 15-7: Port Function Register (PORT7/PORT8) 533
Figure 15-8: Port Function Register 7 (PORT7) 534
Figure 15-9: Relation between Analog Input Voltage and A/D Conversion Result 536
Figure 15-10: No write operation is made to ADM or ADS register during A/D conversion operation 538
Figure 15-11: ADCS bit is cleared (0) during A/D conversion operation... 539
Figure 15-12: A write operation is made to the ADS register during A/D conversion operation 540
Figure 15-13: Analog Input Pin Handling 541
Figure 15-14: Overall Error 542
Figure 15-15: Quantization Error 543
Figure 15-16: Zero-Scale Error 543
Figure 15-17: Full-Scale Error.. 544
Figure 15-18: Nonlinearity Error... 544
Figure 16-1: Port Configuration... 546
Figure 16-2: Type A Block Diagram.. 551
Figure 16-3: Type B Block Diagram.. 552
Figure 16-4: Type C Block Diagram.. 553
Figure 16-5: Type D Block Diagram.. 554
Figure 16-6: Type E Block Diagram.. 555
Figure 16-7: Port 1 (P1) 556
Figure 16-8: Port 1 Mode Register (PM1) 557
Figure 16-9: Port 1 Mode Control Register (PMC1) 558
Figure 16-10: Port 2 (P2) 559
Figure 16-11: Port 2 Mode Register (PM2) 560
Figure 16-12: Port 2 Mode Control Register (PMC2) 561
Figure 16-13: Port 3 (P3) 562
Figure 16-14: Port 3 Mode Register (PM3) 563
Figure 16-15: Port 3 Mode Control Register (PMC3) 563
Figure 16-16: Port 4 (P4) 564
Figure 16-17: Port 4 Mode Register (PM4) 565
Figure 16-18: Port 4 Mode Control Register (PMC4) 565
Figure 16-19: Port 5 (P5) 566
Figure 16-20: Port 5 Mode Register (PM5) 567
Figure 16-21: Port 5 Mode Control Register (PMC5) 568
Figure 16-22: Port 6 (P6) 569
Figure 16-23: Port 6 Mode Register (PM6) 570
Figure 16-24: Port 6 Mode Control Register (PMC6) 571
Figure 16-25: Port Function Register 7 (P7) 572
Figure 16-26: Port Function Register 7/8 (P7/P8) 573
Figure 16-27: Port 9 (P9) 574
Figure 16-28: Port 9 Mode Register (PM9) 574
Figure 16-29: Port AH (PAH) 575
Figure 16-30: Port AH Mode Register (PMAH) 576
Figure 16-31: Port AH Mode Control Register (PMCAH) 576
Figure 16-32: Port CS (PCS) 577
Figure 16-33: Port CS Mode Register (PMCS) 578
Figure 16-34: Port CS Mode Control Register (PMCCS) 578
Figure 16-35: Port CT (PCT) 579
Figure 16-36: Port CT Mode Register (PMCT) 580

Figure 16-37: Port CT Mode Control Register (PMCCT) 580
Figure 16-38: Port CM (PCM) 581
Figure 16-39: Port CM Mode Register (PMCM) 582
Figure 16-40: Port CM Mode Control Register (PMCCM) 582
Figure 17-1: Reset signal acknowledgment 585
Figure 17-2: Reset at power-on 586
Figure A-1: How to Read Instruction Set List 589

List of Tables

Table 2-1: Port Pins 33
Table 2-2: Non-Port Pins 36
Table 2-3: Pin Status in Reset and Standby Mode 39
Table 2-4: Types of Pin I/O Circuit and Connection of Unused Pins 51
Table 3-1: Program Registers 57
Table 3-2: System Register Numbers 58
Table 3-3: Saturation-Processed Operation Result 61
Table 3-4: Operation Modes 63
Table 3-5: Interrupt/Exception Table 70
Table 3-6: List of Peripheral I/O Registers 76
Table 3-7: List of programmable peripheral I/O registers 85
Table 3-8: The Values of VSWC Register depending on System Clock 108
Table 4-1: Number of Bus Access Clocks 116
Table 4-2: Bus Priority Order 135
Table 7-1: Relationship Between Transfer Type and Transfer Object 193
Table 8-1: Interrupt/Exception Source List 200
Table 8-2: Addresses and Bits of Interrupt Control Registers 217
Table 8-3: Interrupt Response Time 237
Table 9-1: Power Saving Modes Overview 251
Table 9-2: Power Saving Mode Functions 254
Table 9-3: Power Saving Mode Functions 255
Table 9-4: Operating states in HALT mode 256
Table 9-5: Operation after HALT mode release by interrupt request 257
Table 9-6: Operating States in IDLE Mode 258
Table 9-7: Operating States in WATCH Mode 259
Table 9-8: Operation after WATCH mode release by interrupt request 260
Table 9-9: Operating States in WATCH Mode 261
Table 9-10: Operation after SUB WATCH mode release by interrupt request 262
Table 9-11: Operating States in STOP Mode 265
Table 10-1: Timer C Configuration List 274
Table 10-2: TOC0 Output Control 291
Table 10-3: Timer Dn Configuration List (n = 0, 1) 297
Table 10-4: Timer Gn Configuration List 309
Table 10-5: Interrupt output and timer output states dependent on the register setting values 322
Table 10-6: Interrupt output and timer output states dependent on the register setting values 323
Table 11-1: Configuration of Watch Timer 350
Table 11-2: Selection of the Watch Timer Clock 352
Table 11-3: Example for Interval Time of Watch Timer 353
Table 11-4: Example for Interval Time of Interval Timer 354
Table 12-1: Watchdog Timer Configuration 358
Table 13-1: Generated Interrupts and Default Priorities 374
Table 13-2: Transmission Status and Whether or Not Writing Is Enabled 377
Table 13-3: Reception Error Causes 381
Table 13-4: Baud Rate Generator Setting Data 388
Table 13-5: Maximum and Minimum Allowable Baud Rate Error 390
Table 13-6: Baud Rate Generator Setting Data 426
Table 14-1: Configuration of the CAN Message Buffer Section 430
Table 14-2: CAN Message Buffer Registers Layout 431
Table 14-3: Relative Addresses of CAN Interrupt Pending Registers 432
Table 14-4: Relative Addresses of CAN Common Registers 433
Table 14-5: Relative Addresses of CAN Module 1 Registers 434
Table 14-6: Relative Addresses of CAN Module 2 Registers 435
Table 14-7: Relative Addresses of CAN Module 3[Note1] Registers 436

Table 14-8: Relative Addresses of CAN Module 4 Registers 437
Table 14-9: Transmitted Data On the CAN Bus (ATS = 1) 442
Table 14-10: Example for Automatic Transmission Priority Detection 444
Table 14-11: Example for Transmit Buffer Allocation When More Than 5 Buffers Linked to a CAN Module 445
Table 14-12: Storage Priority for Reception of Data Frames 446
Table 14-13: Storage priority for Reception of Remote Frames 446
Table 14-14: Inner Storage Priority Within a Priority Class 447
Table 14-15: Remote Frame Handling upon Reception into a Transmit Message Buffer 451
Table 14-16: CAN Message Processing by TRQ and RDY Bits 476
Table 14-17: Address Offsets of the CAN 1 to 4 Mask Registers 486
Table 15-1: A/D Converter Configuration 524
Table 15-2: Register format of A/D Converter Control Register 527
Table 16-1: Functions of each port 547
Table 16-2: Port Pin Functions 548
Table 17-1: Operation Status of each pin during Reset period 584
Table 17-2: Initial Values of CPU and Internal RAM After Reset 588
Table A-1: Symbols in Operand Description 590
Table A-2: Symbols Used for Op Code 590
Table A-3: Symbols Used for Operation Description 591
Table A-4: Symbols Used for Flag Operation 591
Table A-5: Condition Codes 592
Table A-6: Instruction Set List 593

Chapter 1 Introduction

The V850E/CA2 Jupiter is a product in NEC's V850 family of ROM-less microcontrollers designed for Automotive applications.

1.1 General

The V850E/CA2 Jupiter Rom-less microcontroller, is a member of NEC's V850 32-bit RISC family, which match the performance gains attainable with RISC-based controllers to the needs of embedded control applications. The V850 CPU offers easy pipeline handling and programming, resulting in compact code size comparable to 16-bit CISC CPUs.
The V850E/CA2 Jupiter offers an excellent combination of general purpose peripheral functions, like serial communication interfaces (UART, clocked SI) and measurement inputs (A/D converter), with dedicated CAN network support.
The device offers power-saving modes to manage the power consumption effectively under varying conditions.
Thus equipped, the V850E/CA2 Jupiter is ideally suited for automotive applications, like dashboard or body. It is also an excellent choice for other applications where a combination of sophisticated peripheral functions and CAN network support is required.

(1) V850E CPU

The V850E CPU supports the RISC instruction set, and through the use of basic instructions that can each be executed in 1-clock period and an optimized pipeline, achieves marked improvements in instruction execution speed. In addition, in order to make it ideal for use in digital servo control, a 32-bit hardware multiplier enables this CPU to support multiply instructions, saturated multiply instructions, bit operation instructions, etc.
Also, through 2-byte basic instructions and instructions compatible with high level languages, etc., object code efficiency in a C compiler is increased, and program size can be made more compact.
Further, since the on-chip interrupt controller provides high speed interrupt response, including processing, this device is suited for high level real time control fields.

(2) External memory interface function

The V850E/CA2 contains a non multiplexed external bus interface, including an address bus (24 bits) and data bus (16 bits). SRAM and ROM can be connected as well as page ROM memories.
The DMA controller allows, data transfers between internal RAM and peripheral I/O. This reduces the CPU load.

(3) A full range of development environment products

A development environment system that includes an optimized C compiler, debugger, in-circuit emulator, simulator, system performance analyzer, and other elements is also available.

1.2 Device Features

- CPU

  - Core: V850E1
  - Number of instructions: 81
  - Min. instruction execution time: 31.25 ns (@ ϕ = 32 MHz)
  - General registers: 32 bits $\times$ 32

- Instruction set:
  - V850E (compatible with V850 plus additional powerful instructions for reducing code and increasing execution speed)
  - Signed multiplication
    (16 bits x 16 bits $\rightarrow$ 32 bits or 32 bits x 32 bits $\rightarrow$ 64 bits): 1 to 2 clocks
  - Saturated operation instructions (with overflow/underflow detection function)
  - 32-bit shift instructions: 1 clock
  - Bit manipulation instructions
  - Load/store instructions with long/short format
  - Signed load instructions

- Memory

Part Number	Internal ROM	Internal RAM	Full-Can (2.0b active)	Full-CAN RAM
µPD703128 (A)	Rom-less	12 Kbytes	2 Channels	1 Kbytes (32 message buffers)
µPD703129 (A)	Rom-less	16 Kbytes	4 Channels	1 Kbytes (32 message buffers)
µPD703129 (A1)	Rom-less	16 Kbytes	4 Channels	1 Kbytes (32 message buffers)

- Cache Controller. 2-way associative 4K Bytes

- Boot Loader
  - Internal Boot Loader for downloading Flash-Self-Programming routines into RAM
  - Support of virgin programming for external flash memories

- Clock Generator
  - Internal Spread-Spectrum PLL
  - (CPU Core/ BCU clock supply)
  - Internal PLL (Peripheral clock supply) 4 fold PLL
  - Frequency range: up to 32 MHz
  - Crystal frequency range: 4 MHz - 5 MHz
  - Internal “Slow-Running” clock oscillator

- Built-in power saving modes: WATCH, HALT, STOP

- Power supply voltage range V_{DD5} $4.5\ V \leq V_{DD5} \leq 5.5\ V$

- Power supply voltage range V_{DD3} $3.0\ V \leq V_{DD3} \leq 3.6\ V$

- Temperature range: Ta = - 40 to + 85°C @ ϕ = 32 MHz
  Ta = - 40 to + 110°C @ ϕ = 20 MHz

- Bus control unit:
  - Address/data separated bus — 24-bit address/ 16-bit data bus
  - Supply voltage power for Bus Interface — 3.3 V
  - Chip Select Signals — 3

- DMA control unit: — 4 channels

- I/O lines (5 V): — 51

- Input lines (5 V): — 12

- I/O lines (3.3 V): — 15

- A/D Converter: — 10-bit resolution; 12 channels
  - Select Mode
  - AV_{REF} switchable “On/Off” by Software
  - Analog input channels shared with port functionality

- Serial Interfaces
  - 3-wire mode: — 3 channels
  - UART mode: — 2 channels
  - Full CAN Interface (2.0b active) — 2 - 4 channels

- Timers
  - 16-bit multi purpose timer/event counter: — 2 channel
  - 16-bit multi purpose timer/counter: — 1 channel
  - 16-bit OS timer: — 2 channel
  - Watch timer: — 1 channel
  - Watchdog timer: — 1 channel

- Interrupts and exceptions
  - Non-maskable interrupts: — 2 source
  - Maskable interrupts: — 57 sources (µPD703128)
    63 sources (µPD703129)
  - Software exceptions: — 32 sources
  - Exception trap: — 1 source

- Clock Correction of Sub-Oscillator

- Package — 144 QFP, 0.5 mm pin-pitch

- CMOS technology

Remark: The CAN macro of this device fulfils the requirements according ISO 11898. Additionally the CAN macro was tested according to the test procedures required by ISO 16845. The CAN macro successfully passed all test patterns. Beyond these test patterns, other tests like robustness tests and processor interface tests as recommended by C&S/FH Wolfenbuettel have successfully been issued.

1.3 Application Fields

The V850E CA2/ Jupiter is ideally suited for automotive applications, like dashboard or gateway applications. It is also an excellent choice for other applications where a combination of sophisticated peripheral functions and CAN network support is required.

1.4 Ordering Information

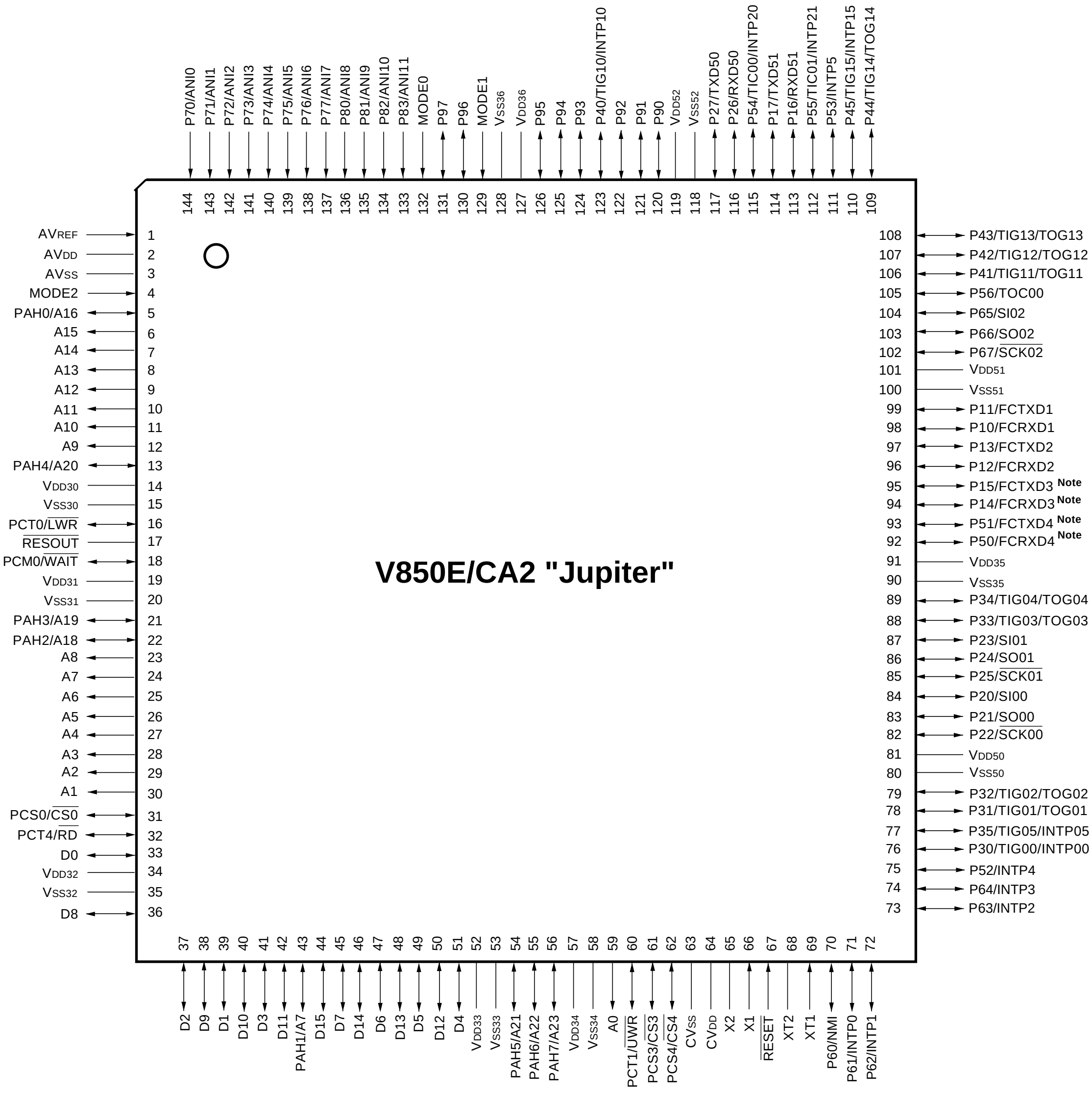
Part number	Package	Internal ROM [bytes]	Internal RAM [bytes]	Full-CAN RAM [bytes] / Channels
µPD703128 (A)	144-pin QFP (fine pitch) (20 × 20 mm)	Rom-less	12 K	1K/ 32 message buffers 2 FCAN Channels
µPD703129 (A)	144-pin QFP (fine pitch) (20 × 20 mm)	Rom-less	16 K	1K/ 32 message buffers 4 FCAN Channels
µPD703129 (A1)	144-pin QFP (fine pitch) (20 × 20 mm)	Rom-less	16 K	1K/ 32 message buffers 4 FCAN Channels

1.5 Pin Configuration (Top View)

- 144 pin QFP (fine pitch) (20 × 20 mm)

  - µPD703128 (A)
  - µPD703129 (A)
  - µPD703129 (A1)

Figure 1-1: V850E/CA2 Jupiter Pin Configuration



Note: FCRXD3, FCTXD3, FCRXD4 and FCTXD4 are available only in the derivatives µPD703129 (A) and µPD703129 (A1).

Pin Identification

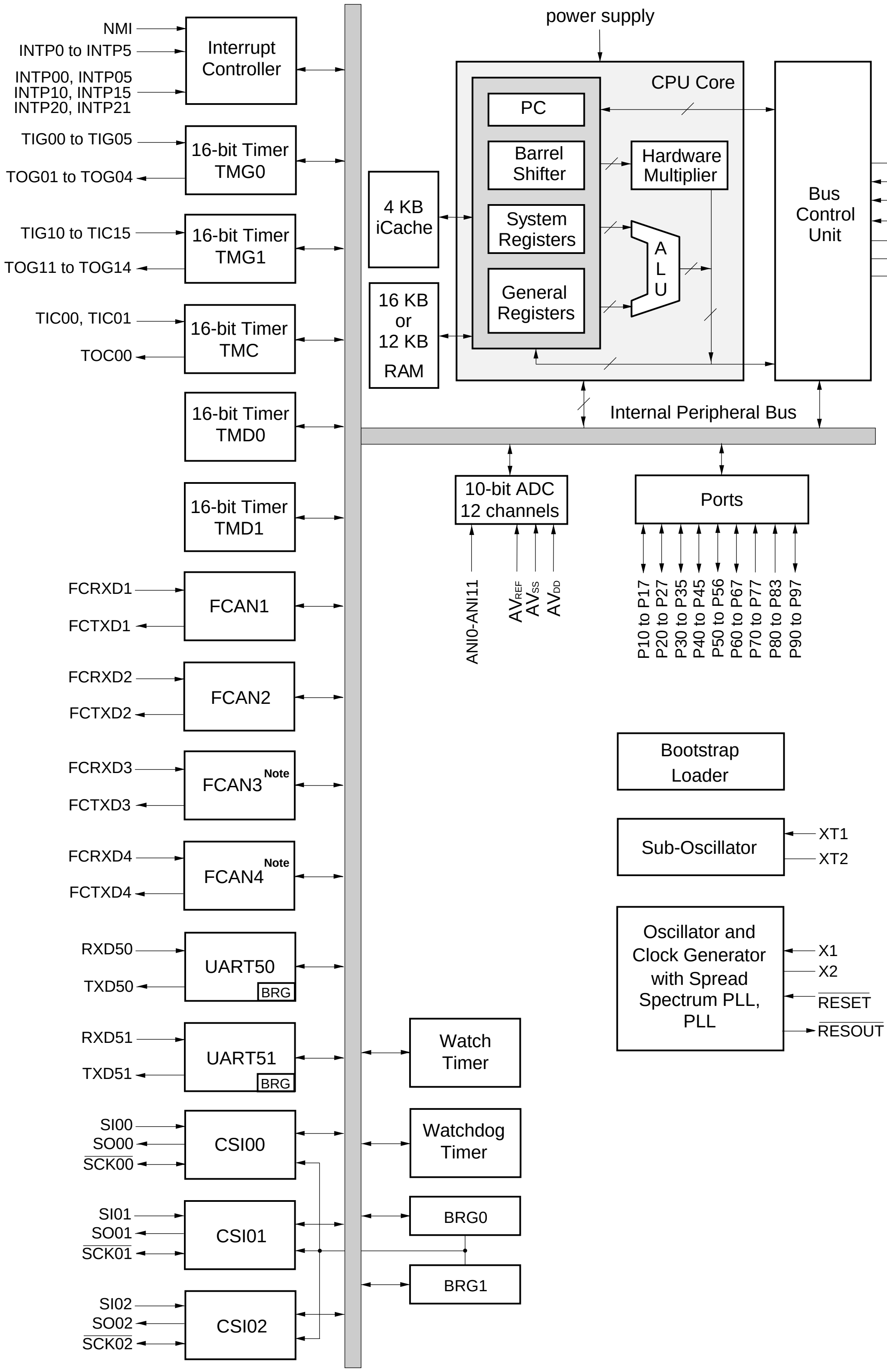
A0 to A23	: Address Bus	PAH0 to PAH7	: Port AH
D0 to D15	: Data Bus	PCM0	: Port CM0
ANI00 to ANI11	: Analog Input	PCS0, PCS3, PCS4	: Port CS
AV_{DD}	: Analog Power Supply	PCT0, PCT1, PCT4	: Port CT
AV_{REF}	: Analog Reference Voltage	$\overline{\text{RESET}}$	: Reset
AV_{SS}	: Analog Ground	$\overline{\text{RESOUT}}$	: Reset Out
FCRXD1 to FCRXD4	: CAN Receive Line Input	RXD50 to RXD51	: Receive Data Input
FCTXD1 to FCTXD4	: CAN Transmit Line Output	$\overline{\text{SCK00}}$, $\overline{\text{SCK01}}$, $\overline{\text{SCK02}}$	: Serial Clock
CV_{DD}	: Clock Generator Power Supply	SI00, SI01, SI02	: Serial Input
CV_{SS}	: Clock Generator Ground	SO00, SO01, SO02	: Serial Output
GND_{30} to GND_{36}	Ground for 3 V Power Supply	TIG00 to TIG05, TIG10 to TIG05, TIC00, TIC01	: Timer Input
GND_{50} to GND_{52}	Ground for 5 V Power Supply	TOG01 to TOG04, TOG11 to TOG14, TOC00	Timer Output
INTP0 to INTP5	External interrupt request	TXD50 to TXD51	: Transmit Data Output
INTPn0, INTPn5, INTP2n	: Interrupt Request from Peripherals	V_{DD30} to V_{DD36}	: 3 V Power Supply
MODE0 to MODE2	: Mode Inputs	V_{DD50} to V_{DD52}	: 5 V Power Supply
NMI	: Non-Maskable Interrupt Request	$\overline{\text{WAIT}}$	: Wait
P10 to P17	: Port 1	$\overline{\text{LWR}}$, $\overline{\text{UWR}}$	Write Enable
P20 to P27	: Port 2	$\overline{\text{RD}}$	: Read
P30 to P35	: Port 3	$\overline{\text{CS0}}$, $\overline{\text{CS3}}$, $\overline{\text{CS4}}$	Chip Select
P40 to P45	: Port 4	X1, X2	: Crystal (Main-OSC)
P50 to P56	: Port 5	XT1, XT2	: Crystal (Sub-OSC)
P60 to P67	Port 6		
P70 to P77	Port 7		
P80 to P83	Port 8		
P90 to P97	Port 9		

Note: FCRXD3, FCTXD3, FCRXD4 and FCTXD4 are available only in the derivatives µPD703129 (A) and µPD703129 (A1).

Remark: n = 0, 1

1.6 Function Block Diagram

Figure 1-2: V850E/CA2 Jupiter Block Diagram



Note: FCRXD3, FCTXD3, FCRXD4 and FCTXD4 are available only in the derivatives µPD703129 (A) and µPD703129 (A1).

1.6.1 On-chip units

(1) CPU

The CPU uses five-stage pipeline control to enable single-clock execution of address calculations, arithmetic logic operations, data transfers, and almost all other instruction processing.
Other dedicated on-chip hardware, such as the multiplier (16 bits x 16 bits → 32 bits or 32 bits x 32 bits → 64 bits) and the barrel shifter (32 bits), help accelerate processing of complex instructions.

(2) Bus control unit (BCU)

BCU starts a required external bus cycle based on the physical address obtained by the CPU. When an instruction is fetched from external memory area and the CPU does not send a bus cycle start request, the BCU generates a prefetch address and prefetches the instruction code. The prefetched instruction code is stored in an instruction queue in the CPU.
The BCU provides a page ROM controller (ROMC) and a DMA controller (DMAC).

(a) Page ROM controller (ROMC)
This controller supports accessing ROM that includes the page access function.
It performs address comparisons with the immediately preceding bus cycle and executes wait control for normal access (off page)/page access (on page). It can handle page widths of 8 to 128 bytes.

(b) DMA controller (DMAC)
Instead of the CPU, this controller controls data transfer between memory and I/O.
There is one address mode: 2-cycle transfer and there are three bus modes: single transfer, single step transfer, and block transfer.

(3) ROM

The µPD703128, µPD703129 is a ROM-less MCU containing a 16-bit wide non-multiplexed bus interface to be able to fetch instructions/data from external memories.

(4) RAM

RAM are mapped from address FFFF8000H.
During instruction fetch, data can be accessed from the CPU in 1-clock cycles.

(5) Interrupt controller (INTC)

This controller handles hardware interrupt requests (NMI, INTP0 to INTP5) from on-chip peripheral I/O and external hardware. Eight levels of interrupt priorities can be specified for these interrupt requests, and multiple-interrupt servicing control can be performed for interrupt sources.

(6) Spread spectrum Clock generator (SSCG)

The spread spectrum clock generator (SSCG) generates a spread spectrum system clock for the CPU/BCU system based on the main-oscillator input clock. Four types of clocks are generated ($f_{XX}/8$, $f_{XX}/6$, $f_{XX}/4$, $f_{XX}/3$), and can be supplied as the operating clock for the CPU/BCU (f_{CPU}).

(7) Clock generator (CG)

The clock generator includes two types of oscillators (Main-OSC and Sub-OSC). The Peripheral PLL can also be used as the clock supply for the CPU/BCU (f_{XXP}, $f_{XXP}/2$).
The peripherals can be supplied with the clock f_{XXP} or $f_{XXP}/2$.

(8) Real-time pulse unit (RPU)

This unit has 3 channels of 16-bit multi purpose timer/event counter and 2 channels of 16-bit interval timer built in, and it is possible to measure pulse widths or frequency and to output a programmable pulse.

(9) Serial interface (SIO)

A 2-channel asynchronous serial interface (UART), 3-channel clocked serial interface (CSI), and 4-channel FCAN are provided as serial interface.

UART transfers data by using the TXDn and RXDn pins. (n = 0 - 1)
CSI transfers data by using the SOn, SIn, and SCKn pins. (n = 0 - 2)
FCAN performs data transfer using CTXDn and CRXDn pins. (n = 1 - 4)

(10) A/D converter (ADC)

One high-resolution 10-bit A/D converter, it includes 12 analog input pins. Conversion uses the successive approximation method.

(11) Ports

As shown below, the following ports have general port functions and control pin functions.

Port	Port Function	Control Function
Port 1	8-bit input/output	Serial interface input/output
Port 2	8-bit input/output	Serial interface input/output
Port 3	6-bit input/output	Real-time pulse unit input/output, external interrupt input, PWM output
Port 4	6-bit input/output	Real-time pulse unit input/output, external interrupt input, PWM output
Port 5	7-bit input/output	Serial interface input/output, external interrupt input
Port 6	8-bit input/output	Serial interface input/output, external interrupt input
Port 7	8-bit input	A/D converter analog input
Port 8	4-bit input	A/D converter analog input
Port 9	8-bit input/output	-
Port AH	8-bit input/output	External address bus
Port CS	3-bit input/output	External bus interface control signal output ($\overline{CS0}$, $\overline{CS3}$, $\overline{CS4}$)
Port CT	3-bit input/output	External bus interface control signal output ($\overline{LWR}$, $\overline{UWR}$, $\overline{RD}$)
Port CM	1-bit input/output	Wait insertion signal input ($\overline{WAIT}$)

[MEMO]

Chapter 2 Pin Functions

2.1 List of Pin Functions

The names and functions of this product’s pins are listed below. These pins can be divided into port pins and non-port pins according to their functions.

(1) Port pins

Table 2-1: Port Pins (1/3)

Port	I/O	Function	Alternate
P10	I/O	Port 1 8-bit input/output port	FCRXD1
P11			FCTXD1
P12			FCRXD2
P13			FCTXD2
P14			FCRXD3**Note**
P15			FCTXD3**Note**
P16			RXD1
P17			TXD1
P20	I/O	Port 2 8-bit input/output port	SI0
P21			SO0
P22			$\overline{\text{SCK0}}$
P23			SI1
P24			SO1
P25			$\overline{\text{SCK1}}$
P26			RXD0
P27			TXD0
P30	I/O	Port 3 6-bit input/output port	TIG00/INTP00
P31			TIG01/TOG01
P32			TIG02/TOG02
P33			TIG03/TOG03
P34			TIG04/TOG04
P35			TIG05/INTP05
P40	I/O	Port 4 6-bit input/output port	TIG10/INTP10
P41			TIG11/TOG11
P42			TIG12/TOG12
P43			TIG13/TOG13
P44			TIG14/TOG14
P45			TIG15/INTP15

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Table 2-1: Port Pins (2/3)

Port	I/O	Function	Alternate
P50	I/O	Port 5 7-bit input/output port	FCRXD4[Note]
P51			FCTXD4[Note]
P52			INTP4
P53			INTP5
P54			TI0/INTP20
P55			TI1/INTP21
P56			TO0
P60	I/O	Port 6 8-bit input/output port	$\overline{\text{NMI}}$
P61			INTP0
P62			INTP1
P63			INTP2
P64			INTP3
P65			SI2
P66			SO2
P67			$\overline{\text{SCK2}}$
P70	I	Port 7 8-bit input port	ANI0
P71			ANI1
P72			ANI2
P73			ANI3
P74			ANI4
P75			ANI5
P76			ANI6
P77			ANI7
P80	I	Port 8 4-bit input port	ANI8
P81			ANI9
P82			ANI10
P83			ANI11
P90	I/O	Port 9 8-bit input/output port	-
P91			-
P92			-
P93			-
P94			-
P95			-
P96			-
P97			-

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Table 2-1: Port Pins (3/3)

Port	I/O	Function	Alternate
PAH0	I/O	Port AH 8-bit input/output port	A16
PAH1			A17
PAH2			A18
PAH3			A19
PAH4			A20
PAH5			A21
PAH6			A22
PAH7			A23
PCS0	I/O	Port CS 3-bit input/output port	$\overline{\text{CS0}}$
PCS3			$\overline{\text{CS3}}$
PCS4			$\overline{\text{CS4}}$
PCT0	I/O	Port CT 3-bit input/output port	$\overline{\text{LWR}}$
PCT1			$\overline{\text{UWR}}$
PCT4			$\overline{\text{RD}}$
PCM0	I/O	Port CM 1-bit input/output port	$\overline{\text{WAIT}}$

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

(2) Non-port pins

Table 2-2: Non-Port Pins (1/3)

Pin Name	I/O	Function	Alternate
V_{DD50}-V_{DD52}	–	Power supply 5 V	-
V_{SS50}-V_{SS52}	–	GND potential for 5 V power supply	-
V_{DD30}-V_{DD36} Note 1	–	Power supply 3.3 V	-
V_{SS30}-V_{SS36}	–	GND potential for 3.3 V power supply	-
CV_{DD} Note 2	–	Connection for 3.3 V clock oscillator power supply	-
CV_{SS}		GND potential for 3.3 V clock oscillator power supply	-
X1	input	System clock oscillator connection pins.	-
X2	output		-
XT1	input	Sub clock oscillator connection pins.	-
XT2	output		-
MODE0-MODE2	input	Selects operating mode	-
$\overline{\text{RESET}}$	input	System reset input	-
$\overline{\text{RESOUT}}$	output	System reset output (incl. Watchdog timer reset)	-
AV_{DD}	–	Power supply for A/D converter	-
AV_{SS}	–	Ground potential for A/D converter	-
AV_{REF}	input	reference voltage input for A/D converter	-
NMI	input	non maskable interrupt input	P60
ANI0-ANI7	input	analog input to A/D converter	P77 to P70
ANI8-ANI11	input	analog input to A/D converter	P80 to P83
SI00	input	serial receive data input to CSI00-CSI02	P20
SI01			P23
SI02			P65
SO00	output	serial transmit data output from CSI00-CSI02	P21
SO01			P24
SO02			P66
$\overline{\text{SCK00}}$	I/O	serial clock I/O from/to CSI00-CSI02	P22
$\overline{\text{SCK01}}$			P25
$\overline{\text{SCK02}}$			P67
RXD50	input	serial receive data input to UART50-UART51	P26
RXD51			P16
TXD50	output	serial transmit data output from UART50-UART51	P27
TXD51			P17

Notes: 1. All V_{DD3} pins have to be connected to each other. On each pin of V_{DD3}, a capacitor containing a very low serial impedance has to be attached as tight as possible to the pin.

2. On CV_{DD}, a capacitor containing a very low serial impedance has to be attached as tight as possible to the pin. V_{DD3} and CV_V must be connected to each other.

3. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Table 2-2: Non-Port Pins (2/3)

Pin Name	I/O	Function	Alternate
FCRXD1	input	serial receive data input to FCAN1-FCAN4	P10
FCRXD2			P12
FCRXD3 Note 3			P14
FCRXD4 Note 3			P50
FCTXD1	output	serial transmit data output to FCAN1-FCAN4	P11
FCTXD2			P13
FCTXD3 Note 3			P15
FCTXD4 Note 3			P51
INTP0	input	external interrupt request	P61
INTP1			P62
INTP2			P63
INTP3			P64
INTP4			P52
INTP5			P53
INTP00	input	Timer G 0 external interrupt 0	P30/TIG00
INTP05	input	Timer G 0 external interrupt 5	P35/TIG05
INTP10	input	Timer G 1 external interrupt 0	P40/TIG10
INTP15	input	Timer G 1 external interrupt 5	P45/TIG15
INTP20	input	Timer C 0 external interrupt 0	P54/TIC00
INTP21	input	Timer C 0 external interrupt 1	P55/TIC01
TIG00	input	Timer G 0 capture input 0	P30/INTP00
TIG01	input	Timer G 0 capture input 1	P31/TOG01
TIG02	input	Timer G 0 capture input 2	P32/TOG02
TIG03	input	Timer G 0 capture input 3	P33/TOG03
TIG04	input	Timer G 0 capture input 4	P34/TOG04
TIG05	input	Timer G 0 capture input 5	P35/TOG05
TOG01	output	Timer G 0 compare output 1	P31/TIG01
TOG02	output	Timer G 0 compare output 2	P32/TIG02
TOG03	output	Timer G 0 compare output 3	P33/TIG03
TOG04	output	Timer G 0 compare output 4	P34/TIG04
TIG10	input	Timer G 1 capture input 0	P40/INTP10
TIG11	input	Timer G 1 capture input 1	P41/TOG11
TIG12	input	Timer G 1 capture input 2	P42/TOG12
TIG13	input	Timer G 1 capture input 3	P43/TOG13
TIG14	input	Timer G 1 capture input 4	P44/TOG14

Notes: 1. All V_{DD3} pins have to be connected to each other. On each pin of V_{DD3}, a capacitor containing a very low serial impedance has to be attached as tight as possible to the pin.

2. On CV_{DD}, a capacitor containing a very low serial impedance has to be attached as tight as possible to the pin. V_{DD3} and CV_V must be connected to each other.

3. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Table 2-2: Non-Port Pins (3/3)

Pin Name	I/O	Function	Alternate
TIG15	input	Timer G 1 capture input 5	P45/TOG15
TOG11	output	Timer G 1 compare output 1	P41/TIG11
TOG12	output	Timer G 1 compare output 2	P42/TIG12
TOG13	output	Timer G 1 compare output 3	P43/TIG13
TOG14	output	Timer G 1 compare output 4	P44/TIG14
TIC00	input	Timer C 0 capture input 0	P54/INTP20
TIC01	input	Timer C 0 capture input 1	P55/INTP21
TOC0	output	Timer C0 compare output	P56
D0-D15	I/O	Data bus of external bus	-
A0-A7			-
A8-A15	output	Address bus of external bus	-
A16-A23			PAH0-PAH7
$\overline{\text{LWR}}$	output	Write strobe signal for lower byte (bit 0 - bit 7)	PCT0
$\overline{\text{UWR}}$	output	Write strobe signal for upper byte (bit 0 - bit 7)	PCT1
$\overline{\text{RD}}$	output	Read strobe signal for external bus	PCT4
$\overline{\text{WAIT}}$	input	Control signal input for external bus	PCM0
$\overline{\text{CS0}}$			PCS0
$\overline{\text{CS3}}$	output	Chip select output for external bus	PCS3
$\overline{\text{CS4}}$			PCS4

Notes: 1. All V_{DD3} pins have to be connected to each other. On each pin of V_{DD3}, a capacitor containing a very low serial impedance has to be attached as tight as possible to the pin.

2. On CV_{DD}, a capacitor containing a very low serial impedance has to be attached as tight as possible to the pin. V_{DD3} and CV_V must be connected to each other.

3. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

(3) Pin status in RESET and STANDBY mode

Table 2-3: Pin Status in Reset and Standby Mode

Pin / Operating Status	RESET	STOP	WATCH	Sub-WATCH	IDLE	HALT	Idle state (TI)
D0 to D15	Hi-Z	Hi-Z/--[1]	Hi-Z/--[1]	Hi-Z/--[1]	Hi-Z/--[1]	operate	operate
A0 to A23	Hi-Z	HOLD	HOLD	HOLD	HOLD	operate	operate
$\overline{\text{CS4}}$, $\overline{\text{CS3}}$, $\overline{\text{CS0}}$	Hi-Z	H	H	H	H	operate	operate
$\overline{\text{UWR}}$, $\overline{\text{LWR}}$	Hi-Z	H	H	H	H	operate	operate
$\overline{\text{RD}}$	Hi-Z	H	H	H	H	operate	operate
$\overline{\text{WAIT}}$	--	--	--	--	--	operate	operate
$\overline{\text{RESOUT}}$	LOW	HIGH	HIGH	HIGH	HIGH	HIGH	HIGH
TIG05 to TIG00	N.A.	--	--	--	--	operate	operate
TIG15 to TIG10	N.A.	--	--	--	--	operate	operate
TIC01 to TIC00	N.A.	--	--	--	--	operate	operate
INTP05 to INTP00	N.A	operate	operate	operate	operate	operate	operate
INTP15 to INTP10	N.A	operate	operate	operate	operate	operate	operate
INTP21 to INTP20	N.A	operate	operate	operate	operate	operate	operate
INTP5 to INTP0	N.A	operate	operate	operate	operate	operate	operate
NMI	N.A	operate	operate	operate	operate	operate	operate
TOG04 to TOG01	N.A	HOLD	HOLD	HOLD	HOLD	operate	operate
TOG14 to TOG11	N.A	HOLD	HOLD	HOLD	HOLD	operate	operate
TOC0	N.A	HOLD	HOLD	HOLD	HOLD	operate	operate
SO02, SO01, SO00	N.A.	HOLD	HOLD	HOLD	HOLD	operate	operate
SI02, SI01, SI00	N.A.	--	--	--	--	operate	operate
$\overline{\text{SCK2}}$, $\overline{\text{SCK1}}$, $\overline{\text{SCK0}}$	N.A.	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate
RXD51 to RXD50	N.A.	--	--	--	--	operate	operate
TXD51 to TXD50	N.A.	HOLD	HOLD	HOLD	HOLD	operate	operate
FCRXD4[Note] to FCRXD1	N.A.	--	--	--	--	operate	operate
FCTXD4[Note] to FCTXD1	N.A.	HOLD	HOLD	HOLD	HOLD	operate	operate
ANI11 to ANI0	--	--	--	--	--	operate	operate
P1, P2, P3, P4, P5, P6, P9	Hi-Z	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate
PAH7 to PAH0	N.A	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate
PCS4, PCS3, PCS0	N.A	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate
PCT4, PCT1, PCT0	N.A	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate
PCM0	N.A	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	HOLD/--[1]	operate	operate

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Remarks:
1. N.A. : This configuration is not available
2. -- : Input data is not sampled
3. [1] : During output / input
4. Hi-Z : High Impedance

2.2 Description of Pin Functions

(1) P10 to P17 (Port 1) … Input/output

Port 1 is an 8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode, P10 to P17 operate as the serial interface (UART1, FCAN1, FCAN2, FCAN3[NOTE]) input/output.
An operation mode of port or control mode can be selected for each bit and specified by the port 1 mode control register (PMC1).

(a) Port mode
P10 to P17 can be set to input or output in 1-bit units using the port 1 mode register (PM1).

(b) Control mode
P10 to P17 can be set to port or control mode in 1-bit units using PMC1.

(c) CTXD1, CTXD2, CTXD3 (Transmit data for controller area network) … Output
This pin outputs FCAN serial transmit data.

(d) CRXD1, CRXD2, CRXD3 (Receive data for controller area network) … Input
This pin inputs FCAN serial receive data.

(e) TXD1 (Transmit data) … Output
This pin output serial transmit data of UART1.

(f) RXD1 (Receive data) … Input
This pin input serial receive data of UART1.

Note: CAN module 3 is available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

(2) P20 to P27 (Port 2) … Input/output

Port 2 is an 8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P20 to P27 operate as the serial interface (UART0, CSI0,CS1) input/output.
An operation mode of port or control mode can be selected for each bit and specified by the port 2 mode control register (PMC2).

(a) Port mode
P20 to P27 can be set to input or output in 1-bit units using the port 2 mode register (PM2).

(b) Control mode
P20 to P27 can be set to port or control mode in 1-bit units using PMC2.

(c) SO0, SO1 (Serial output) … Output
These pins output CSI0 and CSI1 serial transmit data.

(d) SI0, SI1 (Serial input) … Input
These pins input CSI0 and CSI1 serial receive data.

(e) $\overline{\text{SCK0}}$, $\overline{\text{SCK1}}$ (Serial clock) … Input/output
These are CSI0 and CSI1 serial clock input/output pins.

(f) TXD0 (Transmit data) … Output
This pin output serial transmit data of UART0.

(g) RXD0 (Receive data) … Input
This pin input serial receive data of UART0.

(3) P30 to P35 (Port 3) … Input/output

Port 3 is a 6-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P30 to P35 operate as the real-time pulse unit (RPU) input/output and external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 3 mode control register (PMC3).

(a) Port mode

P30 to P35 can be set to input or output in 1-bit units using the port 3 mode register (PM3).

(b) Control mode

P30 to P35 can be set to port or control mode in 1-bit units using PMC3.

(c) TOG01 to TOG04 (Timer output) … Output

These pins output a timer G 0 pulse signal.

(d) TIG00 to TIG05 (Timer input) … Input

These pins are the timer G 0 external capture trigger input pins.

(e) INPT00, INTP05 (Interrupt request from peripherals) … Input

These are external interrupt request input pins.

(4) P40 to P45 (Port 4) … Input/output

Port 4 is a 6-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P40 to P45 operate as the real-time pulse unit (RPU) input/output and external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 4 mode control register (PMC4).

(a) Port mode
P40 to P45 can be set to input or output in 1-bit units using the port 4 mode register (PM4).

(b) Control mode
P40 to P45 can be set to port or control mode in 1-bit units using PMC4.

(c) TOG11 to TOG14 (Timer output) … Output
These pins output a timer G 1 pulse signal.

(d) TIG10 to TIG15 (Timer input) … Input
These pins are the timer G 1 external capture trigger input pins.

(e) INPT10, INTP15 (Interrupt request from peripherals) … Input
These are external interrupt request input pins.

(5) P50 to P56 (Port 5) … Input/output

Port 5 is a 7-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P50 to P56 operate as the real-time pulse unit (RPU) input/output, as the serial interface (FCAN4[NOTE]) and as external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 5 mode control register (PMC5).

(a) Port mode

P50 to P56 can be set to input or output in 1-bit units using the port 5 mode register (PM5).

(b) Control mode

P50 to P56 can be set to port or control mode in 1-bit units using PMC5.

(c) TO0 (Timer output) … Output

This pin output a timer C pulse signal.

(d) TI0, TI1 (Timer input) … Input

These pins are the timer C external capture trigger input pins.

(e) CTXD4 (Transmit data for controller area network) … Output

This pin outputs FCAN serial transmit data.

(f) CRXD4 (Receive data for controller area network) … Input

This pin inputs FCAN serial receive data.

(g) INPT20, INTP21, INTP4, INTP5 (Interrupt request from peripherals) … Input

These are external interrupt request input pins.

Note: CAN module 4 is available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

(6) P60 to P67 (Port 6) … Input/Output

Port 6 is an 8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, in control mode, P60 to P67 operate as the serial interface (CSI2) or as an external interrupt request input.
An operation mode of port or control mode can be selected for each bit and specified by the port 6 mode control register (PMC6).

(a) Port mode
P60 to P67 can be set to input or output in 1-bit units using the port 6 mode register (PM6).

(b) Control mode
P60 to P67 can be set to port or control mode in 1-bit units using PMC6.

(c) SO2 (Serial output) … Output
This pin output CSI2 serial transmit data.

(d) SI2 (Serial input) … Input
This pin input CSI2 serial receive data.

(e) $\overline{\text{SCK2}}$ (Serial clock) … Input/output
This pin is the CSI2 serial clock input/output pin.

(f) INTP0 - INTP3 (Interrupt request from peripherals) … Input
These are external interrupt request input pins.

(g) NMI (Non-maskable interrupt request)... Input
This pin is the non-maskable external interrupt request input pin.

(7) P70 to P77 (Port 7), P80 to P83 (Port 8) … Input

Port 7 is an 8-bit input-only port in which all pins are fixed as input pins. Port 8 is a 4-bit input-only port. P70 to P77 and P80 to P83 can function as input ports and as analog input pins for the A/D converter in control mode. However, they cannot be switched between these input port and analog input pin.

(a) Port mode

P70 to P77 and P80 to P83 are input-only pins.

(b) Control mode

P70 to P77 also function as pins ANI0 to ANI7 and P80 to P83 also function as ANI8 to ANI11, but these alternate functions are not switchable.

(c) ANI0 to ANI11 (Analog Input 0 to 11)

These are the analog input pins for the A/D converter. Connect a capacitor between AV_{DD} and AV_{SS} to prevent noise-related operation faults. Also, do not apply voltage that is outside the range for AV_{DD} and AV_{SS} to pins that are being used as inputs for the A/D converter. If it is possible for noise above the AV_{DD} range or below the AV_{SS} to enter, clamp these pins using a diode that has a small V_F value.

(8) P90 to P97 (Port 9) … Input/Output

Port 9 is an 8-bit input/output port in which input or output can be set in 1-bit units.
An operation mode control register is not available for port 9, since no port pin of port 9 is shared with peripheral input/output ports.

(a) Port mode

P90 to P97 can be set to input or output in 1-bit units using the port 9 mode register (PM9).

(9) PAH0 to PAH7 (Port AH) … Input/output

Port AH is an 8-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode, this port operates as the address bus (A16 to A23) for when memory is accessed externally.
An operation mode of port or control mode can be selected for each bit and specified by the port AH mode control register (PMCAH).

(a) Port mode
PAH0 to PAH7 can be set to input or output in 1-bit units using the port AH mode register (PMAH).

(b) Control mode
PAH0 to PAH7 can be used as A16 to A23 by using PMCAH.

(c) A16 to A23 (Address) … Output
This pin outputs the upper 8-bit address of the 24-bit address in the address bus on an external access.

(10) PCS0, PCS3, PCS4 (Port CS) … Input/output

Port CS is a 3-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode, it operates as a chip-select control signal output when memory is accessed externally.
An operation mode of port or control mode can be selected for each bit and specified by the port CS mode control register (PMCCS).

(a) Port mode
PCS0, PCS3, PCS4 can be set to input or output in 1-bit units using the port CS mode register (PMCS).

(b) Control mode
PCS0, PCS3, PCS4 can be used as CS0, CS3, CS4 by using PMCCS.

(c) $\overline{\text{CS0}}$, $\overline{\text{CS3}}$, $\overline{\text{CS4}}$ (Chip select) … Output
This is the chip select signal for external SRAM, external ROM, or external peripheral I/O.
The signal CSn is assigned to memory block n (n = 0, 3, 4).
This is active for the period during which a bus cycle that accesses the corresponding memory block is activated.
It is inactive in an idle state (TI).

(11) PCT0, PCT1, PCT4 (Port CT) … Input/output

Port CT is a 3-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode, it operates as control signal output when memory is accessed externally.
An operation mode of port or control mode can be selected for each bit and specified by the port CT code control register (PMCCT).

(a) Port mode

PCT0, PCT1, PCT4 can be set to input or output in 1-bit units using the port CT mode register (PMCT).

(b) Control mode

PCT0, PCT1, PCT4 can be used as $\overline{\text{LWR}}$, $\overline{\text{UWR}}$, $\overline{\text{RD}}$ by using PMCCT.

(c) $\overline{\textbf{LWR}}$ (Lower byte write strobe) … Output

This is a strobe signal that shows that the executing bus cycle is a write cycle for SRAM, external ROM, or an external peripheral I/O area.

(d) $\overline{\textbf{UWR}}$ (Upper byte write strobe) … Output

This is a strobe signal that shows that the executing bus cycle is a write cycle for SRAM, external ROM, or an external peripheral I/O area.

(e) $\overline{\textbf{RD}}$ (Read strobe) … Output

This is a strobe signal that shows that the executing bus cycle is a read cycle for SRAM, external ROM, or external peripheral I/O. It is inactive in an idle state (TI).

(12) PCM0 (Port CM) … Input/output

Port CM is a 1-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as a port, in control mode, it operates as control signal output when memory is accessed externally.
An operation mode of port or control mode can be selected for each bit and specified by the port CM code control register (PMCCM).

(a) Port mode

PCM0 can be set to input or output in 1-bit units using the port CM mode register (PMCM).

(b) Control mode

PCM0 can be used as $\overline{\text{WAIT}}$ by using PMCCM.

(c) $\overline{\textbf{WAIT}}$ (Wait) … Input

This control signal input pin, which inserts a data wait in a bus cycle. If the setup or hold time is not secured in the sampling timing, wait insertion may not be performed.

(13) ANI00 to ANI11 (Analog input) … Input

These are analog input pins to the A/D converter.

(14) MODE0 to MODE2 (Mode) … Input

These are the input pins that specify the operation mode. Operation modes are broadly divided into normal operation modes and flash memory programming mode. The operation mode is determined by sampling the status of each of pins MODE0 to MODE2 on a reset.
Fix these so that the input level does not change during operation.

(15) $\overline{\text{RESET}}$ (Reset) … Input

$\overline{\text{RESET}}$ input is asynchronous input. When a signal having a certain low level width is input in asynchronous with the operation clock, a system reset that takes precedence over all operations occurs.
Besides a normal initialize or start, this signal is also used to release a standby mode (HALT, IDLE, Watch, Sub-Watch software STOP).

(16) $\overline{\text{RESOUT}}$ (Reset) … Output

$\overline{\text{RESOUT}}$ output is a 3.3 V reset output signal. It is the internal system reset output. $\overline{\text{RESOUT}}$ is active (low) in case of an external reset by $\overline{\text{RESET}}$ input pin or internal reset by watch-dog timer. If the $\overline{\text{RESOUT}}$ output pin is connected to a RESET-IN of an external flash memory it can be used to terminate an embedded erase/programming operation at the occurrence of a valid $\overline{\text{RESET}}$ signal.

(17) NMI (NON-Maskable Interrupt Request)... input

This is the non-maskable interrupt request input pin.

(18) X1, X2 (Crystal)

These pins connect a resonator for system main-clock generation.

(19) XT1, XT2 (Crystal)

These pins connect a resonator for the sub-clock generation.

(20) CV_{DD} (Power supply for clock generator)

This is the positive power supply pin for the clock generator.

(21) CV_{SS} (Ground for clock generator)

This is the ground pin for the clock generator.

(22) V_{DD50} to V_{DD52} (Power supply)

These are the positive 5 V power supply pins.

(23) V_{SS50} to V_{SS52} (Ground)

These are the ground pins for the 5 V power supply.

(24) V_{DD30} to V_{DD36} (Power supply)

These are the positive 3.3 V power supply pins.

(25) V_{SS30} to V_{SS36} (Ground)

These are the ground pins for the 3.3 V power supply.

(26) AV_{DD} (Analog power supply)

This is the analog positive power supply pin for the A/D converter.

(27) AV_{SS} (Analog ground)

This is the ground pin for the A/D converter.

(28) AV_{REF} (Analog reference voltage) … Input

This is the reference voltage supply pin for the A/D converter.

(29) A0 to A15 (Address output) … Output

These pins are the address output pins.

(30) D0 to D15 (Data input/output) … Input/output

These pins are the data input/output pins.

2.3 Types of Pin I/O Circuit and Connection of Unused Pins

Table 2-4: Types of Pin I/O Circuit and Connection of Unused Pins (1/3)

Pin			I/O Circuit Type	Recommended connection
P10	FCRXD1		5-K	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
P11	FCTXD1			
P12	FCRXD2			
P13	FCTXD2			
P14	FCRXD3			
P15	FCTXD3			
P16	RXD51			
P17	TXD51			
P20	SI00		5-K	
P21	SO00			
P22	$\overline{\text{SCK00}}$			
P23	SI01			
P24	SO01			
P25	$\overline{\text{SCK01}}$			
P26	RXD50			
P27	TXD50			
P30	TIG00	INTP00	5-K	
P31	TIG01	TOG01		
P32	TIG02	TOG02		
P33	TIG03	TOG03		
P34	TIG04	TOG04		
P35	TIG05	INTP05		
P40	TIG10	INTP10	5-K	
P41	TIG11	TOG11		
P42	TIG12	TOG12		
P43	TIG13	TOG13		
P44	TIG14	TOG14		
P45	TIG15	INTP15		
P50	FCRXD4		5-K	
P51	FCTXD4			
P52	INTP4			
P53	INTP5			
P54	TIC00	INTP20		
P55	TIC01	INTP21		
P56	TOC0			

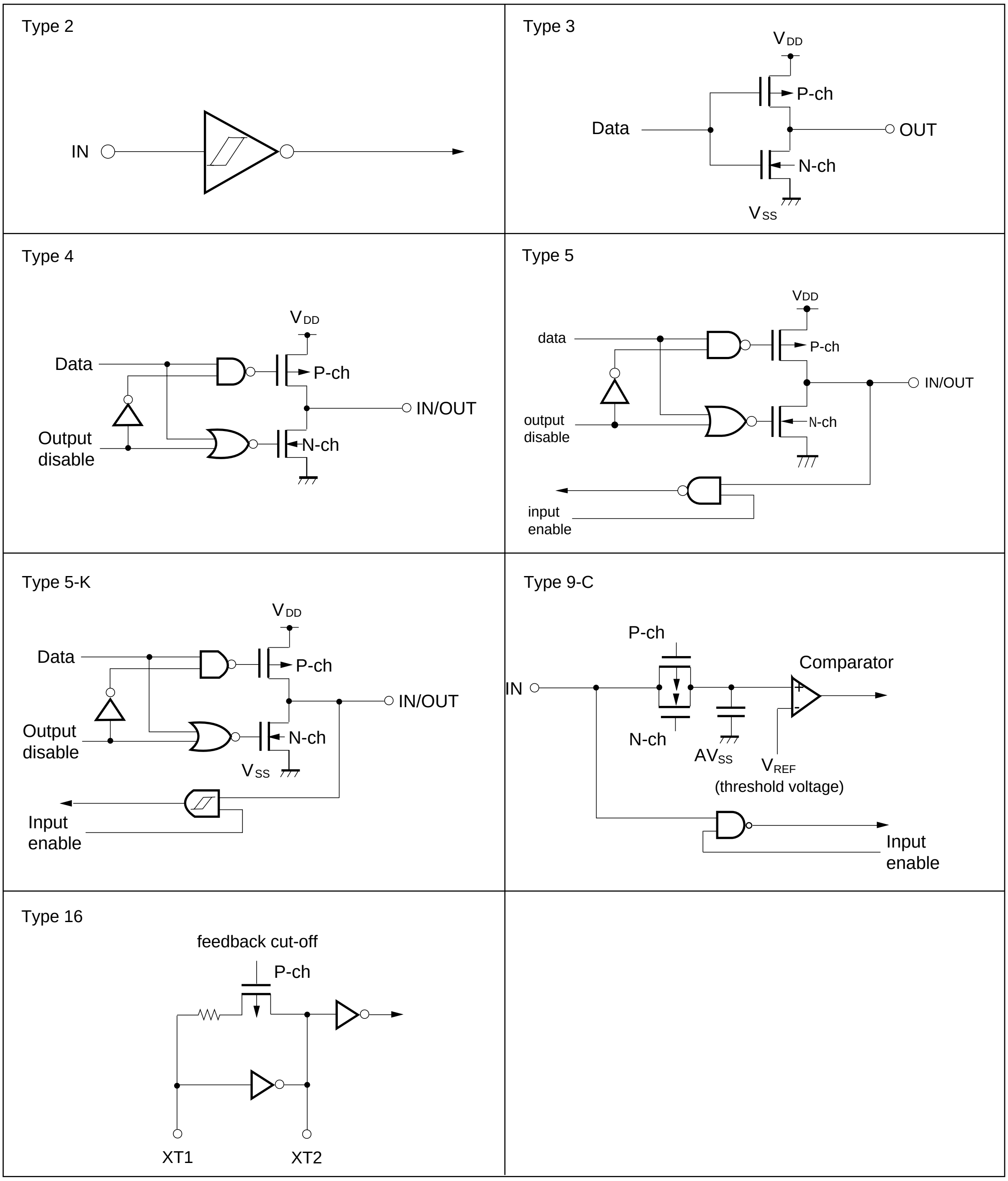
Table 2-4: Types of Pin I/O Circuit and Connection of Unused Pins (2/3)

Pin			I/O Circuit Type	Recommended connection
P60	NMI		5-K	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
P61	INTP0			
P62	INTP1			
P63	INTP2			
P64	INTP3			
P65	SI02			
P66	SO02			
P67	$\overline{SCK02}$			
P70	ANI0		9-C	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
P71	ANI1			
P72	ANI2			
P73	ANI3			
P74	ANI4			
P75	ANI5			
P76	ANI6			
P77	ANI7			
P80	ANI8		9-C	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
P81	ANI9			
P82	ANI10			
P83	ANI11			
P90			5-K	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
P91				
P92				
P93				
P94				
P95				
P96				
P97				
PAH0	A16		5	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
PAH1	A17			
PAH2	A18			
PAH3	A19			
PAH4	A20			
PAH5	A21			
PAH6	A22			
PAH7	A23			
PCS0	$\overline{CS0}$		5	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
PCS3	$\overline{CS3}$			
PCS4	$\overline{CS4}$			

Table 2-4: Types of Pin I/O Circuit and Connection of Unused Pins (3/3)

Pin			I/O Circuit Type	Recommended connection
PCT0	$\overline{\text{LWR}}$		5	For input: individually connect to V_{DD5} or V_{SS5} via a resistor. For output: leave open.
PCT1	$\overline{\text{UWR}}$			
PCT4	$\overline{\text{RD}}$			
PCM0	$\overline{\text{WAIT}}$		5	
AIN0-AIN11			9-C	
MODE0			2	
MODE1			2	connect to V_{SS5x} via a resistor.
MODE2			2	V_{DD3x}
$\overline{\text{RESET}}$			2	-
$\overline{\text{RESOUT}}$			3	connect to V_{DD3x} via a resistor.
X2			16	Please refer to the datasheet
XT1			16	Please refer to the data sheet
XT2			16	Please refer to the data sheet
AV_{DD}			-	V_{DD5x}
AV_{REF}			-	AV_{DD}
AV_{SS}			-	V_{SS5x}
A0 to A15			4	
D0 to D15			5	

Figure 2-1: Pin I/O Circuits



Chapter 3 CPU Function

The CPU of the V850E/CA2 Jupiter is based on a RISC architecture and executes almost all the instructions which can be accessed from the iCache in one clock cycle, using a 5-stage pipeline control.

3.1 Features

- Minimum instruction cycle: 31.25 ns (@ internal 32 MHz operation)
- Memory space
  - Program space: 64 MB linear
  - Data space: 4 GB linear
- Thirty-two 32-bit general registers
- Internal 32-bit architecture
- Five-stage pipeline control
- Multiplication/division instructions
- Saturated operation instructions
- One-clock 32-bit shift instruction (barrel shifter)
- Long/short instruction format
- Four types of bit manipulation instructions
  - Set
  - Clear
  - Not
  - Test

3.2 CPU Register Set

The registers of the V850E/CA2 Jupiter can be classified into two categories: a general program register set and a dedicated system register set. All the registers are 32-bit width. For details, refer to V850E User's Manual Architecture.

Figure 3-1: CPU Register Set

(1) Program register set

31 … 0

r0	(Zero Register)
r1	(Reserved for Assembler)
r2	(Interrupt Stack Pointer)
r3	(Stack Pointer (SP))
r4	(Global Pointer (GP))
r5	(Text Pointer (TP))
r6	
r7	
r8	
r9	
r10	
r11	
r12	
r13	
r14	
r15	
r16	
r17	
r18	
r19	
r20	
r21	
r22	
r23	
r24	
r25	
r26	
r27	
r28	
r29	
r30	(Element Pointer (EP))
r31	(Link Pointer (LP))

31 … 0

PC	(Program Counter)

(2) System register set

31 … 0

EIPC	(Status Saving Register during interrupt)
EIPSW	(Status Saving Register during interrupt)

FEPC	(Status Saving Register during NMI)
FEPSW	(Status Saving Register during NMI)

ECR	(Interrrupt Source Register)

PSW	(Program Status Word)

CTPC	(Status Saving Register during CALLT execution)
CTPSW	(Status Saving Register during CALLT execution)

DBPC	(Status Saving Register during exception/debug trap)
DBPSW	(Status Saving Register during exception/debug trap)

CTBP	(CALLT Base Pointer)

3.2.1 Program register set

The program register set includes general registers and a program counter.

(1) General registers

Thirty-two general registers, r0 to r31, are available. Any of these registers can be used as a data variable or address variable.
However, r0 and r30 are implicitly used by instructions, and care must be exercised when using these registers. r0 is a register that always holds 0, and is used for operations using 0 and offset 0 addressing. r30 is used, by means of the SLD and SST instructions, as a base pointer for when memory is accessed. Also, r1, r3 to r5, and r31 are implicitly used by the assembler and C compiler. Therefore, before using these registers, their contents must be saved so that they are not lost. The contents must be restored to the registers after the registers have been used.

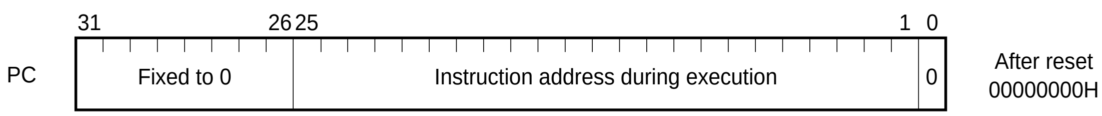
Table 3-1: Program Registers

Name	Usage	Operation
r0	Zero register	Always holds 0
r1	Assembler-reserved register	Working register for generating 32-bit immediate data
r2	Address/data variable registers	
r3	Stack pointer	Used to generate stack frame when function is called
r4	Global pointer	Used to access global variable in data area
r5	Text pointer	Register to indicate the start of the text area (where program code is located)
r6 to r29	Address/data variable registers	
r30	Element pointer	Base pointer when memory is accessed
r31	Link pointer	Used by compiler when calling function
PC	Program counter	Holds instruction address during program execution

(2) Program counter

This register holds the instruction address during program execution. The lower 26 bits of this register are valid, and bits 31 to 26 are fixed to 0. If a carry occurs from bit 25 to 26, it is ignored. Bit 0 is fixed to 0, and branching to an odd address cannot be performed.

Figure 3-2: Program Counter (PC)



3.2.2 System register set

System registers control the status of the CPU and hold interrupt information.
To read/write these system registers, use the system register load/store instruction (LDSR or STSR instruction) with a specific system register number indicated below.

Table 3-2: System Register Numbers

No.	System Register Name	Operand Specification	
		LDSR Instruction	STSR Instruction
0	Status saving register during interrupt (EIPC)Note 1	O	O
1	Status saving register during interrupt (EIPSW)	O	O
2	Status saving register during NMI (FEPC)	O	O
3	Status saving register during NMI (FEPSW)	O	O
4	Interrupt source register (ECR)	×	O
5	Program status word (PSW)	O	O
6 to 15	Reserved number for future function expansion (operations that access these register numbers cannot be guaranteed).	×	×
16	Status saving register during CALLT execution (CTPC)	O	O
17	Status saving register during CALLT execution (CTPSW)	O	O
18	Status saving register during exception/debug trap (DBPC)	ONote 2	O
19	Status saving register during exception/debug trap (DBPSW)	ONote 2	O
20	CALLT base pointer (CTBP)	O	O
21 to 31	Reserved number for future function expansion (operations that access these register numbers cannot be guaranteed).	×	×

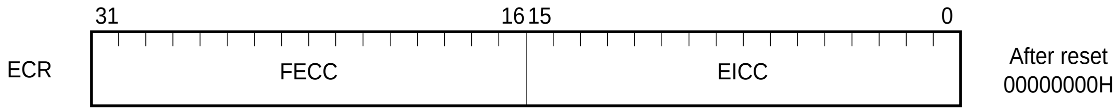
Notes: 1. Because this register has only one set, to approve multiple interrupts, it is necessary to save this register by program.

2. Access is only possible while the DBTRAP instruction is executed.

Caution: Even if bit 0 of EIPC, FEPC, or CTPC is set to 1 with the LDSR instruction, bit 0 will be ignored when the program returned by RETI instruction after interrupt servicing (because bit 0 of the PC is fixed to 0). When setting the value of EIPC, FEPC, or CTPC, use the even value (bit 0 = 0).

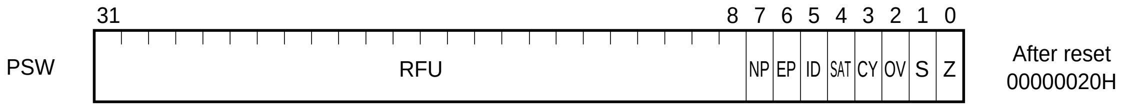
Remark: O: Access allowed
×: Access prohibited

Figure 3-3: Interrupt Source Register (ECR)



Bit Position	Bit Name	Function
31 to 16	FECC	Exception code of non-maskable interrupt (NMI)
15 to 0	EICC	Exception code of exception/maskable interrupt

Figure 3-4: Program Status Word (PSW)



Bit Position	Flag	Function
31 to 8	RFU	Reserved field (fixed to 0).
7	NP	Indicates that non-maskable interrupt (NMI) processing is in progress. This flag is set when NMI is accepted, and disables multiple interrupts. 0: NMI servicing not under execution. 1: NMI servicing under execution.
6	EP	Indicates that exception processing is in progress. This flag is set when an exception is generated. Moreover, interrupt requests can be accepted when this bit is set. 0: Exception processing not under execution. 1: Exception processing under execution.
5	ID	Displays whether a maskable interrupt request has been acknowledged or not. 0: Interrupt enabled. 1: Interrupt disabled.
4	SAT[Note]	Displays that the operation result of a saturated operation processing instruction is saturated due to overflow. Due to the cumulative flag, if the operation result is saturated by the saturation operation instruction, this bit is set (1), but is not cleared (0) even if the operation results of subsequent instructions are not saturated. To clear (0) this bit, load the data in PSW. Note that in a general arithmetic operation, this bit is neither set (1) nor cleared (0). 0: Not saturated. 1: Saturated.
3	CY	This flag is set if carry or borrow occurs as result of operation (if carry or borrow does not occur, it is reset). 0: Carry or borrow does not occur. 1: Carry or borrow occurs.
2	OV[Note]	This flag is set if overflow occurs during operation (if overflow does not occur, it is reset). 0: Overflow does not occur. 1: Overflow occurs.
1	S[Note]	This flag is set if the result of operation is negative (it is reset if the result is positive). 0: The operation result was positive or 0. 1: The operation result was negative.
0	Z	This flag is set if the result of operation is zero (if the result is not zero, it is reset). 0: The operation result was not 0. 1: The operation result was 0.

Note: The result of a saturation-processed operation is determined by the contents of the OV and S flags in the saturation operation. Simply setting the OV flag (1) will set the SAT flag (1) in a saturation operation.

Table 3-3: Saturation-Processed Operation Result

Status of Operation Result	Flag Status			Saturation-Processed Operation Result
	SAT	OV	S	
Maximum positive value exceeded	1	1	0	7FFFFFFFH
Maximum negative value exceeded	1	1	1	80000000H
Positive (maximum not exceeded)	Retains the value before operation	0	0	Operation result itself
Negative (maximum not exceeded)			1	

3.3 Operation Modes

3.3.1 Operation modes

The V850E/CA2 Jupiter has the following operations modes. Mode specification is carried out by the MODE0 to MODE2 pins.

(1) Normal operation mode

ROM-less mode
Access to the external ROM is enabled.
In ROM-less mode, after system reset is cleared, each pin related to the bus interface enters the control mode, program execution branches to the reset entry address of the external ROM and instruction processing starts.

(2) Flash memory programming mode

If this mode is specified, it becomes possible to modify (erase, program) the contents of external flash-memories which are connected to the V850E/CA2 Jupiter device via its external memory inter-face.
The V850E/CA2 Jupiter device provides a built in Boot-Loader offering the possibility to download programming algorithms and the new ROM-code itself. To be able to use this feature there's no need to have already a dedicated boot software being programmed in the external flash-memory. It is possible to program external flash-memory devices even in the case that the flash-memory is completely erased (Support of "Virgin-programming"). The complete Boot-Loader functionality is provided from the V850E/ CA2 Jupiter device itself. This Boot-Loader is enabled when the so-called "Flash-Programming Mode" is enabled by a dedicated configuration of the V850E/CA2 Jupiter device's MODE0 to MODE2 pins.
As an programming interface, the UART0 or the CSI0 serial interface can be selected. The selection of the interface intended to be used can be done by applying a fixed voltage level (UART0) or a dedicated amount of pulses (CSI0) to the V850E/CA2 Jupiter device's MODE 1 pin.
For further information please refer to the document "Preliminary Application Note EASE-AN-4050".

3.3.2 Operation mode specification

The operation mode is specified according to the status of pins MODE0 to MODE2. In an application system fix the specification of these pins and do not change them during operation. Operation is not guaranteed if these pins are changed during operation.
To program/erase the contents of the external memory device, it is required to enable a “Flash-Programming-Mode”. The following operation modes are generally available for the V850E/CA2 Jupiter device:

(a) µPD703128, µPD703129

Table 3-4: Operation Modes

MODE2	MODE1	MODE0	Operation Mode
L	L	L	ROM-less mode 0 (Direct): Normal operation mode
L	L	H	ROM-less mode 1(Low EMI): Normal operation mode
L	H	L	Flash memory programming mode 0 (Direct)
L	H	H	Flash memory programming mode 1 (Low EMI)
Other than above			Setting prohibited

Remarks: 1. L: Low-level input

2. H: High-level input

(1) ROM-less Mode 0

When a system reset is released, the bus interface pins enter the peripheral mode and the program branches to the reset entry address in the external memory to start instruction execution. Address output is masked by additional circuitry in the ROMLESS Mode 0 to reduce EMI. Address is output only in case external access is performed.
Data and instructions in the internal boot ROM cannot be accessed or fetched.

(2) ROM-less Mode 1

When a system reset is released, the bus interface pins enter the peripheral mode and the program branches to the reset entry address in the external memory to start instruction execution. Address output is not masked in the ROMLESS Mode 1. Address is always output. This is the ordinary operation of external memory interface.
Data and instructions in the internal boot ROM cannot be accessed or fetched.

(3) FLASH Programming Mode 0

In FLASH Programming Mode 0 external flash memory programming is enabled by starting from the internal boot ROM of Jupiter. This boot ROM contains bootstrap code to download FLASH programming routines into iRAM and execute these routines. Address masking on the external memory interface is active.

Note: The lower 1MB memory area is occupied by the boot ROM in FLASH Programming Mode 0. Please refer to Chapter 3.5 “Memory Map” on page 68 for a memory map.

(4) FLASH Programming Mode 1

In FLASH Programming Mode 1 external flash memory programming is enabled by starting from the internal boot ROM of Jupiter. This boot ROM contains bootstrap code to download FLASH programming routines into iRAM and execute these routines. Address on the external memory interface is not masked.

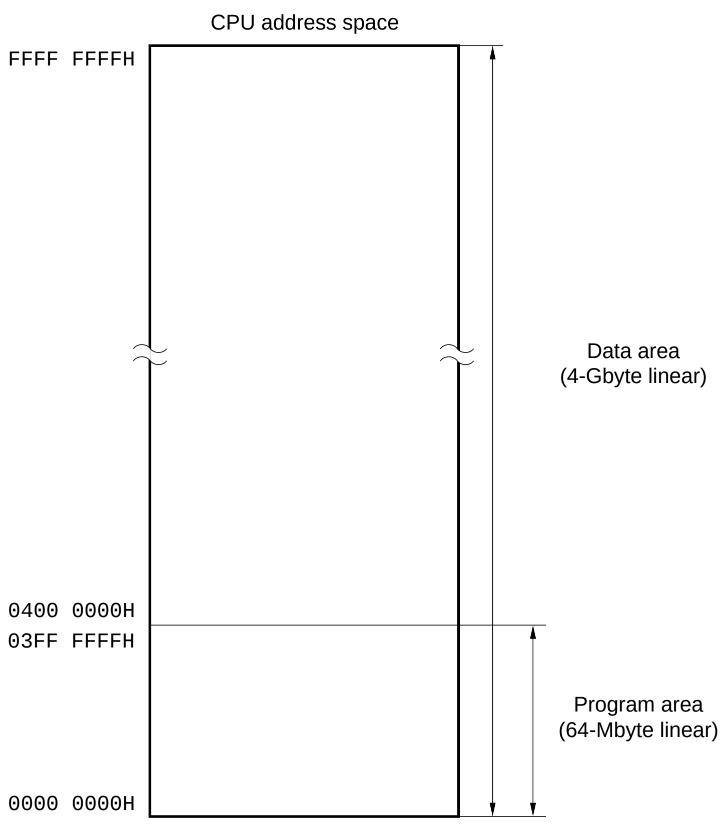
Note: The lower 1MB memory area is occupied by the boot ROM in FLASH Programming Mode 1. Please refer to Chapter 3.5 “Memory Map” on page 68 for a memory map.

3.4 Address Space

3.4.1 CPU address space

The CPU of the V850E/CA2 is of 32-bit architecture and supports up to 4 GB of linear address space (data space) during operand addressing (data access).
Also, in instruction address addressing, a maximum of 64 MB of linear address space (program space) is supported.
Figure 3-5 shows the CPU address space.

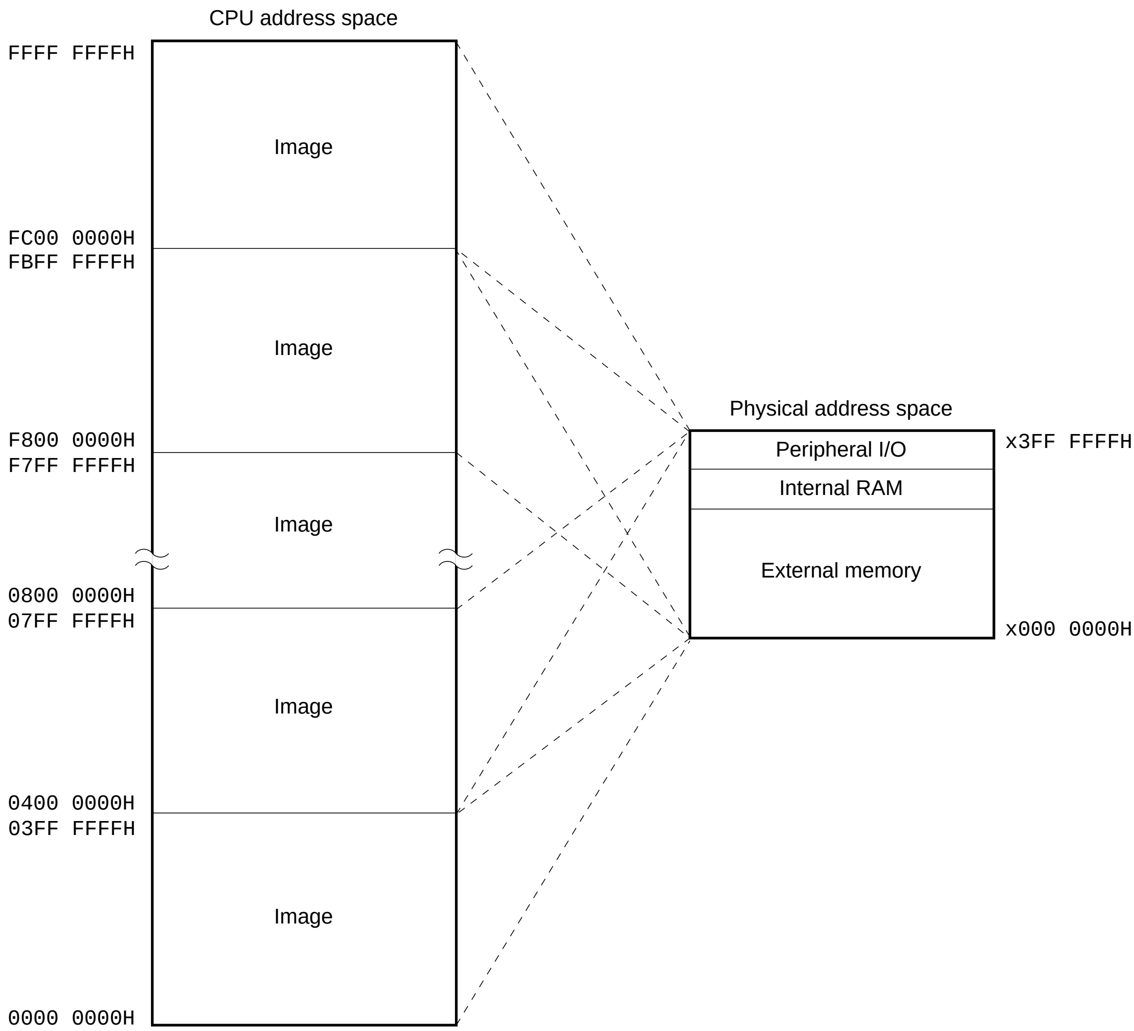
Figure 3-5: CPU Address Space



3.4.2 Image

64 MB physical address space is seen as 64 images in the 4 GB CPU address space. In actuality, the same 64 MB physical address space is accessed regardless of the values of bits 31 to 26 of the CPU address. Figure 3-6 shows the image of the virtual addressing space.
Physical address x000 0000H can be seen as CPU address 0000 0000H, and in addition, can be seen as address 0400 0000H, address 0800 0000H, …, address F800 0000H, or address FC00 0000H.

Figure 3-6: Image on Address Space



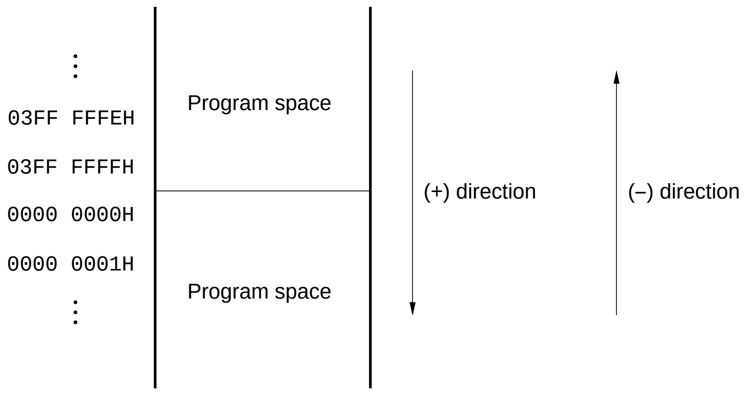
3.4.3 Wrap-around of CPU address space

(1) Program space

Of the 32 bits of the PC (program counter), the higher 6 bits are set to “0”, and only the lower 26 bits are valid. Even if a carry or borrow occurs from bit 25 to 26 as a result of branch address calculation, the higher 6 bits ignore the carry or borrow.

Therefore, the lower-limit address of the program space, address 0000 0000H, and the upper-limit address 03FF FFFFH become contiguous addresses. Wrap-around refers to the situation that the lower-limit address and upper-limit address become contiguous like this.

Figure 3-7: Wrap-around of Program Space



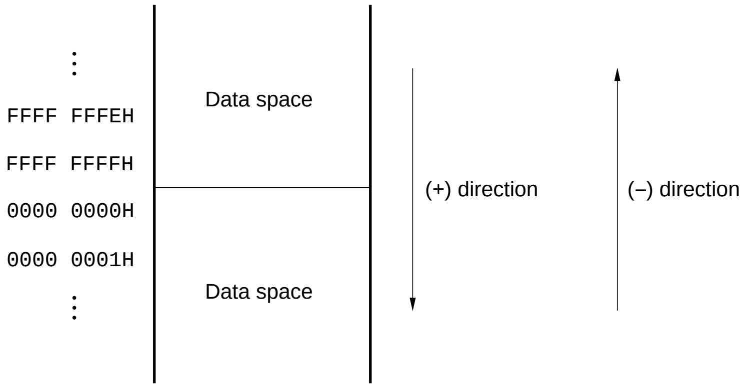
Caution: No instruction can be fetched from the 4 KB area of 03FF F000H to 03FF FFFFH because this area is defined as peripheral I/O area. Therefore, do not execute any branch address calculation in which the result will reside in any part of this area.

(2) Data space

The result of operand address calculation that exceeds 32 bits is ignored.

Therefore, the lower-limit address of the program space, address 0000 0000H, and the upper-limit address FFFF FFFFH are contiguous addresses, and the data space is wrapped around at the boundary of these addresses.

Figure 3-8: Wrap-around of Data Space

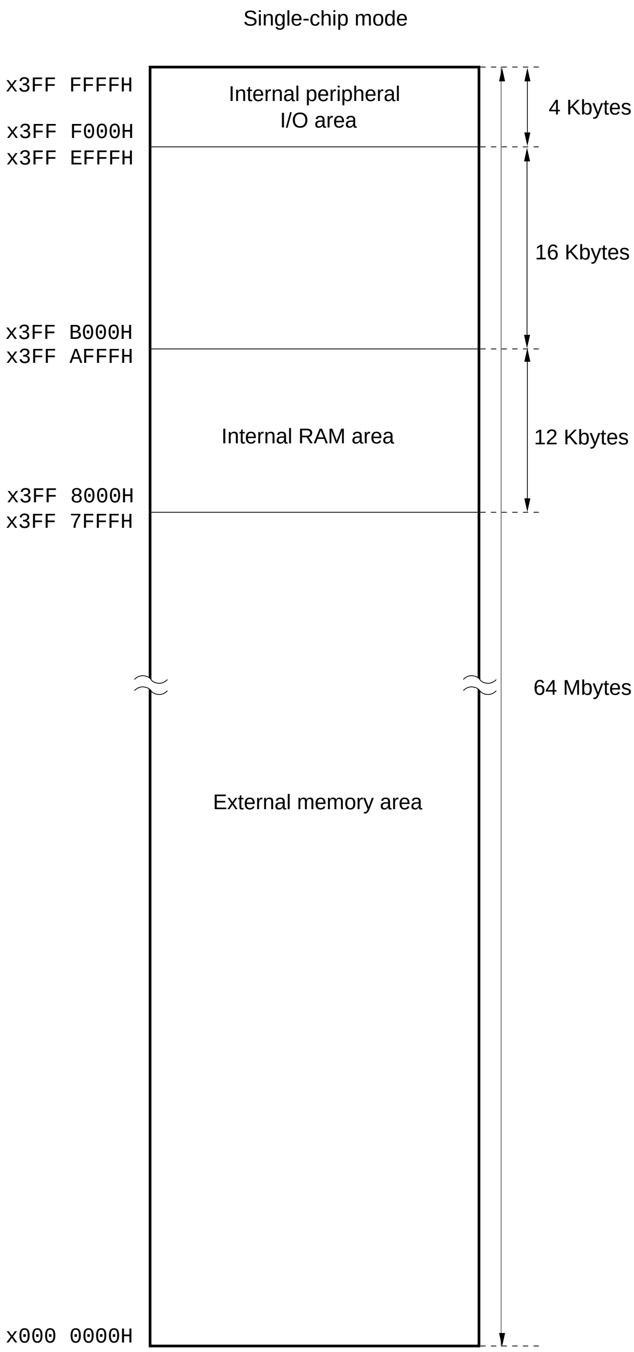


3.5 Memory Map

The V850E/CA2 reserves areas as shown in Figure 3-9.

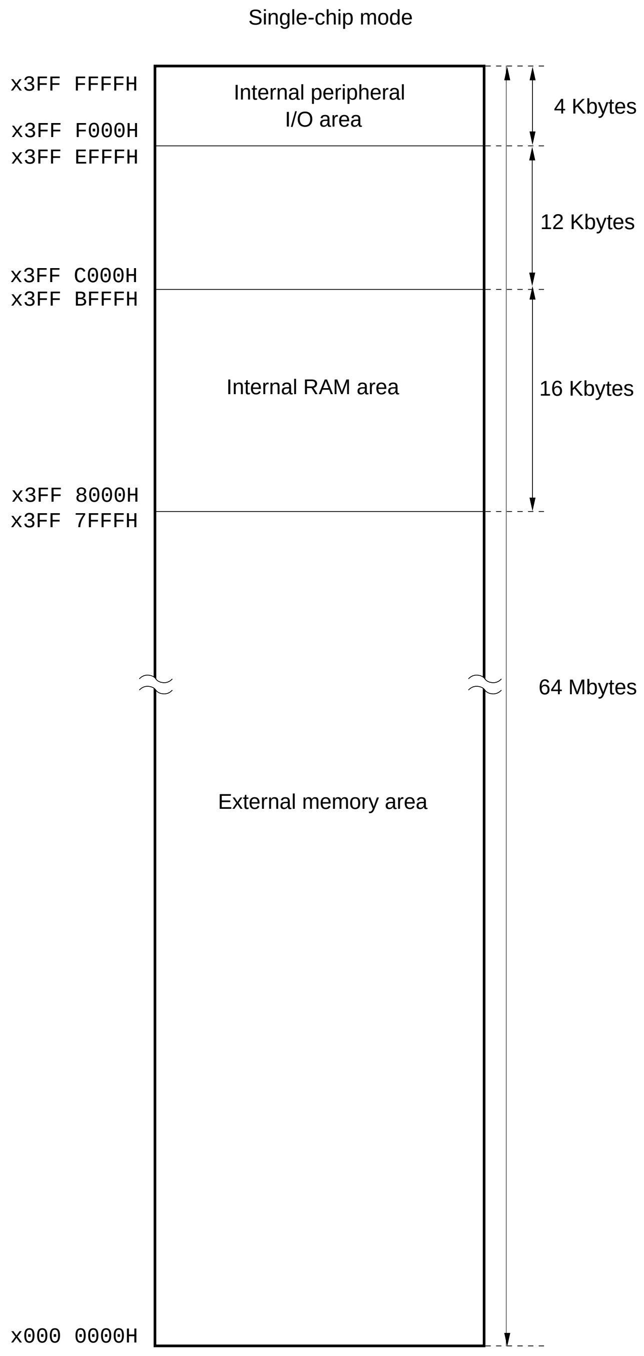
(1) For µPD703128

Figure 3-9: Memory Map (µPD703128 (A))



(2) For µPD703129

Figure 3-10: Memory Map (µPD703129 (A), µPD703129 (A1))



3.5.1 Area

(1) External ROM area

The following areas can be used as external memory area.

(a) μPD703128, μPD703129

x000 0000H to x3FF 7FFFH

Access to the external memory area uses the chip select signal assigned to each memory block (which is carried out in the CS unit set by chip area selection control registers 0 and 1 (CSC0, CSC1)).
Furthermore, the internal ROM, internal RAM, and internal peripheral I/O areas cannot be accessed as external memory areas.

(b) Interrupt/exception table

The V850E/CA2 increases the interrupt response speed by assigning handler addresses corresponding to interrupts/exceptions.
The collection of these handler addresses is called an interrupt/exception table, which is located in the external ROM area. When an interrupt/exception request is accepted, execution jumps to the handler address, and the program written at that memory is executed.
Table 3-5 shows the sources of interrupts/exceptions, and the corresponding addresses.

Table 3-5: Interrupt/Exception Table (1/3)

Start Address of Interrupt/ Exception Table	Interrupt/Exception Source
0000 0000H	$\overline{\text{RESET}}$ input
0000 0010H	P60 NMI Input
0000 0020H	Watchdog timer
0000 0040H	TRAP instruction
0000 0050H	TRAP instruction
0000 0060H	Illegal opcode/DBTRAP instruction
0000 0080H	Real Time Clock Divider Tick
0000 0090H	Compare Match
0000 00A0H	Compare Match
0000 00B0H	Interval time
0000 00C0H	P61
0000 00D0H	P62
0000 00E0H	P63
0000 00F0H	P64
0000 0100H	P52
0000 0110H	P53
0000 0120H	Time base 0 Overflow
0000 0130H	Time base 1 Overflow

Note: Reserved for internal use only please leave at $\overline{\text{RESET}}$ value.

Table 3-5: Interrupt/Exception Table (2/3)

Start Address of Interrupt/ Exception Table	Interrupt/Exception Source
0000 0140H	CC coincidence Channel 0
0000 0150H	CC coincidence Channel 1
0000 0160H	CC coincidence Channel 2
0000 0170H	CC coincidence Channel 3
0000 0180H	CC coincidence Channel 4
0000 0190H	CC coincidence Channel 5
0000 01A0H	Time base 0 Overflow
0000 01B0H	Time base 1 Overflow
0000 01C0H	CC coincidence Channel 0
0000 01D0H	CC coincidence Channel 1
0000 01E0H	CC coincidence Channel 2
0000 01F0H	CC coincidence Channel 3
0000 0200H	CC coincidence Channel 4
0000 0210H	CC coincidence Channel 5
0000 0220H	Time base Overflow
0000 0230H	CC coincidence Channel 0
0000 0240H	CC coincidence Channel 1
0000 0250H	A/D conversion end
0000 0260H	MAC Interrupt CGINTP 1-2
0000 0270H	CAN1 Receive Interrupt
0000 0280H	CAN1 Transmit Interrupt
0000 0290H	CAN1 Error Interrupt
0000 02A0H	CAN2 Receive Interrupt
0000 02B0H	CAN2 Transmit Interrupt
0000 02C0H	CAN2 Error Interrupt
0000 02D0H	CAN3 Receive Interrupt
0000 02E0H	CAN3 Transmit Interrupt
0000 02F0H	CAN3 Error Interrupt
0000 0300H	CAN4 Receive Interrupt
0000 0310H	CAN4 Transmit Interrupt
0000 0320H	CAN4 Error Interrupt
0000 0330H	Transmission/ Reception Completion CSI0
0000 0340H	Transmission/ Reception Completion CSI1
0000 0350H	Transmission/ Reception Completion CSI2
0000 0360H	Reception Error UART50
0000 0370H	Reception Completion UART50
0000 0380H	Transmission Completion UART50

Note: Reserved for internal use only please leave at $\overline{\text{RESET}}$ value.

Table 3-5: Interrupt/Exception Table (3/3)

Start Address of Interrupt/ Exception Table	Interrupt/Exception Source
0000 0390H	Reception Error UART51
0000 03A0H	Reception Completion UART51
0000 03B0H	Transmission Completion UART51
0000 03C0H	DMA Channel 0 transfer completed
0000 03D0H	DMA Channel 1 transfer completed
0000 03E0H	DMA Channel 2 transfer completed
0000 03F0H	DMA Channel 3 transfer completed
0000 0400H	DMA Overflow
0000 0410H	P30
0000 0420H	P35
0000 0430H	P40
0000 0440H	P45
0000 0450H	P54
0000 0460H	P55
0000 0470H	reserved[Note]

Note: Reserved for internal use only please leave at $\overline{\text{RESET}}$ value.

(2) Internal RAM area

For the µPD703128 12 KB of memory, addresses 3FF 8000H to 3FF AFFFH, are reserved for the internal RAM area.
In the µPD703129 the 16 KB of addresses 3FF 8000H to 3FF BFFFH are provided as internal physical RAM.

Figure 3-11: Internal RAM Area of µPD703129

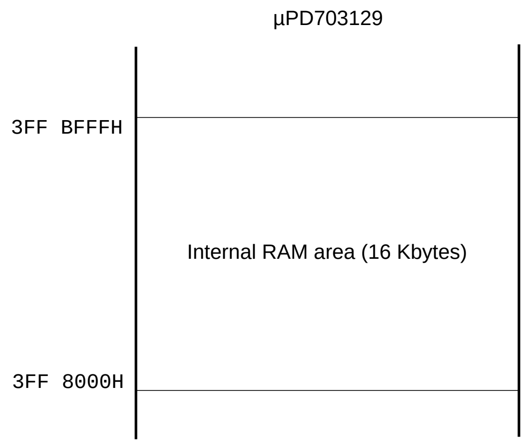
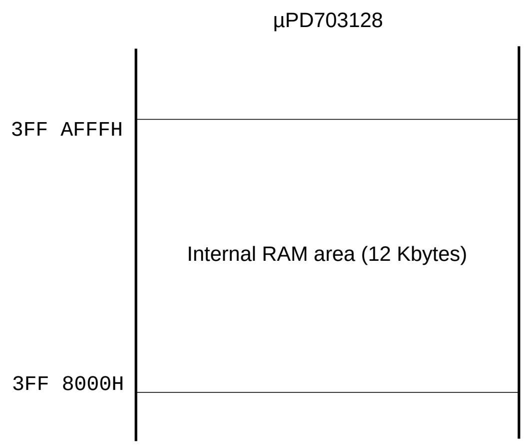


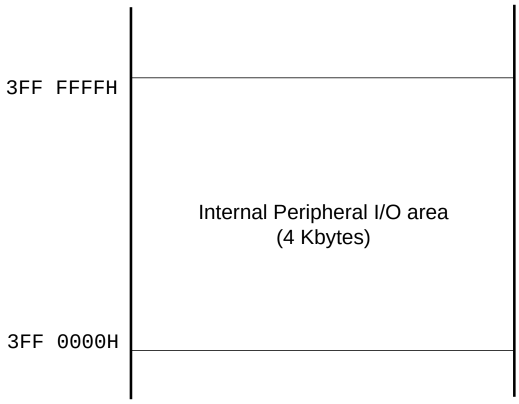
Figure 3-12: Internal RAM Area of µPD703128



(3) Internal peripheral I/O area

4 KB of memory, addresses 3FFF000H to 3FFFFFFH, is provided as an internal peripheral I/O area.

Figure 3-13: Internal Peripheral I/O Area



Peripheral I/O registers associated with the operation mode specification and the state monitoring for the internal peripherals I/O are all memory-mapped to the internal peripheral I/O area. Program fetches cannot be executed from this area.

Cautions: 1. In the V850E/CA2, no registers exist which are capable of word access. But if a register is word accessed, half word access is performed twice in the order of lower address, then higher address of the word area, ignoring the lower 2 bits of the address.

2. For registers in which byte access is possible, if half word access is executed, the higher 8 bits become undefined during the read operation, and the lower 8 bits of data are written to the register during the write operation.

3. Addresses that are not defined as registers are reserved for future expansion. If these addresses are accessed, the operation is undefined and not guaranteed.

4. Addresses 3FF F000H to 3FF FFFFH cannot be specified as the source/destination address of DMA transfer. Be sure to use addresses FFF F000H to FFF FFFFH for source/destination address of DMA transfer.

Additionally to the peripheral I/O area, a 16 KB area is provided as a programmable peripheral I/O area (refer to **3.5.4 "Programmable peripheral I/O registers" on page 83**).

3.5.2 Recommended use of address space

The architecture of the V850E/CA2 requires that a register is utilized for address generation when accessing operand data in the data space. Operand data access from instruction can be directly executed at the address in this pointer register ±32 KB. However, the use of general registers as pointer registers decreases the number of usable general registers for handling variables, but minimizes the deterioration of address calculation performance when changing the pointer value and minimizes the program size as well.
To enhance the efficiency of using the pointer in consideration of the memory map of the V850E/CA2, the following points are recommended:

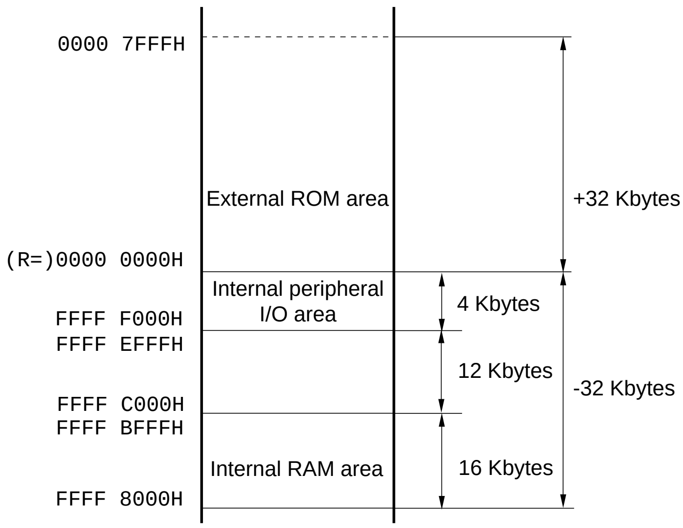
(1) Program space

Of the 32 bits of the PC (program counter), the higher 6 bits are fixed to zero (0), and only the lower 26 bits are valid. Therefore, a contiguous 64 MB space, starting from address 0000 0000H, unconditionally corresponds to the memory map of the program space.

(2) Data space

For the efficient use of resources to be performed through the wrap-around feature of the data space, the continuous 16 MB address spaces 0000 0000H to 00FF FFFFH and FF00 0000H to FFFF FFFFH of the 4 GB CPU address space are used as the data space. With the V850E/CA2, 64 MB physical address space is seen as 64 images in the 4 GB CPU address space. The highest bit (bit 25) of this 26-bit address is assigned as address sign-extended to 32 bits.

Figure 3-14: Example Application of wrap-around (µPD703129)



When R = r0 (zero register) is specified with the LD/ST disp16 [R] instruction, an addressing range of 0000 0000H ±32 KB can be referenced with the sign-extended, 16-bit displacement value.
The zero register (r0) is a register set to 0 by hardware, and eliminates the need for additional registers for the pointer.

3.5.3 Peripheral I/O Registers

Table 3-6: List of Peripheral I/O Registers (1/7)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
FFFF F002	Port AH	PAH	R/W	×	×		00H
FFFF F008	Port CS	PCS	R/W	×	×		undefined
FFFF F00A	Port CT	PCT	R/W	×	×		undefined
FFFF F00C	Port CM	PCM	R/W	×	×		undefined
FFFF F022	Port AH mode	PMAH	R/W	×	×		0000H
FFFF F028	Port CS mode	PMCS	R/W	×	×		18H
FFFF F02A	Port CT mode	PMCT	R/W	×	×		13H
FFFF F02C	Port CM mode	PMCM	R/W	×	×		01H
FFFF F042	Port AH mode control	PMCAH	R/W	×	×		FFH
FFFF F048	Port CS mode control	PMCCS	R/W	×	×		01H
FFFF F04A	Port CT mode control	PMCCT	R/W	×	×		10H
FFFF F04C	Port CM mode control	PMCCM	R/W	×	×		01H
FFFF F060	CPU: Chip Area Select Control register 0	CSC0	R/W			×	2C11H
FFFF F062	CPU: Chip Area Select Control register 1	CSC1	R/W			×	2C11H
FFFF F064	CPU: Peripheral Area Select Control register	BPC	R/W			×	0FFFH
FFFF F066	CPU: Bus Size Configuration register	BSC	R/W			×	5555H
FFFF F068	CPU: Endian Configuration register	BEC	R/W			×	0000H
FFFF F06A	CPU: Cache Configuration register	BHC	R/W			×	0000H
FFFF F06E	CPU: VPB Strobe Wait Control register	VSWC	R/W	×	×		77H
FFFF F070	Instruction Cache Control Register	ICC	R/W			×	0003H
FFFF F072	Instruction Cache Index Register	ICI	R/W			×	FFFFH
FFFF F074	Instruction Cache Data Configuration	ICD	R/W			×	undefined
FFFF F080	DMA source address register 0L	DSAL0	R/W			×	undefined
FFFF F082	DMA source address register 0H	DSAH0	R/W			×	undefined
FFFF F084	DMA destination address register 0L	DDAL0	R/W			×	undefined
FFFF F086	DMA destination address register 0H	DDAH0	R/W			×	undefined
FFFF F088	DMA source address register 1L	DSAL1	R/W			×	undefined
FFFF F08A	DMA source address register 1H	DSAH1	R/W			×	undefined
FFFF F08C	DMA destination address register 1L	DDAL1	R/W			×	undefined
FFFF F08E	DMA destination address register 1H	DDAH1	R/W			×	undefined
FFFF F090	DMA source address register 2L	DSAL2	R/W			×	undefined
FFFF F092	DMA source address register 2H	DSAH2	R/W			×	undefined
FFFF F094	DMA destination address register 2L	DDAL2	R/W			×	undefined
FFFF F096	DMA destination address register 2H	DDAH2	R/W			×	undefined
FFFF F098	DMA source address register 3L	DSAL3	R/W			×	undefined
FFFF F09A	DMA source address register 3H	DSAH3	R/W			×	undefined
FFFF F09C	DMA destination address register 3L	DDAL3	R/W			×	undefined
FFFF F09E	DMA destination address register 3H	DDAH3	R/W			×	undefined

Table 3-6: List of Peripheral I/O Registers (2/7)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
FFFF F0C0	DMA transfer count register 0	DBC0	R/W			×	undefined
FFFF F0C2	DMA transfer count register 1	DBC1	R/W			×	undefined
FFFF F0C4	DMA transfer count register 2	DBC2	R/W			×	undefined
FFFF F0C6	DMA transfer count register 3	DBC3	R/W			×	undefined
FFFF F0D0	DMA addressing control register 0	DADC0	R/W			×	0000H
FFFF F0D2	DMA addressing control register 1	DADC1	R/W			×	0000H
FFFF F0D4	DMA addressing control register 2	DADC2	R/W			×	0000H
FFFF F0D6	DMA addressing control register 3	DADC3	R/W			×	0000H
FFFF F0E0	DMA channel control register 0	DCHC0	R/W	×	×		00H
FFFF F0E2	DMA channel control register 1	DCHC1	R/W	×	×		00H
FFFF F0E4	DMA channel control register 2	DCHC2	R/W	×	×		00H
FFFF F0E6	DMA channel control register 3	DCHC3	R/W	×	×		00H
FFFF F0F0	DMA disable status register	DDIS	R	×	×		00H
FFFF F0F2	DMA restart register	DRST	R/W	×	×		00H
FFFF F100	Interrupt Mask register 0	IMR0	R/W	×		×	FFFFH
FFFF F102	Interrupt Mask register 1	IMR1	R/W	×		×	FFFFH
FFFF F104	Interrupt Mask register 2	IMR2	R/W	×		×	FFFFH
FFFF F106	Interrupt Mask register 3	IMR3	R/W	×		×	FFFFH
FFFF F110	Interrupt control register 0	WTIC	R/W	×	×		47H
FFFF F112	Interrupt control register 1	TMD0IC	R/W	×	×		47H
FFFF F114	Interrupt control register 2	TMD1IC	R/W	×	×		47H
FFFF F116	Interrupt control register 3	WTIIC	R/W	×	×		47H
FFFF F118	Interrupt control register 4	P0IC	R/W	×	×		47H
FFFF F11A	Interrupt control register 5	P1IC	R/W	×	×		47H
FFFF F11C	Interrupt control register 6	P2IC	R/W	×	×		47H
FFFF F11E	Interrupt control register 7	P3IC	R/W	×	×		47H
FFFF F120	Interrupt control register 8	P4IC	R/W	×	×		47H
FFFF F122	Interrupt control register 9	P5IC	R/W	×	×		47H
FFFF F124	Interrupt control register 10	TMG00IC	R/W	×	×		47H
FFFF F126	Interrupt control register 11	TMG01IC	R/W	×	×		47H
FFFF F128	Interrupt control register 12	CCG00 IC	R/W	×	×		47H
FFFF F12A	Interrupt control register 13	CCG01IC	R/W	×	×		47H
FFFF F12C	Interrupt control register 14	CCG02IC	R/W	×	×		47H
FFFF F12E	Interrupt control register 15	CCG03IC	R/W	×	×		47H
FFFF F130	Interrupt control register 16	CCG04IC	R/W	×	×		47H
FFFF F132	Interrupt control register 17	CCG05IC	R/W	×	×		47H
FFFF F134	Interrupt control register 18	TMG10IC	R/W	×	×		47H
FFFF F136	Interrupt control register 19	TMG11IC	R/W	×	×		47H
FFFF F138	Interrupt control register 20	CCG10 IC	R/W	×	×		47H
FFFF F13A	Interrupt control register 21	CCG11IC	R/W	×	×		47H
FFFF F13C	Interrupt control register 22	CCG12IC	R/W	×	×		47H

Table 3-6: List of Peripheral I/O Registers (3/7)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
FFFF F13E	Interrupt control register 23	CCG13IC	R/W	×	×		47H
FFFF F140	Interrupt control register 24	CCG14IC	R/W	×	×		47H
FFFF F142	Interrupt control register 25	CCG15IC	R/W	×	×		47H
FFFF F144	Interrupt control register 26	TMC0IC	R/W	×	×		47H
FFFF F146	Interrupt control register 27	CCC0 IC	R/W	×	×		47H
FFFF F148	Interrupt control register 28	CCC1IC	R/W	×	×		47H
FFFF F14A	Interrupt control register 29	ADIC	R/W	×	×		47H
FFFF F14C	Interrupt control register 30	MACIC	R/W	×	×		47H
FFFF F14E	Interrupt control register 31	FC1RXIC	R/W	×	×		47H
FFFF F150	Interrupt control register 32	FC1TXIC	R/W	×	×		47H
FFFF F152	Interrupt control register 33	FC1ERIC	R/W	×	×		47H
FFFF F154	Interrupt control register 34	FC2RXIC	R/W	×	×		47H
FFFF F156	Interrupt control register 35	FC2TXIC	R/W	×	×		47H
FFFF F158	Interrupt control register 36	FC2ERIC	R/W	×	×		47H
FFFF F15A	Interrupt control register 37	FC3RXIC	R/W	×	×		47H
FFFF F15C	Interrupt control register 38	FC3TXIC	R/W	×	×		47H
FFFF F15E	Interrupt control register 39	FC3ERIC	R/W	×	×		47H
FFFF F160	Interrupt control register 40	FC4RXIC	R/W	×	×		47H
FFFF F162	Interrupt control register 41	FC4TXIC	R/W	×	×		47H
FFFF F164	Interrupt control register 42	FC4ERIC	R/W	×	×		47H
FFFF F166	Interrupt control register 43	CSI0IC	R/W	×	×		47H
FFFF F168	Interrupt control register 44	CSI1IC	R/W	×	×		47H
FFFF F16A	Interrupt control register 45	CSI2IC	R/W	×	×		47H
FFFF F16C	Interrupt control register 46	SER0IC	R/W	×	×		47H
FFFF F16E	Interrupt control register 47	SR0IC	R/W	×	×		47H
FFFF F170	Interrupt control register 48	ST0IC	R/W	×	×		47H
FFFF F172	Interrupt control register 49	SER1IC	R/W	×	×		47H
FFFF F174	Interrupt control register 50	SR1IC	R/W	×	×		47H
FFFF F176	Interrupt control register 51	ST1IC	R/W	×	×		47H
FFFF F178	Interrupt control register 52	DMA0IC	R/W	×	×		47H
FFFF F17A	Interrupt control register 53	DMA1IC	R/W	×	×		47H
FFFF F17C	Interrupt control register 54	DMA2IC	R/W	×	×		47H
FFFF F17E	Interrupt control register 55	DMA3IC	R/W	×	×		47H
FFFF F180	Interrupt control register 56	DOVFIC	R/W	×	×		47H
FFFF F182	Interrupt control register 57	P00IC	R/W	×	×		47H
FFFF F184	Interrupt control register 58	P05IC	R/W	×	×		47H
FFFF F186	Interrupt control register 59	P10IC	R/W	×	×		47H
FFFF F188	Interrupt control register 60	P15IC	R/W	×	×		47H
FFFF F18A	Interrupt control register 61	P20IC	R/W	×	×		47H
FFFF F18C	Interrupt control register 62	P21IC	R/W	×	×		47H
FFFF F18E	Interrupt control register 63	Reserved	R/W	×	×		47H

Table 3-6: List of Peripheral I/O Registers (4/7)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
FFFF F1FA	In-service Priority register	ISPR	R	×	×		00H
FFFF F1FC	Command register	PRCMD	W		×		undefined
FFFF F1FE	Power Save Control register	PSC	R/W	×	×		00H
FFFF F200	A/D converter mode register	ADM	R/W	×	×		00H
FFFF F201	A/D Select Register	ADS	R/W	×	×		00H
FFFF F202	A/D Conversion Result Register	ADCR	R	×	×	×	undefined
FFFF F203	A/D Conversion Result Register H	ADCRH	R	×	×		undefined
FFFF F400	Port 1	P1	R/W	×	×		undefined
FFFF F402	Port 2	P2	R/W	×	×		undefined
FFFF F404	Port 3	P3	R/W	×	×		undefined
FFFF F406	Port 4	P4	R/W	×	×		undefined
FFFF F408	Port 5	P5	R/W	×	×		undefined
FFFF F40A	Port 6	P6	R/W	×	×		undefined
FFFF F40C	Port 7	P7	R	×	×		undefined
FFFF F40C	Port 7/Port 8	P78	R			×	undefined
FFFF F40E	Port 9	P9	R/W	×	×		undefined
FFFF F420	Port 1 mode	PM1	R/W	×	×		FFH
FFFF F422	Port 2 mode	PM2	R/W	×	×		FFH
FFFF F424	Port 3 mode	PM3	R/W	×	×		3FH
FFFF F426	Port 4 mode	PM4	R/W	×	×		3FH
FFFF F428	Port 5 mode	PM5	R/W	×	×		7FH
FFFF F42A	Port 6 mode	PM6	R/W	×	×		FFH
FFFF F42E	Port 9 mode	PM9	R/W	×	×		FFH
FFFF F440	Port 1 mode control	PMC1	R/W	×	×		00H
FFFF F442	Port 2 mode control	PMC2	R/W	×	×		00H
FFFF F444	Port 3 mode control	PMC3	R/W	×	×		00H
FFFF F446	Port 4 mode control	PMC4	R/W	×	×		00H
FFFF F448	Port 5 mode control	PMC5	R/W	×	×		00H
FFFF F44A	Port 6 mode control	PMC6	R/W	×	×		00H
FFFF F480	MEMC: Bus Cycle Type Control register 0	BCT0	R/W			×	8888H
FFFF F482	MEMC: Bus Cycle Type Control register 1	BCT1	R/W			×	8888H
FFFF F484	MEMC: Data Wait Control register 0	DWC0	R/W			×	7777H
FFFF F486	MEMC: Data Wait Control register 1	DWC1	R/W			×	7777H
FFFF F488	MEMC: Bus Cycle Control register	BCC	R/W			×	FFFFH
FFFF F48A	MEMC: Address Setup Wait Control Reg.	ASC	R/W			×	FFFFH
FFFF F49A	MEMC: Page-ROM Configuration register	PRC	R/W			×	7000H
FFFF F540	Timer D0 counter	TMD0	R			×	0000H
FFFF F542	Timer D0 compare register	CMD0	R/W			×	0000H
FFFF F544	Timer D0 Control register	TMCD0	R/W	×	×		00H
FFFF F550	Timer D1 counter	TMD1	R			×	0000H
FFFF F552	Timer D1 compare register	CMD1	R/W			×	0000H

Table 3-6: List of Peripheral I/O Registers (5/7)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
FFFF F554	Timer D1 Control register	TMCD1	R/W	×	×		00H
FFFF F560	Watch timer mode register	WTM	R/W	×	×		00H
FFFF F571	Watchdog timer time select register	WDCS	R/W	×	×		00H
FFFF F572	Watchdog timer mode register	WDTM	R/W	×	×		00H
FFFF F580	Watchdog Timer command register	WCMD	W		×		undefined
FFFF F582	Watchdog Timer command status register	WPHS	R/W	×	×		00H
FFFF F600	Timer C0 timer counter register	TMC0	R			×	0000H
FFFF F602	Timer C0 capture compare register 0	CCC00	R/W			×	0000H
FFFF F604	Timer C0 capture compare register 1	CCC01	R/W			×	0000H
FFFF F606	Timer C0 control register 0	TMCC00	R/W	×	×		00H
FFFF F608	Timer C0 control register 1	TMCC01	R/W	×	×		20H
FFFF F609	Timer C0 signal edge select register	SESC0	R/W	×	×		00H
FFFF F640	Timer mode register	TMGM0	R/W	×	×	×	0000H
		TMGM0L	R/W	×	×		00H
FFFF F641		TMGM0H	R/W	×	×		00H
FFFF F642	Channel mode register	TMGCM0	R/W	×	×	×	0000H
		TMGCM0L	R/W	×	×		00H
FFFF F643		TMGCM0H	R/W	×	×		00H
FFFF F644	Output control register	OCTLG0	R/W	×	×	×	0000H
		OCTLG0L	R/W	×	×		00H
FFFF F645		OCTLG0H	R/W	×	×		00H
FFFF F646	State register	TMGST0	R	×	×		00H
FFFF F648	Timer Count Register 0	TMG00	R			×	0000H
FFFF F64A	Timer Count Register 1	TMG01	R			×	0000H
FFFF F64C	Capture/Compare register 0	GCC00	R/W			×	0000H
FFFF F64E	Capture/Compare register 1	GCC01	R/W			×	0000H
FFFF F650	Capture/Compare register 2	GCC02	R/W			×	0000H
FFFF F652	Capture/Compare register 3	GCC03	R/W			×	0000H
FFFF F654	Capture/Compare register 4	GCC04	R/W			×	0000H
FFFF F656	Capture/Compare register 5	GCC05	R/W			×	0000H
FFFF F680	Timer mode register	TMGM1	R/W	×	×	×	0000H
		TMGM1L	R/W	×	×		00H
FFFF F681		TMGM1H	R/W	×	×		00H
FFFF F682	Channel mode register	TMGCM1	R/W	×	×	×	0000H
		TMGCM1L	R/W	×	×		00H
FFFF F683		TMGCM1H	R/W	×	×		00H
FFFF F684	Output control register	OCTLG1	R/W	×	×	×	0000H
		OCTLG1L	R/W	×	×		00H
FFFF F685		OCTLG1H	R/W	×	×		00H
FFFF F686	State register	TMGST1	R	×	×		00H
FFFF F688	Timer Count Register 0	TMG10	R			×	0000H

Table 3-6: List of Peripheral I/O Registers (6/7)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
FFFF F68A	Timer Count Register 1	TMG11	R			×	0000H
FFFF F68C	Capture/Compare register 0	GCC10	R/W			×	0000H
FFFF F68E	Capture/Compare register 1	GCC11	R/W			×	0000H
FFFF F690	Capture/Compare register 2	GCC12	R/W			×	0000H
FFFF F692	Capture/Compare register 3	GCC13	R/W			×	0000H
FFFF F694	Capture/Compare register 4	GCC14	R/W			×	0000H
FFFF F696	Capture/Compare register 5	GCC15	R/W			×	0000H
FFFF F800	Peripheral command register	PHCMD	W		×		undefined
FFFF F802	Peripheral status register	PHS	R/W	×	×		00H
FFFF F820	Power save mode register	PSM	R/W	×	×		00H
FFFF F822	Clock control register	CKC	R/W	×	×		00H
FFFF F824	Clock generator status register	CGSTAT	R	×	×		00H
FFFF F826	Watch dog clock control register	WCC	R/W	×	×		00H
FFFF F828	Processor clock control register	PCC	R/W	×	×		00H
FFFF F82A	Frequency modulation control register	SCFMC	R/W	×	×		0AH
FFFF F82C	Frequency control 0	SCFC0	R/W	×	×		3FH
FFFF F82E	Frequency control 1	SCFC1	R/W	×	×		40H
FFFF F830	Reset source monitor register	RSM	R/W	×	×		00H/01H
FFFF F840	DMA trigger source select register 0	DTFR0	R/W	×	×		00H
FFFF F842	DMA trigger source select register 1	DTFR1	R/W	×	×		00H
FFFF F844	DMA trigger source select register 2	DTFR2	R/W	×	×		00H
FFFF F846	DMA trigger source select register 3	DTFR3	R/W	×	×		00H
FFFF F880	Interrupt Mode Control Register 0	INTM0	R/W	×	×		00H
FFFF F882	Interrupt Mode Control Register 1	INTM1	R/W	×	×		00H
FFFF F884	Interrupt Mode Control Register 2	INTM2	R/W	×	×		00H
FFFF F886	Interrupt Mode Control Register 3	INTM3	R/W	×	×		00H
FFFF FA00	UART operation mode register	ASIM0	R/W	×	×		01H
FFFF FA02	Reception buffer register	RXB0	R		×	RXB_ASIS0	FFH
FFFF FA03	UART reception error status register	ASIS0	R		×		00H
FFFF FA04	Transmission buffer register	TXB0	R/W		×		FFH
FFFF FA05	UART transmission error status register	ASIF0	R		×		00H
FFFF FA06	Clock selection register	CHKSR0	R/W	×	×	CHKSR_BRGC0	00H
FFFF FA07	Baudrate definition register	BRGC0	R/W	×	×		FFH
FFFF FA40	UART operation mode register	ASIM1	R/W	×	×		01H
FFFF FA42	Reception buffer register	RXB1	R		×	RXB_ASIS1	FFH
FFFF FA43	UART reception error status register	ASIS1	R		×		00H
FFFF FA44	Transmission buffer register	TXB1	R/W		×		FFH
FFFF FA45	UART transmission error status register	ASIF1	R		×		00H

Table 3-6: List of Peripheral I/O Registers (7/7)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation 1-bit	8-bit	16-bit	Initial Value
FFFF FA46	Clock selection register	CHKSR1	R/W	×	×	CHKSR_ BRGC1	00H
FFFF FA47	Baudrate definition register	BRGC1	R/W	×	×		FFH
FFFF FD00	CSI operation mode register	CSIM0	R/W	×	×	CSIM_C SIC0	00H
FFFF FD01	Clock selection register	CSIC0	R/W	×	×		00H
FFFF FD02	Reception data buffer register	SIRB0/ SIRBL0	R/O		×	×	0000H/ 00H
FFFF FD04	Transmission data buffer register	SOTB0/ SOTBL0	R/W		×	×	0000H/ 00H
FFFF FD08	First transmission data buffer register	SOTBF0/ SOTBFL0	R/W		×	×	0000H/ 00H
FFFF FD0A	Shift register	SIO0/ SIOL0	R/O		×	×	0000H/ 00H
FFFF FD40	CSI operation mode register	CSIM1	R/W	×	×	CSIM_C SIC1	00H
FFFF FD41	Clock selection register	CSIC1	R/W	×	×		00H
FFFF FD42	Reception data buffer register	SIRB1/ SIRBL1	R/O		×	×	0000H/ 00H
FFFF FD44	Transmission data buffer register	SOTB1/ SOTBL1	R/W		×	×	0000H/ 00H
FFFF FD48	First transmission data buffer register	SOTBF1/ SOTBFL1	R/W		×	×	0000H/ 00H
FFFF FD4A	Shift register	SIO1/ SIOL1	R/O		×	×	0000H/ 00H
FFFF FD80	CSI operation mode register	CSIM2	R/W	×	×	CSIM_C SIC2	00H
FFFF FD81	Clock selection register	CSIC2	R/W	×	×		00H
FFFF FD82	Reception data buffer register	SIRB2/ SIRBL2	R/O		×	×	0000H/ 00H
FFFF FD84	Transmission data buffer register	SOTB2/ SOTBL2	R/W		×	×	0000H/ 00H
FFFF FD88	First transmission data buffer register	SOTBF2/ SOTBFL2	R/W		×	×	0000H/ 00H
FFFF FD8A	Shift register	SIO2/ SIOL2	R/O		×	×	0000H/ 00H
FFFF FDC0	BRG0 Prescaler mode register	PRSM0	R/W	×	×		00H
FFFF FDC1	BRG0 Prescaler compare register	PRSCM0	R/W	×	×		00H
FFFF FDE0	BRG1 Prescaler mode register	PRSM1	R/W	×	×		00H
FFFF FDE1	BRG1 Prescaler compare register	PRSCM1	R/W	×	×		00H

3.5.4 Programmable peripheral I/O registers

In the V850E/CA2, the 16 KB area of x0000H to x3FFFH is provided as a programmable peripheral I/O area. In this area, the area between x0000H and x1200H is used exclusively for the FCAN controller.

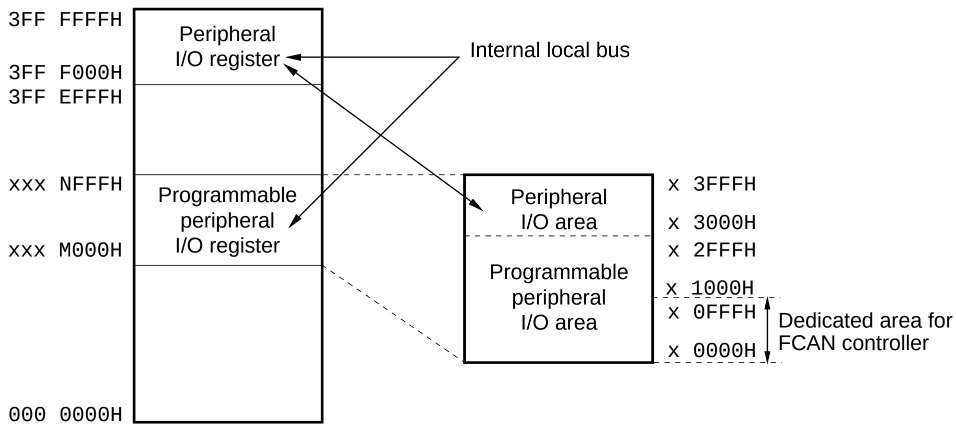
The internal bus of the V850E/CA2 becomes active when
- the peripheral I/O register area (3FF F000H to 3FF FFFFH) or
- the programmable peripheral I/O register area (xxxx m000H to xxxx nFFFH)

is accessed (m = xx00B, n = xx11B).

Note that when data is written to the peripheral I/O register area, the written contents are reflected also on the upper 4 KB area of the programmable peripheral I/O area.

The base address of the programmable peripheral I/O area is specified by the initialization of the peripheral area selection control register (BPC).

Figure 3-15: Programmable Peripheral I/O Register (Outline)



Cautions: 1. The CAN message buffer register can allocate address xxxx freely as a programmable peripheral I/O register. but once the address xxxx is set, it cannot be changed.

2. If the programmable peripheral I/O area overlaps the following areas, the programmable peripheral I/O area becomes ineffective.

- **Peripheral I/O area**
- **ROM area**
- **RAM area**

Remark: M = xx00B
N = xx11B

(1) Peripheral area selection control register (BPC)

The BPC register is a 16-bit register that specifies the base address or the programmable peripheral area.

This register can be read/written in 16-bit units.

Figure 3-16: Peripheral Area Selection Control Register (BPC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BPC	PA15	0	PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	FFFF F064H	0000H

Bit Position	Bit Name	Function
15	PA15	Enables/disables usage of programmable peripheral I/O area. PA15 / Usage of Programmable Peripheral I/O Area 0: Disables usage of programmable peripheral I/O area 1: Enables usage of programmable peripheral I/O area
13 to 0	PA13 to PA0	Specifies an address in programmable peripheral I/O area (corresponds to A27 to A14, respectively).

Remark: For V850E/CA2, the recommended setting of the BPC setting is 8600H.
With that initialization the base address of the programmable peripheral area is located at 180 0000H.
Therefore the FCAN macro is mapped to the memory location 180 0000H to 180 11FFH.

A list of the programmable peripheral I/O registers is shown below:

Table 3-7: List of programmable peripheral I/O registers (1/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn000H	CAN message event pointer 000	M_EVT000	R/W		x		Undefined
xxxxn001H	CAN message event pointer 001	M_EVT001	R/W		x		Undefined
xxxxn002H	CAN message event pointer 002	M_EVT002	R/W		x		Undefined
xxxxn003H	CAN message event pointer 003	M_EVT003	R/W		x		Undefined
xxxxn004H	CAN message data length register 00	M_DLC00	R/W		x		Undefined
xxxxn005H	CAN message control register 00	M_CTRL00	R/W		x		Undefined
xxxxn006H	CAN message time stamp register 00	M_TIME00	R/W			x	Undefined
xxxxn008H	CAN message data register 000	M_DATA000	R/W		x	x	Undefined
xxxxn009H	CAN message data register 001	M_DATA001	R/W		x		Undefined
xxxxn00AH	CAN message data register 002	M_DATA002	R/W		x		Undefined
xxxxn00BH	CAN message data register 003	M_DATA003	R/W		x		Undefined
xxxxn00CH	CAN message data register 004	M_DATA004	R/W		x		Undefined
xxxxn00DH	CAN message data register 005	M_DATA005	R/W		x		Undefined
xxxxn00EH	CAN message data register 006	M_DATA006	R/W		x		Undefined
xxxxn00FH	CAN message data register 007	M_DATA007	R/W		x		Undefined
xxxxn010H	CAN message ID register L00	M_IDL00	R/W			x	Undefined
xxxxn012H	CAN message ID register H00	M_IDH00	R/W			x	Undefined
xxxxn014H	CAN message configuration register 00	M_CONF00	R/W		x		Undefined
xxxxn015H	CAN message status register 00	M_STAT00	R		x		Undefined
xxxxn016H	CAN status set/cancel register 00	SC_STAT00	W			x	0000H
xxxxn020H	CAN message event pointer 010	M_EVT010	R/W		x		Undefined
xxxxn021H	CAN message event pointer 011	M_EVT011	R/W		x		Undefined
xxxxn022H	CAN message event pointer 012	M_EVT012	R/W		x		Undefined
xxxxn023H	CAN message event pointer 013	M_EVT013	R/W		x		Undefined
xxxxn024H	CAN message data length register 01	M_DLC01	R/W		x		Undefined
xxxxn025H	CAN message control register 01	M_CTRL01	R/W		x		Undefined
xxxxn026H	CAN message time stamp register 01	M_TIME01	R/W			x	Undefined
xxxxn028H	CAN message data register 010	M_DATA010	R/W		x		Undefined
xxxxn029H	CAN message data register 011	M_DATA011	R/W		x		Undefined
xxxxn02AH	CAN message data register 012	M_DATA012	R/W		x		Undefined
xxxxn02BH	CAN message data register 013	M_DATA013	R/W		x		Undefined
xxxxn02CH	CAN message data register 014	M_DATA014	R/W		x		Undefined
xxxxn02DH	CAN message data register 015	M_DATA015	R/W		x		Undefined
xxxxn02EH	CAN message data register 016	M_DATA016	R/W		x		Undefined
xxxxn02FH	CAN message data register 017	M_DATA017	R/W		x		Undefined
xxxxn030H	CAN message ID register L01	M_IDL01	R/W			x	Undefined
xxxxn032H	CAN message ID register H01	M_IDH01	R/W			x	Undefined
xxxxn034H	CAN message configuration register 01	M_CONF01	R/W		x		Undefined
xxxxn035H	CAN message status register 01	M_STAT01	R		x		Undefined
xxxxn036H	CAN status set/cancel register 01	SC_STAT01	W			x	0000H

Table 3-7: List of programmable peripheral I/O registers (2/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn040H	CAN message event pointer 020	M_EVT020	R/W		x		Undefined
xxxxn041H	CAN message event pointer 021	M_EVT021	R/W		x		Undefined
xxxxn042H	CAN message event pointer 022	M_EVT022	R/W		x		Undefined
xxxxn043H	CAN message event pointer 023	M_EVT023	R/W		x		Undefined
xxxxn044H	CAN message data length register 02	M_DLC02	R/W		x		Undefined
xxxxn045H	CAN message control register 02	M_CTRL02	R/W		x		Undefined
xxxxn046H	CAN message time stamp register 02	M_TIME02	R/W			x	Undefined
xxxxn048H	CAN message data register 020	M_DATA020	R/W		x		Undefined
Xxxxn049H	CAN message data register 021	M_DATA021	R/W		x		Undefined
xxxxn04AH	CAN message data register 022	M_DATA022	R/W		x		Undefined
Xxxxn04BH	CAN message data register 023	M_DATA023	R/W		x		Undefined
xxxxn04CH	CAN message data register 024	M_DATA024	R/W		x		Undefined
xxxxn04DH	CAN message data register 025	M_DATA025	R/W		x		Undefined
xxxxn04EH	CAN message data register 026	M_DATA026	R/W		x		Undefined
xxxxn04FH	CAN message data register 027	M_DATA027	R/W		x		Undefined
xxxxn050H	CAN message ID register L02	M_IDL02	R/W			x	Undefined
xxxxn052H	CAN message ID register H02	M_IDH02	R/W			x	Undefined
xxxxn054H	CAN message configuration register 02	M_CONF02	R/W		x		Undefined
xxxxn055H	CAN message status register 02	M_STAT02	R		x		Undefined
xxxxn056H	CAN status set/cancel register 02	SC_STAT02	W			x	0000H
xxxxn060H	CAN message event pointer 030	M_EVT030	R/W		x		Undefined
xxxxn061H	CAN message event pointer 031	M_EVT031	R/W		x		Undefined
xxxxn062H	CAN message event pointer 032	M_EVT032	R/W		x		Undefined
xxxxn063H	CAN message event pointer 033	M_EVT033	R/W		x		Undefined
xxxxn064H	CAN message data length register 03	M_DLC03	R/W		x		Undefined
xxxxn065H	CAN message control register 03	M_CTRL03	R/W		x		Undefined
xxxxn066H	CAN message time stamp register 03	M_TIME03	R/W			x	Undefined
xxxxn068H	CAN message data register 030	M_DATA030	R/W		x		Undefined
xxxxn069H	CAN message data register 031	M_DATA031	R/W		x		Undefined
xxxxn06AH	CAN message data register 032	M_DATA032	R/W		x		Undefined
xxxxn06BH	CAN message data register 033	M_DATA033	R/W		x		Undefined
xxxxn06CH	CAN message data register 034	M_DATA034	R/W		x		Undefined
xxxxn06DH	CAN message data register 035	M_DATA035	R/W		x		Undefined
xxxxn06EH	CAN message data register 036	M_DATA036	R/W		x		Undefined
xxxxn06FH	CAN message data register 037	M_DATA037	R/W		x		Undefined
xxxxn070H	CAN message ID register L03	M_IDL03	R/W			x	Undefined
xxxxn072H	CAN message ID register H03	M_IDH03	R/W			x	Undefined
xxxxn074H	CAN message configuration register 03	M_CONF03	R/W		x		Undefined
xxxxn075H	CAN message status register 03	M_STAT03	R		x		Undefined
xxxxn076H	CAN status set/cancel register 03	SC_STAT03	W			x	0000H
xxxxn080H	CAN message event pointer 040	M_EVT040	R/W		x		Undefined
xxxxn081H	CAN message event pointer 041	M_EVT041	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (3/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn082H	CAN message event pointer 042	M_EVT042	R/W		x		Undefined
xxxxn083H	CAN message event pointer 043	M_EVT043	R/W		x		Undefined
xxxxn084H	CAN message data length register 04	M_DLC04	R/W		x		Undefined
xxxxn085H	CAN message control register 04	M_CTRL04	R/W		x		Undefined
xxxxn086H	CAN message time stamp register 04	M_TIME04	R/W			x	Undefined
xxxxn088H	CAN message data register 040	M_DATA040	R/W		x		Undefined
xxxxn089H	CAN message data register 041	M_DATA041	R/W		x		Undefined
xxxxn08AH	CAN message data register 042	M_DATA042	R/W		x		Undefined
xxxxn08BH	CAN message data register 043	M_DATA043	R/W		x		Undefined
xxxxn08CH	CAN message data register 044	M_DATA044	R/W		x		Undefined
xxxxn08DH	CAN message data register 045	M_DATA045	R/W		x		Undefined
xxxxn08EH	CAN message data register 046	M_DATA046	R/W		x		Undefined
xxxxn08FH	CAN message data register 047	M_DATA047	R/W		x		Undefined
Xxxxn090H	CAN message ID register L04	M_IDL04	R/W			x	Undefined
xxxxn092H	CAN message ID register H04	M_IDH04	R/W			x	Undefined
xxxxn094H	CAN message configuration register 04	M_CONF04	R/W		x		Undefined
xxxxn095H	CAN message status register 04	M_STAT04	R		x		Undefined
xxxxn096H	CAN status set/cancel register 04	M_STAT04	W			x	0000H
xxxxn0A0H	CAN message event pointer 050	M_EVT050	R/W		x		Undefined
xxxxn0A1H	CAN message event pointer 051	M_EVT051	R/W		x		Undefined
xxxxn0A2H	CAN message event pointer 052	M_EVT052	R/W		x		Undefined
xxxxn0A3H	CAN message event pointer 053	M_EVT053	R/W		x		Undefined
xxxxn0A4H	CAN message data length register 05	M_DLC05	R/W		x		Undefined
xxxxn0A5H	CAN message control register 05	M_DTRL05	R/W		x		Undefined
xxxxn0A6H	CAN message time stamp register 05	M_TIME05	R/W			x	Undefined
xxxxn0A8H	CAN message data register 050	M_DATA050	R/W		x		Undefined
xxxxn0A9H	CAN message data register 051	M_DATA051	R/W		x		Undefined
xxxxn0AAH	CAN message data register 052	M_DATA052	R/W		x		Undefined
xxxxn0ABH	CAN message data register 053	M_DATA053	R/W		x		Undefined
xxxxn0ACH	CAN message data register 054	M_DATA054	R/W		x		Undefined
xxxxn0ADH	CAN message data register 055	M_DATA055	R/W		x		Undefined
xxxxn0AEH	CAN message data register 056	M_DATA056	R/W		x		Undefined
xxxxn0AFH	CAN message data register 057	M_DATA057	R/W		x		Undefined
xxxxn0B0H	CAN message ID register L05	M_IDL05	R/W			x	Undefined
xxxxn0B2H	CAN message ID register H05	M_IDH05	R/W			x	Undefined
xxxxn0B4H	CAN message configuration register 05	M_CONF05	R/W		x		Undefined
xxxxn0B5H	CAN message status register 05	M_STAT05	R		x		Undefined
xxxxn0B6H	CAN status set/cancel register 05	SC_STAT05	W			x	0000H
xxxxn0C0H	CAN message event pointer 060	M_EVT060	R/W		x		Undefined
xxxxn0C1H	CAN message event pointer 061	M_EVT061	R/W		x		Undefined
xxxxn0C2H	CAN message event pointer 062	M_EVT062	R/W		x		Undefined
xxxxn0C3H	CAN message event pointer 063	M_EVT063	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (4/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn0C4H	CAN message data length register 06	M_DLC06	R/W		x		Undefined
xxxxn0C5H	CAN message control register 06	M_DTRL06	R/W		x		Undefined
xxxxn0C6H	CAN message time stamp register 06	M_TIME06	R/W			x	Undefined
xxxxn0C8H	CAN message data register 060	M_DATA060	R/W		x		Undefined
xxxxn0C9H	CAN message data register 061	M_DATA061	R/W		x		Undefined
xxxxn0CAH	CAN message data register 062	M_DATA062	R/W		x		Undefined
xxxxn0CBH	CAN message data register 063	M_DATA063	R/W		x		Undefined
xxxxn0CCH	CAN message data register 064	M_DATA064	R/W		x		Undefined
xxxxn0CDH	CAN message data register 065	M_DATA065	R/W		x		Undefined
xxxxn0CEH	CAN message data register 066	M_DATA066	R/W		x		Undefined
xxxxn0CFH	CAN message data register 067	M_DATA067	R/W		x		Undefined
xxxxn0D0H	CAN message ID register L06	M_IDL06	R/W			x	Undefined
xxxxn0D2H	CAN message ID register H06	M_IDH06	R/W			x	Undefined
xxxxn0D4H	CAN message configuration register 06	M_CONF06	R/W		x		Undefined
xxxxn0D5H	CAN message status register 06	M_STAT06	R		x		Undefined
xxxxn0D6H	CAN status set/cancel register 06	SC_STAT06	W			x	0000H
xxxxn0E0H	CAN message event pointer 070	M_EVT070	R/W		x		Undefined
xxxxn0E1H	CAN message event pointer 071	M_EVT071	R/W		x		Undefined
xxxxn0E2H	CAN message event pointer 072	M_EVT072	R/W		x		Undefined
xxxxn0E3H	CAN message event pointer 073	M_EVT073	R/W		x		Undefined
xxxxn0E4H	CAN message data length register 07	M_DLC07	R/W		x		Undefined
xxxxn0E5H	CAN message control register 07	M_CTRL07	R/W		x		Undefined
xxxxn0E6H	CAN message time stamp register 07	M_TIME07	R/W			x	Undefined
xxxxn0E8H	CAN message data register 070	M_DATA070	R/W		x		Undefined
xxxxn0E9H	CAN message data register 071	M_DATA071	R/W		x		Undefined
xxxxn0EAH	CAN message data register 072	M_DATA072	R/W		x		Undefined
xxxxn0EBH	CAN message data register 073	M_DATA073	R/W		x		Undefined
xxxxn0ECH	CAN message data register 074	M_DATA074	R/W		x		Undefined
xxxxn0EDH	CAN message data register 075	M_DATA075	R/W		x		Undefined
xxxxn0EEH	CAN message data register 076	M_DATA076	R/W		x		Undefined
xxxxn0EFH	CAN message data register 077	M_DATA077	R/W		x		Undefined
xxxxn0F0H	CAN message ID register L07	M_IDL07	R/W			x	Undefined
xxxxn0F2H	CAN message ID register H07	M_IDH07	R/W			x	Undefined
xxxxn0F4H	CAN message configuration register 07	M_CONF07	R/W		x		Undefined
xxxxn0F5H	CAN message status register 07	M_STAT07	R		x		Undefined
xxxxn0F6H	CAN status set/cancel register 07	SC_STAT07	W			x	0000H
xxxxn100H	CAN message event pointer 080	M_EVT080	R/W		x		Undefined
xxxxn101H	CAN message event pointer 081	M_EVT081	R/W		x		Undefined
xxxxn102H	CAN message event pointer 082	M_EVT082	R/W		x		Undefined
xxxxn103H	CAN message event pointer 083	M_EVT083	R/W		x		Undefined
xxxxn104H	CAN message data length register 08	M_DLC08	R/W		x		Undefined
xxxxn105H	CAN message control register 08	M_CTRL08	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (5/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn106H	CAN message time stamp register 08	M_TIME08	R/W			x	Undefined
xxxxn108H	CAN message data register 080	M_DATA080	R/W		x		Undefined
xxxxn109H	CAN message data register 081	M_DATA081	R/W		x		Undefined
xxxxn10AH	CAN message data register 082	M_DATA082	R/W		x		Undefined
xxxxn10BH	CAN message data register 083	M_DATA083	R/W		x		Undefined
xxxxn10CH	CAN message data register 084	M_DATA084	R/W		x		Undefined
xxxxn10DH	CAN message data register 085	M_DATA085	R/W		x		Undefined
xxxxn10EH	CAN message data register 086	M_DATA086	R/W		x		Undefined
xxxxn10FH	CAN message data register 087	M_DATA087	R/W		x		Undefined
xxxxn110H	CAN message ID register L08	M_IDL08	R/W			x	Undefined
xxxxn112H	CAN message ID register H08	M_IDH08	R/W			x	Undefined
xxxxn114H	CAN message configuration register 08	M_CONF08	R/W		x		Undefined
xxxxn115H	CAN message status register 08	M_STAT08	R		x		Undefined
xxxxn116H	CAN status set/cancel register 08	SC_STAT08	W			x	0000H
xxxxn120H	CAN message event pointer 090	M_EVT090	R/W		x		Undefined
xxxxn121H	CAN message event pointer 091	M_EVT091	R/W		x		Undefined
xxxxn122H	CAN message event pointer 092	M_EVT092	R/W		x		Undefined
xxxxn123H	CAN message event pointer 093	M_EVT093	R/W		x		Undefined
xxxxn124H	CAN message data length register 09	M_DLC09	R/W		x		Undefined
xxxxn125H	CAN message control register 09	M_CTRL09	R/W		x		Undefined
xxxxn126H	CAN message time stamp register 09	M_TIME09	R/W			x	Undefined
xxxxn128H	CAN message data register 090	M_DATA090	R/W		x		Undefined
xxxxn129H	CAN message data register 091	M_DATA091	R/W		x		Undefined
xxxxn12AH	CAN message data register 092	M_DATA092	R/W		x		Undefined
xxxxn12BH	CAN message data register 093	M_DATA093	R/W		x		Undefined
xxxxn12CH	CAN message data register 094	M_DATA094	R/W		x		Undefined
xxxxn12DH	CAN message data register 095	M_DATA095	R/W		x		Undefined
xxxxn12EH	CAN message data register 096	M_DATA096	R/W		x		Undefined
xxxxn12FH	CAN message data register 097	M_DATA097	R/W		x		Undefined
xxxxn130H	CAN message ID register L09	M_IDL09	R/W			x	Undefined
xxxxn132H	CAN message ID register H09	M_IDH09	R/W			x	Undefined
xxxxn134H	CAN message configuration register 09	M_CONF09	R/W		x		Undefined
xxxxn135H	CAN message status register 09	M_STAT09	R		x		Undefined
xxxxn136H	CAN status set/cancel register 09	SC_STAT09	W			x	0000H
xxxxn140H	CAN message event pointer 100	M_EVT100	R/W		x		Undefined
xxxxn141H	CAN message event pointer 101	M_EVT101	R/W		x		Undefined
xxxxn142H	CAN message event pointer 102	M_EVT102	R/W		x		Undefined
xxxxn143H	CAN message event pointer103	M_EVT103	R/W		x		Undefined
xxxxn144H	CAN message data length register 10	M_DLC10	R/W		x		Undefined
xxxxn145H	CAN message control register 10	M_CTRL10	R/W		x		Undefined
xxxxn146H	CAN message time stamp register 10	M_TIME10	R/W			x	Undefined
xxxxn148H	CAN message data register 100	M_DATA100	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (6/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn149H	CAN message data register 101	M_DATA101	R/W		x		Undefined
xxxxn14AH	CAN message data register 102	M_DATA102	R/W		x		Undefined
xxxxn14BH	CAN message data register 103	M_DATA103	R/W		x		Undefined
xxxxn14CH	CAN message data register 104	M_DATA104	R/W		x		Undefined
xxxxn14DH	CAN message data register 105	M_DATA105	R/W		x		Undefined
xxxxn14EH	CAN message data register 106	M_DATA106	R/W		x		Undefined
xxxxn14FH	CAN message data register 107	M_DATA107	R/W		x		Undefined
xxxxn150H	CAN message ID register L10	M_IDL10	R/W			x	Undefined
xxxxn152H	CAN message ID register H10	M_IDH10	R/W			x	Undefined
xxxxn154H	CAN message configuration register 10	M_CONF10	R/W		x		Undefined
xxxxn155H	CAN message status register 10	M_STAT10	R		x		Undefined
xxxxn156H	CAN status set/cancel register 10	SC_STAT10	W			x	0000H
xxxxn160H	CAN message event pointer 110	M_EVT110	R/W		x		Undefined
xxxxn161H	CAN message event pointer111	M_EVT111	R/W		x		Undefined
xxxxn162H	CAN message event pointer 112	M_EVT112	R/W		x		Undefined
xxxxn163H	CAN message event pointer 113	M_EVT113	R/W		x		Undefined
xxxxn164H	CAN message data length register 11	M_DLC11	R/W		x		Undefined
xxxxn165H	CAN message control register 11	M_CTRL11	R/W		x		Undefined
xxxxn166H	CAN message time stamp register 11	M_TIME11	R/W			x	Undefined
xxxxn168H	CAN message data register 110	M_DATA110	R/W		x		Undefined
xxxxn169H	CAN message data register 111	M_DATA111	R/W		x		Undefined
xxxxn16AH	CAN message data register 112	M_DATA112	R/W		x		Undefined
xxxxn16BH	CAN message data register 113	M_DATA113	R/W		x		Undefined
xxxxn16CH	CAN message data register 114	M_DATA114	R/W		x		Undefined
xxxxn16DH	CAN message data register 115	M_DATA115	R/W		x		Undefined
xxxxn16EH	CAN message data register 116	M_DATA116	R/W		x		Undefined
xxxxn16FH	CAN message data register 117	M_DATA117	R/W		x		Undefined
xxxxn170H	CAN message ID register L11	M_IDL11	R/W			x	Undefined
xxxxn172H	CAN message ID register H11	M_IDH11	R/W			x	Undefined
xxxxn174H	CAN message configuration register 11	M_CONF11	R/W		x		Undefined
xxxxn175H	CAN message status register 11	M_STAT11	R		x		Undefined
xxxxn176H	CAN status set/cancel register 11	SC_STAT11	W			x	0000H
xxxxn180H	CAN message event pointer 120	M_EVT120	R/W		x		Undefined
xxxxn181H	CAN message event pointer 121	M_EVT121	R/W		x		Undefined
xxxxn182H	CAN message event pointer 122	M_EVT122	R/W		x		Undefined
xxxxn183H	CAN message event pointer 123	M_EVT123	R/W		x		Undefined
xxxxn184H	CAN message data length register 12	M_DLC12	R/W		x		Undefined
xxxxn185H	CAN message control register 12	M_CTRL12	R/W		x		Undefined
xxxxn186H	CAN message time stamp register 12	M_TIME12	R/W			x	Undefined
xxxxn188H	CAN message data register 120	M_DATA120	R/W		x		Undefined
xxxxn189H	CAN message data register 121	M_DATA121	R/W		x		Undefined
xxxxn18AH	CAN message data register 122	M_DATA122	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (7/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation 1-bit	8-bit	16-bit	Initial Value
xxxxn18BH	CAN message data register 123	M_DATA123	R/W		x		Undefined
xxxxn18CH	CAN message data register 124	M_DATA124	R/W		x		Undefined
xxxxn18DH	CAN message data register 125	M_DATA125	R/W		x		Undefined
xxxxn18EH	CAN message data register 126	M_DATA126	R/W		x		Undefined
xxxxn18FH	CAN message data register 127	M_DATA127	R/W		x		Undefined
xxxxn190H	CAN message ID register L12	M_IDL12	R/W			x	Undefined
xxxxn192H	CAN message ID register H12	M_IDH12	R/W			x	Undefined
xxxxn194H	CAN message configuration register 12	M_CONF12	R/W		x		Undefined
xxxxn195H	CAN message status register 12	M_STAT12	R		x		Undefined
xxxxn196H	CAN status set/cancel register 12	SC_STAT12	W			x	0000H
xxxxn1A0H	CAN message event pointer 130	M_EVT130	R/W		x		Undefined
xxxxn1A1H	CAN message event pointer 131	M_EVT131	R/W		x		Undefined
xxxxn1A2H	CAN message event pointer 132	M_EVT132	R/W		x		Undefined
xxxxn1A3H	CAN message event pointer 133	M_EVT133	R/W		x		Undefined
xxxxn1A4H	CAN message data length register 13	M_DLC13	R/W		x		Undefined
xxxxn1A5H	CAN message control register 13	M_CTRL13	R/W		x		Undefined
xxxxn1A6H	CAN message time stamp register 13	M_TIME13	R/W			x	Undefined
xxxxn1A8H	CAN message data register 130	M_DATA130	R/W		x		Undefined
xxxxn1A9H	CAN message data register 131	M_DATA131	R/W		x		Undefined
xxxxn1AAH	CAN message data register 132	M_DATA132	R/W		x		Undefined
xxxxn1ABH	CAN message data register 133	M_DATA133	R/W		x		Undefined
xxxxn1ACH	CAN message data register 134	M_DATA134	R/W		x		Undefined
xxxxn1ADH	CAN message data register 135	M_DATA135	R/W		x		Undefined
xxxxn1AEH	CAN message data register 136	M_DATA136	R/W		x		Undefined
xxxxn1AFH	CAN message data register 137	M_DATA137	R/W		x		Undefined
xxxxn1B0H	CAN message ID register L13	M_IDL13	R/W			x	Undefined
xxxxn1B2H	CAN message ID register H13	M_IDH13	R/W			x	Undefined
xxxxn1B4H	CAN message configuration register 13	M_CONF13	R/W		x		Undefined
xxxxn1B5H	CAN message status register 13	M_STAT13	R		x		Undefined
xxxxn1B6H	CAN status set/cancel register 13	SC_STAT13	W			x	0000H
xxxxn1C0H	CAN message event pointer 140	M_EVT140	R/W		x		Undefined
xxxxn1C1H	CAN message event pointer 141	M_EVT141	R/W		x		Undefined
xxxxn1C2H	CAN message event pointer 142	M_EVT142	R/W		x		Undefined
xxxxn1C3H	CAN message event pointer 143	M_EVT143	R/W		x		Undefined
xxxxn1C4H	CAN message data length register 14	M_DLC14	R/W		x		Undefined
xxxxn1C5H	CAN message control register 14	M_CTRL14	R/W		x		Undefined
xxxxn1C6H	CAN message time stamp register 14	M_TIME14	R/W			x	Undefined
xxxxn1C8H	CAN message data register 140	M_DATA140	R/W		x		Undefined
xxxxn1C9H	CAN message data register 141	M_DATA141	R/W		x		Undefined
xxxxn1CAH	CAN message data register 142	M_DATA142	R/W		x		Undefined
xxxxn1CBH	CAN message data register 143	M_DATA143	R/W		x		Undefined
xxxxn1CCH	CAN message data register 144	M_DATA144	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (8/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn1CDH	CAN message data register 145	M_DATA145	R/W		x		Undefined
xxxxn1CEH	CAN message data register 146	M_DATA146	R/W		x		Undefined
xxxxn1CFH	CAN message data register 147	M_DATA147	R/W		x		Undefined
xxxxn1D0H	CAN message ID register L14	M_IDL14	R/W			x	Undefined
xxxxn1D2H	CAN message ID register H14	M_IDH14	R/W			x	Undefined
xxxxn1D4H	CAN message configuration register 14	M_CONF14	R/W		x		Undefined
xxxxn1D5H	CAN message status register 14	M_STAT14	R		x		Undefined
xxxxn1D6H	CAN status set/cancel register 14	SC_STAT14	W			x	0000H
xxxxn1E0H	CAN message event pointer 150	M_EVT150	R/W		x		Undefined
xxxxn1E1H	CAN message event pointer 151	M_EVT151	R/W		x		Undefined
xxxxn1E2H	CAN message event pointer 152	M_EVT152	R/W		x		Undefined
xxxxn1E3H	CAN message event pointer 153	M_EVT153	R/W		x		Undefined
xxxxn1E4H	CAN message data length register 15	M_DLC15	R/W		x		Undefined
Xxxxn1E5H	CAN message control register 15	M_CTRL15	R/W		x		Undefined
Xxxxn1E6H	CAN message time stamp register 15	M_TIME15	R/W			x	Undefined
xxxxn1E8H	CAN message data register 150	M_DATA150	R/W		x		Undefined
xxxxn1E9H	CAN message data register 151	M_DATA151	R/W		x		Undefined
xxxxn1EAH	CAN message data register 152	M_DATA152	R/W		x		Undefined
xxxxn1EBH	CAN message data register 153	M_DATA153	R/W		x		Undefined
xxxxn1ECH	CAN message data register 154	M_DATA154	R/W		x		Undefined
xxxxn1EDH	CAN message data register 155	M_DATA155	R/W		x		Undefined
xxxxn1EEH	CAN message data register 156	M_DATA156	R/W		x		Undefined
xxxxn1EFH	CAN message data register 157	M_DATA157	R/W		x		Undefined
xxxxn1F0H	CAN message ID register L15	M_IDL15	R/W			x	Undefined
xxxxn1F2H	CAN message ID register H15	M_IDH15	R/W			x	Undefined
xxxxn1F4H	CAN message configuration register 15	M_CONF15	R/W		x		Undefined
xxxxn1F5H	CAN message status register 15	M_STAT15	R		x		Undefined
xxxxn1F6H	CAN status set/cancel register 15	SC_STAT15	W			x	0000H
xxxxn200H	CAN message event pointer 160	M_EVT160	R/W		x		Undefined
xxxxn201H	CAN message event pointer 161	M_EVT161	R/W		x		Undefined
xxxxn202H	CAN message event pointer 162	M_EVT162	R/W		x		Undefined
xxxxn203H	CAN message event pointer 163	M_EVT163	R/W		x		Undefined
xxxxn204H	CAN message data length register 16	M_DLC16	R/W		x		Undefined
xxxxn205H	CAN message control register 16	M_CTRL16	R/W		x		Undefined
xxxxn206H	CAN message time stamp register 16	M_TIME16	R/W			x	Undefined
xxxxn208H	CAN message data register 160	M_DATA160	R/W		x		Undefined
xxxxn209H	CAN message data register 161	M_DATA161	R/W		x		Undefined
xxxxn20AH	CAN message data register 162	M_DATA162	R/W		x		Undefined
xxxxn20BH	CAN message data register 163	M_DATA163	R/W		x		Undefined
xxxxn20CH	CAN message data register 164	M_DATA164	R/W		x		Undefined
xxxxn20DH	CAN message data register 165	M_DATA165	R/W		x		Undefined
xxxxn20EH	CAN message data register 166	M_DATA166	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (9/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn20FH	CAN message data register 167	M_DATA167	R/W		x		Undefined
xxxxn210H	CAN message ID register L16	M_IDL16	R/W			x	Undefined
xxxxn212H	CAN message ID register H16	M_IDH16	R/W			x	Undefined
xxxxn214H	CAN message configuration register 16	M_CONF16	R/W		x		Undefined
xxxxn215H	CAN message status register 16	M_STAT16	R		x		Undefined
xxxxn216H	CAN status set/cancel register 16	SC_STAT16	W			x	0000H
xxxxn220H	CAN message event pointer 170	M_EVT170	R/W		x		Undefined
xxxxn221H	CAN message event pointer 171	M_EVT171	R/W		x		Undefined
xxxxn222H	CAN message event pointer 172	M_EVT172	R/W		x		Undefined
xxxxn223H	CAN message event pointer 173	M_EVT173	R/W		x		Undefined
xxxxn224H	CAN message data length register 17	M_DLC17	R/W		x		Undefined
xxxxn225H	CAN message control register 17	M_CTRL17	R/W		x		Undefined
xxxxn226H	CAN message time stamp register 17	M_TIME17	R/W			x	Undefined
xxxxn228H	CAN message data register 170	M_DATA170	R/W		x		Undefined
xxxxn229H	CAN message data register 171	M_DATA171	R/W		x		Undefined
xxxxn22AH	CAN message data register 172	M_DATA172	R/W		x		Undefined
xxxxn22BH	CAN message data register 173	M_DATA173	R/W		x		Undefined
xxxxn22CH	CAN message data register 174	M_DATA174	R/W		x		Undefined
xxxxn22DH	CAN message data register 175	M_DATA175	R/W		x		Undefined
xxxxn22EH	CAN message data register 176	M_DATA176	R/W		x		Undefined
xxxxn22FH	CAN message data register 177	M_DATA177	R/W		x		Undefined
xxxxn230H	CAN message ID register L17	M_IDL17	R/W			x	Undefined
xxxxn232H	CAN message ID register H17	M_IDH17	R/W			x	Undefined
xxxxn234H	CAN message configuration register 17	M_CONF17	R/W		x		Undefined
xxxxn235H	CAN message status register 17	M_STAT17	R		x		Undefined
xxxxn236H	CAN status set/cancel register 17	SC_STAT17	W			x	0000H
xxxxn240H	CAN message event pointer 180	M_EVT180	R/W		x		Undefined
xxxxn241H	CAN message event pointer 181	M_EVT181	R/W		x		Undefined
xxxxn242H	CAN message event pointer 182	M_EVT182	R/W		x		Undefined
xxxxn243H	CAN message event pointer 183	M_EVT183	R/W		x		Undefined
xxxxn244H	CAN message data length register 18	M_DLC18	R/W		x		Undefined
xxxxn245H	CAN message control register 18	M_CTRL18	R/W		x		Undefined
xxxxn246H	CAN message time stamp register 18	M_TIME18	R/W			x	Undefined
xxxxn248H	CAN message data register 180	M_DATA180	R/W		x		Undefined
xxxxn249H	CAN message data register 181	M_DATA181	R/W		x		Undefined
xxxxn24AH	CAN message data register 182	M_DATA182	R/W		x		Undefined
xxxxn24BH	CAN message data register 183	M_DATA183	R/W		x		Undefined
xxxxn24CH	CAN message data register 184	M_DATA184	R/W		x		Undefined
xxxxn24DH	CAN message data register 185	M_DATA185	R/W		x		Undefined
xxxxn24EH	CAN message data register 186	M_DATA186	R/W		x		Undefined
xxxxn24FH	CAN message data register 187	M_DATA187	R/W		x		Undefined
xxxxn250H	CAN message ID register L18	M_IDL18	R/W			x	Undefined

Table 3-7: List of programmable peripheral I/O registers (10/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn252H	CAN message ID register H18	M_IDH18	R/W			x	Undefined
xxxxn254H	CAN message configuration register 18	M_CONF18	R/W		x		Undefined
xxxxn255H	CAN message status register 18	M_STAT18	R		x		Undefined
xxxxn256H	CAN status set/cancel register 18	SC_STAT18	W			x	0000H
xxxxn260H	CAN message event pointer 260	M_EVT260	R/W		x		Undefined
xxxxn261H	CAN message event pointer 261	M_EVT261	R/W		x		Undefined
xxxxn262H	CAN message event pointer 262	M_EVT262	R/W		x		Undefined
xxxxn263H	CAN message event pointer 263	M_EVT263	R/W		x		Undefined
xxxxn264H	CAN message data length register 19	M_DLC19	R/W		x		Undefined
xxxxn265H	CAN message control register 19	M_CTRL19	R/W		x		Undefined
xxxxn266H	CAN message time stamp register 19	M_TIME19	R/W			x	Undefined
xxxxn268H	CAN message data register 190	M_DATA190	R/W		x		Undefined
xxxxn269H	CAN message data register 191	M_DATA191	R/W		x		Undefined
xxxxn26AH	CAN message data register 192	M_DATA192	R/W		x		Undefined
xxxxn26BH	CAN message data register 193	M_DATA193	R/W		x		Undefined
xxxxn26CH	CAN message data register 194	M_DATA194	R/W		x		Undefined
xxxxn26DH	CAN message data register 195	M_DATA195	R/W		x		Undefined
xxxxn26EH	CAN message data register 196	M_DATA196	R/W		x		Undefined
xxxxn26FH	CAN message data register 197	M_DATA197	R/W		x		Undefined
xxxxn270H	CAN message ID register L19	M_IDL19	R/W			x	Undefined
xxxxn272H	CAN message ID register H19	M_IDH19	R/W			x	Undefined
xxxxn274H	CAN message configuration register 19	M_CONF19	R/W		x		Undefined
xxxxn275H	CAN message status register 19	M_STAT19	R		x		Undefined
xxxxn276H	CAN status set/cancel register 19	SC_STAT19	W			x	0000H
xxxxn280H	CAN message event pointer 200	M_EVT200	R/W		x		Undefined
xxxxn281H	CAN message event pointer 201	M_EVT201	R/W		x		Undefined
xxxxn282H	CAN message event pointer 202	M_EVT202	R/W		x		Undefined
xxxxn283H	CAN message event pointer 203	M_EVT203	R/W		x		Undefined
xxxxn284H	CAN message data length register 20	M_DLC20	R/W		x		Undefined
xxxxn285H	CAN message control register 20	M_CTRL20	R/W		x		Undefined
xxxxn286H	CAN message time stamp register 20	M_TIME20	R/W			x	Undefined
xxxxn288H	CAN message data register 200	M_DATA200	R/W		x		Undefined
xxxxn289H	CAN message data register 201	M_DATA201	R/W		x		Undefined
xxxxn28AH	CAN message data register 202	M_DATA202	R/W		x		Undefined
xxxxn28BH	CAN message data register 203	M_DATA203	R/W		x		Undefined
xxxxn28CH	CAN message data register 204	M_DATA204	R/W		x		Undefined
xxxxn28DH	CAN message data register 205	M_DATA205	R/W		x		Undefined
xxxxn28EH	CAN message data register 206	M_DATA206	R/W		x		Undefined
xxxxn28FH	CAN message data register 207	M_DATA207	R/W		x		Undefined
xxxxn290H	CAN message ID register L20	M_IDL20	R/W			x	Undefined
xxxxn292H	CAN message ID register H20	M_IDH20	R/W			x	Undefined
xxxxn294H	CAN message configuration register 20	M_CONF20	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (11/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn295H	CAN message status register 20	M_STAT20	R		x		Undefined
xxxxn296H	CAN status set/cancel register 20	SC_STAT20	W			x	0000H
xxxxn2A0H	CAN message event pointer 210	M_EVT210	R/W		x		Undefined
xxxxn2A1H	CAN message event pointer 211	M_EVT211	R/W		x		Undefined
xxxxn22AH	CAN message event pointer 212	M_EVT212	R/W		x		Undefined
xxxxn2A3H	CAN message event pointer 213	M_EVT213	R/W		x		Undefined
xxxxn2A4H	CAN message data length register 21	M_DLC21	R/W		x		Undefined
xxxxn2A5H	CAN message control register 21	M_CTRL21	R/W		x		Undefined
xxxxn2A6H	CAN message time stamp register 21	M_TIME21	R/W			x	Undefined
xxxxn2A8H	CAN message data register 210	M_DATA210	R/W		x		Undefined
xxxxn2A9H	CAN message data register 211	M_DATA211	R/W		x		Undefined
xxxxn2AAH	CAN message data register 212	M_DATA212	R/W		x		Undefined
xxxxn2ABH	CAN message data register 213	M_DATA213	R/W		x		Undefined
xxxxn2ACH	CAN message data register 214	M_DATA214	R/W		x		Undefined
xxxxn2ADH	CAN message data register 215	M_DATA215	R/W		x		Undefined
xxxxn2AEH	CAN message data register 216	M_DATA216	R/W		x		Undefined
xxxxn2AFH	CAN message data register 217	M_DATA217	R/W		x		Undefined
xxxxn2B0H	CAN message ID register L21	M_IDL21	R/W			x	Undefined
xxxxn2B2H	CAN message ID register H21	M_IDH21	R/W			x	Undefined
xxxxn2B4H	CAN message configuration register 21	M_CONF21	R/W		x		Undefined
xxxxn2B5H	CAN message status register 21	M_STAT21	R		x		Undefined
xxxxn2B6H	CAN status set/cancel register 21	SC_STAT21	W			x	0000H
xxxxn2C0H	CAN message event pointer 220	M_EVT220	R/W		x		Undefined
xxxxn2C1H	CAN message event pointer 221	M_EVT221	R/W		x		Undefined
xxxxn2C2H	CAN message event pointer 222	M_EVT222	R/W		x		Undefined
xxxxn2C3H	CAN message event pointer 223	M_EVT223	R/W		x		Undefined
xxxxn2C4H	CAN message data length register 22	M_DLC22	R/W		x		Undefined
xxxxn2C5H	CAN message control register 22	M_CTRL22	R/W		x		Undefined
xxxxn2C6H	CAN message time stamp register 22	M_TIME22	R/W			x	Undefined
xxxxn2C8H	CAN message data register 220	M_DATA220	R/W		x		Undefined
xxxxn2C9H	CAN message data register 221	M_DATA221	R/W		x		Undefined
xxxxn2CAH	CAN message data register 222	M_DATA222	R/W		x		Undefined
xxxxn2CBH	CAN message data register 223	M_DATA223	R/W		x		Undefined
xxxxn2CCH	CAN message data register 224	M_DATA224	R/W		x		Undefined
xxxxn2CDH	CAN message data register 225	M_DATA225	R/W		x		Undefined
xxxxn2CEH	CAN message data register 226	M_DATA226	R/W		x		Undefined
xxxxn2CFH	CAN message data register 227	M_DATA227	R/W		x		Undefined
xxxxn2D0H	CAN message ID register L22	M_IDL22	R/W			x	Undefined
xxxxn2D2H	CAN message ID register H22	M_IDH22	R/W			x	Undefined
xxxxn2D4H	CAN message configuration register 22	M_CONF22	R/W		x		Undefined
xxxxn2D5H	CAN message status register 22	M_STAT22	R		x		Undefined
xxxxn2D6H	CAN status set/cancel register 22	SC_STAT22	W			x	0000H

Table 3-7: List of programmable peripheral I/O registers (12/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn2E0H	CAN message event pointer 230	M_EVT230	R/W		x		Undefined
xxxxn2E1H	CAN message event pointer 231	M_EVT231	R/W		x		Undefined
xxxxn2EH	CAN message event pointer 232	M_EVT232	R/W		x		Undefined
xxxxn2E3H	CAN message event pointer 233	M_EVT233	R/W		x		Undefined
xxxxn2E4H	CAN message data length register 23	M_DLC23	R/W		x		Undefined
xxxxn2E5H	CAN message control register 23	M_CTRL23	R/W		x		Undefined
xxxxn2E6H	CAN message time stamp register 23	M_TIME23	R/W			x	Undefined
xxxxn2E8H	CAN message data register 230	M_DATA230	R/W		x		Undefined
xxxxn2E9H	CAN message data register 231	M_DATA231	R/W		x		Undefined
xxxxn2EAH	CAN message data register 232	M_DATA232	R/W		x		Undefined
xxxxn2EBH	CAN message data register 233	M_DATA233	R/W		x		Undefined
xxxxn2ECH	CAN message data register 234	M_DATA234	R/W		x		Undefined
xxxxn2EDH	CAN message data register 235	M_DATA235	R/W		x		Undefined
xxxxn2EEH	CAN message data register 236	M_DATA236	R/W		x		Undefined
xxxxn2EFH	CAN message data register 237	M_DATA237	R/W		x		Undefined
xxxxn2F0H	CAN message ID register L23	M_IDL23	R/W			x	Undefined
xxxxn2F2H	CAN message ID register H23	M_IDH23	R/W			x	Undefined
xxxxn2F4H	CAN message configuration register 23	M_CONF23	R/W		x		Undefined
xxxxn2F5H	CAN message status register 23	M_STAT23	R		x		Undefined
xxxxn2F6H	CAN status set/cancel register 23	SC_STAT23	W			x	0000H
xxxxn300H	CAN message event pointer 240	M_EVT240	R/W		x		Undefined
xxxxn301H	CAN message event pointer 241	M_EVT241	R/W		x		Undefined
xxxxn302H	CAN message event pointer 242	M_EVT242	R/W		x		Undefined
xxxxn303H	CAN message event pointer 243	M_EVT243	R/W		x		Undefined
xxxxn304H	CAN message data length register 24	M_DLC24	R/W		x		Undefined
xxxxn305H	CAN message control register 24	M_CTRL24	R/W		x		Undefined
xxxxn306H	CAN message time stamp register 24	M_TIME24	R/W			x	Undefined
xxxxn308H	CAN message data register 240	M_DATA240	R/W		x		Undefined
xxxxn309H	CAN message data register 241	M_DATA241	R/W		x		Undefined
xxxxn30AH	CAN message data register 242	M_DATA242	R/W		x		Undefined
xxxxn30BH	CAN message data register 243	M_DATA243	R/W		x		Undefined
xxxxn30CH	CAN message data register 244	M_DATA244	R/W		x		Undefined
xxxxn30DH	CAN message data register 245	M_DATA245	R/W		x		Undefined
xxxxn30EH	CAN message data register 246	M_DATA246	R/W		x		Undefined
xxxxn30FH	CAN message data register 247	M_DATA247	R/W		x		Undefined
xxxxn310H	CAN message ID register L24	M_IDL24	R/W			x	Undefined
xxxxn312H	CAN message ID register H24	M_IDH24	R/W			x	Undefined
xxxxn314H	CAN message configuration register 24	M_CONF24	R/W		x		Undefined
xxxxn315H	CAN message status register 24	M_STAT24	R		x		Undefined
xxxxn316H	CAN status set/cancel register 24	SC_STAT24	W			x	0000H
xxxxn320H	CAN message event pointer 250	M_EVT250	R/W		x		Undefined
xxxxn321H	CAN message event pointer 251	M_EVT251	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (13/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn322H	CAN message event pointer 252	M_EVT252	R/W		x		Undefined
xxxxn323H	CAN message event pointer 253	M_EVT253	R/W		x		Undefined
xxxxn324H	CAN message data length register 25	M_DLC25	R/W		x		Undefined
xxxxn325H	CAN message control register 25	M_CTRL25	R/W		x		Undefined
xxxxn326H	CAN message time stamp register 25	M_TIME25	R/W			x	Undefined
xxxxn328H	CAN message data register 250	M_DATA250	R/W		x		Undefined
xxxxn329H	CAN message data register 251	M_DATA251	R/W		x		Undefined
xxxxn32AH	CAN message data register 252	M_DATA252	R/W		x		Undefined
xxxxn32BH	CAN message data register 253	M_DATA253	R/W		x		Undefined
xxxxn32CH	CAN message data register 254	M_DATA254	R/W		x		Undefined
xxxxn32DH	CAN message data register 255	M_DATA255	R/W		x		Undefined
xxxxn32EH	CAN message data register 256	M_DATA256	R/W		x		Undefined
xxxxn32FH	CAN message data register 257	M_DATA257	R/W		x		Undefined
xxxxn330H	CAN message ID register L25	M_IDL25	R/W			x	Undefined
xxxxn332H	CAN message ID register H25	M_IDH25	R/W			x	Undefined
xxxxn334H	CAN message configuration register 25	M_CONF25	R/W		x		Undefined
xxxxn335H	CAN message status register 25	M_STAT25	R		x		Undefined
xxxxn336H	CAN status set/cancel register 25	SC_STAT25	W			x	0000H
xxxxn340H	CAN message event pointer 260	M_EVT260	R/W		x		Undefined
xxxxn341H	CAN message event pointer 261	M_EVT261	R/W		x		Undefined
xxxxn342H	CAN message event pointer 262	M_EVT262	R/W		x		Undefined
xxxxn343H	CAN message event pointer 263	M_EVT263	R/W		x		Undefined
xxxxn344H	CAN message data length register 26	M_DLC26	R/W		x		Undefined
xxxxn345H	CAN message control register 26	M_CTRL26	R/W		x		Undefined
xxxxn346H	CAN message time stamp register 26	M_TIME26	R/W			x	Undefined
xxxxn348H	CAN message data register 260	M_DATA260	R/W		x		Undefined
xxxxn349H	CAN message data register 261	M_DATA261	R/W		x		Undefined
xxxxn34AH	CAN message data register 262	M_DATA262	R/W		x		Undefined
xxxxn34BH	CAN message data register 263	M_DATA263	R/W		x		Undefined
xxxxn34CH	CAN message data register 264	M_DATA264	R/W		x		Undefined
xxxxn34DH	CAN message data register 265	M_DATA265	R/W		x		Undefined
xxxxn34EH	CAN message data register 266	M_DATA266	R/W		x		Undefined
xxxxn34FH	CAN message data register 267	M_DATA267	R/W		x		Undefined
xxxxn350H	CAN message ID register L26	M_IDL26	R/W			x	Undefined
xxxxn352H	CAN message ID register H26	M_IDH26	R/W			x	Undefined
xxxxn354H	CAN message configuration register 26	M_CONF26	R/W		x		Undefined
xxxxn355H	CAN message status register 26	M_STAT26	R		x		Undefined
xxxxn356H	CAN status set/cancel register 26	SC_STAT26	W			x	0000H
xxxxn360H	CAN message event pointer 270	M_EVT270	R/W		x		Undefined
xxxxn361H	CAN message event pointer 271	M_EVT271	R/W		x		Undefined
xxxxn362H	CAN message event pointer 272	M_EVT272	R/W		x		Undefined
xxxxn363H	CAN message event pointer 273	M_EVT273	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (14/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn364H	CAN message data length register 27	M_DLC27	R/W		x		Undefined
xxxxn365H	CAN message control register 27	M_CTRL27	R/W		x		Undefined
xxxxn366H	CAN message time stamp register 27	M_TIME27	R/W			x	Undefined
xxxxn368H	CAN message data register 270	M_DATA270	R/W		x		Undefined
xxxxn369H	CAN message data register 271	M_DATA271	R/W		x		Undefined
xxxxn36AH	CAN message data register 272	M_DATA272	R/W		x		Undefined
xxxxn36BH	CAN message data register 273	M_DATA273	R/W		x		Undefined
xxxxn36CH	CAN message data register 274	M_DATA274	R/W		x		Undefined
xxxxn36DH	CAN message data register 275	M_DATA275	R/W		x		Undefined
xxxxn36EH	CAN message data register 276	M_DATA276	R/W		x		Undefined
xxxxn36FH	CAN message data register 277	M_DATA277	R/W		x		Undefined
xxxxn370H	CAN message ID register L27	M_IDL27	R/W			x	Undefined
xxxxn372H	CAN message ID register H27	M_IDH27	R/W			x	Undefined
xxxxn374H	CAN message configuration register 27	M_CONF27	R/W		x		Undefined
xxxxn375H	CAN message status register 27	M_STAT27	R		x		Undefined
xxxxn376H	CAN status set/cancel register 27	SC_STAT27	W			x	0000H
xxxxn380H	CAN message event pointer 280	M_EVT280	R/W		x		Undefined
xxxxn381H	CAN message event pointer 281	M_EVT281	R/W		x		Undefined
xxxxn382H	CAN message event pointer 282	M_EVT282	R/W		x		Undefined
xxxxn383H	CAN message event pointer 283	M_EVT283	R/W		x		Undefined
xxxxn384H	CAN message data length register 28	M_DLC28	R/W		x		Undefined
xxxxn385H	CAN message control register 28	M_CTRL28	R/W		x		Undefined
xxxxn386H	CAN message time stamp register 28	M_TIME28	R/W			x	Undefined
xxxxn388H	CAN message data register 280	M_DATA280	R/W		x		Undefined
xxxxn389H	CAN message data register 281	M_DATA281	R/W		x		Undefined
xxxxn38AH	CAN message data register 282	M_DATA282	R/W		x		Undefined
xxxxn38BH	CAN message data register 283	M_DATA283	R/W		x		Undefined
xxxxn38CH	CAN message data register 284	M_DATA284	R/W		x		Undefined
xxxxn38DH	CAN message data register 285	M_DATA285	R/W		x		Undefined
xxxxn38EH	CAN message data register 286	M_DATA286	R/W		x		Undefined
xxxxn38FH	CAN message data register 287	M_DATA287	R/W		x		Undefined
xxxxn390H	CAN message ID register L28	M_IDL28	R/W			x	Undefined
xxxxn392H	CAN message ID register H28	M_IDH28	R/W			x	Undefined
xxxxn394H	CAN message configuration register 28	M_CONF28	R/W		x		Undefined
xxxxn395H	CAN message status register 28	M_STAT28	R		x		Undefined
xxxxn396H	CAN status set/cancel register 28	SC_STAT28	W			x	0000H
xxxxn3A0H	CAN message event pointer 290	M_EVT290	R/W		x		Undefined
xxxxn3A1H	CAN message event pointer 291	M_EVT291	R/W		x		Undefined
xxxxn3A2H	CAN message event pointer 292	M_EVT292	R/W		x		Undefined
xxxxn3A3H	CAN message event pointer 293	M_EVT293	R/W		x		Undefined
xxxxn3A4H	CAN message data length register 29	M_DLC29	R/W		x		Undefined
xxxxn3A5H	CAN message control register 29	M_CTRL29	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (15/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn3A6H	CAN message time stamp register 29	M_TIME29	R/W			x	Undefined
xxxxn3A8H	CAN message data register 290	M_DATA290	R/W		x		Undefined
xxxxn3A9H	CAN message data register 291	M_DATA291	R/W		x		Undefined
xxxxn3AAH	CAN message data register 292	M_DATA292	R/W		x		Undefined
xxxxn3ABH	CAN message data register 293	M_DATA293	R/W		x		Undefined
xxxxn3ACH	CAN message data register 294	M_DATA294	R/W		x		Undefined
xxxxn3ADH	CAN message data register 295	M_DATA295	R/W		x		Undefined
xxxxn3AEH	CAN message data register 296	M_DATA296	R/W		x		Undefined
xxxxn3AFH	CAN message data register 297	M_DATA297	R/W		x		Undefined
xxxxn3B0H	CAN message ID register L29	M_IDL29	R/W			x	Undefined
xxxxn3B2H	CAN message ID register H29	M_IDH29	R/W			x	Undefined
xxxxn3B4H	CAN message configuration register 29	M_CONF29	R/W		x		Undefined
xxxxn3B5H	CAN message status register 29	M_STAT29	R		x		Undefined
xxxxn3B6H	CAN status set/cancel register 29	SC_STAT29	W			x	0000H
xxxxn3C0H	CAN message event pointer 300	M_EVT300	R/W		x		Undefined
xxxxn3C1H	CAN message event pointer 301	M_EVT301	R/W		x		Undefined
xxxxn3C2H	CAN message event pointer 302	M_EVT302	R/W		x		Undefined
xxxxn3C3H	CAN message event pointer 303	M_EVT303	R/W		x		Undefined
xxxxn3C4H	CAN message data length register 30	M_DLC30	R/W		x		Undefined
xxxxn3C5H	CAN message control register 30	M_CTRL30	R/W		x		Undefined
xxxxn3C6H	CAN message time stamp register 30	M_TIME30	R/W			x	Undefined
xxxxn3C8H	CAN message data register 300	M_DATA300	R/W		x		Undefined
xxxxn3C9H	CAN message data register 301	M_DATA301	R/W		x		Undefined
xxxxn3CAH	CAN message data register 302	M_DATA302	R/W		x		Undefined
xxxxn3CBH	CAN message data register 303	M_DATA303	R/W		x		Undefined
xxxxn3CCH	CAN message data register 304	M_DATA304	R/W		x		Undefined
xxxxn3CDH	CAN message data register 305	M_DATA305	R/W		x		Undefined
xxxxn3CEH	CAN message data register 306	M_DATA306	R/W		x		Undefined
xxxxn3CFH	CAN message data register 307	M_DATA307	R/W		x		Undefined
xxxxn3D0H	CAN message ID register L30	M_IDL30	R/W			x	Undefined
xxxxn3D2H	CAN message ID register H30	M_IDH30	R/W			x	Undefined
xxxxn3D4H	CAN message configuration register 30	M_CONF30	R/W		x		Undefined
xxxxn3D5H	CAN message status register 30	M_STAT30	R		x		Undefined
xxxxn3D6H	CAN status set/cancel register 30	SC_STAT30	W			x	0000H
xxxxn3E0H	CAN message event pointer 310	M_EVT310	R/W		x		Undefined
xxxxn3E1H	CAN message event pointer 311	M_EVT311	R/W		x		Undefined
xxxxn3E2H	CAN message event pointer 312	M_EVT312	R/W		x		Undefined
xxxxn3E3H	CAN message event pointer 313	M_EVT313	R/W		x		Undefined
xxxxn3E4H	CAN message data length register 31	M_DLC31	R/W		x		Undefined
xxxxn3E5H	CAN message control register 31	M_CTRL31	R/W		x		Undefined
xxxxn3E6H	CAN message time stamp register 31	M_TIME31	R/W			x	Undefined
xxxxn3E8H	CAN message data register 310	M_DATA310	R/W		x		Undefined

Table 3-7: List of programmable peripheral I/O registers (16/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn3E9H	CAN message data register 311	M_DATA311	R/W		x		Undefined
xxxxn3EAH	CAN message data register 312	M_DATA312	R/W		x		Undefined
xxxxn3EBH	CAN message data register 313	M_DATA313	R/W		x		Undefined
xxxxn3ECH	CAN message data register 314	M_DATA314	R/W		x		Undefined
xxxxn3EDH	CAN message data register 315	M_DATA315	R/W		x		Undefined
xxxxn3EEH	CAN message data register 316	M_DATA316	R/W		x		Undefined
xxxxn3EFH	CAN message data register 317	M_DATA317	R/W		x		Undefined
xxxxn3FCH	CAN message ID register L31	M_IDL31	R/W			x	Undefined
xxxxn3F2H	CAN message ID register H31	M_IDH31	R/W			x	Undefined
xxxxn3F4H	CAN message configuration register 31	M_CONF31	R/W		x		Undefined
xxxxn3F5H	CAN message status register 31	M_STAT31	R		x		Undefined
xxxxn3F7H	CAN status set/cancel register 31	SC_STAT31	W			x	0000H
xxxxn 1000H	CAN stop register [Note 1]	CSTOP	R/(W)		R	x	0000H
xxxxn 1004H	CAN interrupt pending register	CCINTP	R		x	x	0000H
xxxxn 1010H	CAN global status register [Note 1]	CGST	R/(W)		R	x	0100H
xxxxn1012H	CAN global interrupt enable register [Note 1]	CGIE	R/(W)		R	x	0A00H
xxxxn1014H	CAN main clock select register	CGCS	R/W		x	x	7F05H
xxxxn1016H	CAN timer event enable register	CGTEN	R/W		x	x	0000H
xxxxn1018H	CAN time stop count register	CGTSC	R		x	x	0000H
xxxxn101AH	CAN message find start register	CGMSS	W			x	0000H
xxxxn101AH	CAN message find result register	CGMSR	R			x	0000H
xxxxn101CH	CAN test bus register	CTBR	R/W		x	x	0000H
xxxxn101DH	CAN Macro Version Register	CGREV	R		x	x	Revision
xxxxn101EH	CAN Macro Revision Register	CGVER	R		x	x	Version
xxxxn101FH			R		x	x	
xxxxn1020H	CAN global interrupt pending register [Note 1]	CGINTP	R/(W)		R	x	00H
xxxxn1022H	CAN local interrupt pending register 1	C1INTP	R/W		x	x	00H
xxxxn1024H	CAN local interrupt pending register 2	C2INTP	R/W		x	x	00H
xxxxn1026H	CAN local interrupt pending register 3	C3INTP	R/W		x	x	00H
xxxxn1028H	CAN local interrupt pending register 4	C4INTP	R/W		x	x	00H
xxxxn1040H	CAN1 address mask register L0	C1MASKL0	R/W		x	x	Undefined
xxxxn1042H	CAN1 address mask register H0	C1MASKH0	R/W		x	x	Undefined
xxxxn1044H	CAN1 address mask register L1	C1MASKL1	R/W		x	x	Undefined
xxxxn1046H	CAN1 address mask register H1	C1MASKH1	R/W		x	x	Undefined
xxxxn1048H	CAN1 address mask register L2	C1MASKL2	R/W		x	x	Undefined
xxxxn104AH	CAN1 address mask register H2	C1MASKH2	R/W		x	x	Undefined
xxxxn104CH	CAN1 address mask register L3	C1MASKL3	R/W		x	x	Undefined
xxxxn104EH	CAN1 address mask register H3	C1MASKH3	R/W		x	x	Undefined
xxxxn1050H	CAN1 control register [Note 1]	C1CTRL	R/(W)		R	x	0101H
xxxxn1052H	CAN1 definition register [Note 1]	C1DEF	R/(W)		R	x	0000H
xxxxn1054H	CAN1 information register	C1LAST	R		x	x	00FFH

Table 3-7: List of programmable peripheral I/O registers (17/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn1056H	CAN1 error counter register	C1ERC	R		x	x	0000H
xxxxn1058H	CAN1 interrupt enable register Note 1	C1IE	R/(W)		R	x	0000H
xxxxn105AH	CAN1 bus active register	C1BA	R		x	x	00FFH
xxxxn105CH	CAN1 bit rate prescaler register	C1BRP	R/W		x	x	0000H
xxxxn105DH	CAN1 bus diagnostic information register	C1DINF	R		x	x	0000H
xxxxn105EH	CAN1 synchronization control register	C1SYNC	R/W		x	x	0218H
xxxxn1080H	CAN2 address mask register L0	C2MASKL0	R/W		x	x	Undefined
xxxxn1082H	CAN2 address mask register H0	C2MASKH0	R/W		x	x	Undefined
xxxxn1084H	CAN2 address mask register L1	C2MASKL1	R/W		x	x	Undefined
xxxxn1086H	CAN2 address mask register H1	C2MASKH1	R/W		x	x	Undefined
xxxxn1088H	CAN2 address mask register L2	C2MASKL2	R/W		x	x	Undefined
xxxxn108AH	CAN2 address mask register H2	C2MASKH2	R/W		x	x	Undefined
xxxxn108CH	CAN2 address mask register L3	C2MASKL3	R/W		x	x	Undefined
xxxxn108EH	CAN2 address mask register H3	C2MASKH3	R/W		x	x	Undefined
xxxxn1090H	CAN2 control register Note 1	C2CTRL	R/(W)		R	x	0101H
xxxxn1092H	CAN2 definition register Note 1	C2DEF	R/(W)		R	x	0000H
xxxxn1094H	CAN2 information register	C2LAST	R		x	x	00FFH
xxxxn1096H	CAN2 error counter register	C2ERC	R		x	x	0000H
xxxxn1098H	CAN2 interrupt enable register Note 1	C2IE	R/(W)		R	x	0000H
xxxxn109AH	CAN2 bus active register	C2BA	R		x	x	00FFH
xxxxn109CH	CAN2 bit rate prescaler register	C2BRP	R/W		x	x	0000H
xxxxn109DH	CAN2 bus diagnostic information register	C2DINF	R		x	x	0000H
xxxxn109EH	CAN2 synchronization control register	C2SYNC	R/W		x	x	0218H
xxxxn10C0H	CAN3 address mask register L0 Note 2	C3MASKL0	R/W		x	x	Undefined
xxxxn10C2H	CAN3 address mask register H0 Note 2	C3MASKH0	R/W		x	x	Undefined
xxxxn10C4H	CAN3 address mask register L1 Note 2	C3MASKL1	R/W		x	x	Undefined
xxxxn10C6H	CAN3 address mask register H1 Note 2	C3MASKH1	R/W		x	x	Undefined
xxxxn10C8H	CAN3 address mask register L2 Note 2	C3MASKL2	R/W		x	x	Undefined
xxxxn10CAH	CAN3 address mask register H2 Note 2	C3MASKH2	R/W		x	x	Undefined
xxxxn10CCH	CAN3 address mask register L3 Note 2	C3MASKL3	R/W		x	x	Undefined
xxxxn10CEH	CAN3 address mask register H3 Note 2	C3MASKH3	R/W		x	x	Undefined
xxxxn10D0H	CAN3 control register Note 1, 2	C3CTRL	R/(W)		R	x	0101H
xxxxn10D2H	CAN3 definition register Note 1, 2	C3DEF	R/(W)		R	x	0000H
xxxxn10D4H	CAN3 information register Note 2	C3LAST	R		x	x	00FFH
xxxxn10D6H	CAN3 error counter register Note 2	C3ERC	R		x	x	0000H
xxxxn10D8H	CAN3 interrupt enable register Note 1, 2	C3IE	R/(W)		R	x	0000H
xxxxn10DAH	CAN3 bus active register Note 2	C3BA	R		x	x	00FFH
xxxxn10DCH	CAN3 bit rate prescaler register Note 2	C3BRP	R/W		x	x	0000H
xxxxn10DDH	CAN3 bus diagnostic information register Note 2	C3DINF	R		x	x	0000H

Table 3-7: List of programmable peripheral I/O registers (18/18)

Address	Function Register Name	Symbol	R/W	Bit Units for Manipulation			Initial Value
				1-bit	8-bit	16-bit	
xxxxn10DEH	CAN3 synchronization control register **Note 2**	C3SYNC	R/W		x	x	0218H
xxxxn1100H	CAN4 address mask register L0 **Note 2**	C4MASKL0	R/W		x	x	Undefined
xxxxn11C2H	CAN4 address mask register H0 **Note 2**	C4MASKH0	R/W		x	x	Undefined
xxxxn11C4H	CAN4 address mask register L1 **Note 2**	C4MASKL1	R/W		x	x	Undefined
xxxxn11C6H	CAN4 address mask register H1 **Note 2**	C4MASKH1	R/W		x	x	Undefined
xxxxn11C8H	CAN4 address mask register L2 **Note 2**	C4MASKL2	R/W		x	x	Undefined
xxxxn11CAH	CAN4 address mask register H2 **Note 2**	C4MASKH2	R/W		x	x	Undefined
xxxxn11CCH	CAN4 address mask register L3 **Note 2**	C4MASKL3	R/W		x	x	Undefined
xxxxn11CEH	CAN4 address mask register H3 **Note 2**	C4MASKH3	R/W		x	x	Undefined
xxxxn11D0H	CAN4 control register **Note 1, 2**	C4CTRL	R/(W)		R	x	0101H
xxxxn11D2H	CAN4 definition register **Note 1, 2**	C4DEF	R/(W)		R	x	0000H
xxxxn11D4H	CAN4 information register **Note 2**	C4LAST	R		x	x	00FFH
xxxxn11D6H	CAN4 error counter register **Note 2**	C4ERC	R		x	x	0000H
xxxxn11D8H	CAN4 interrupt enable register **Note 1, 2**	C4IE	R/(W)		R	x	0000H
xxxxn11DAH	CAN4 bus active register **Note 2**	C4BA	R		x	x	00FFH
xxxxn11DCH	CAN4 bit rate prescaler register **Note 2**	C4BRP	R/W		x	x	0000H
xxxxn11DDH	CAN4 bus diagnostic information register **Note 2**	C4DINF	R		x	x	0000H
xxxxn11DEH	CAN4 synchronization control register **Note 2**	C4SYNC	R/W		x	x	0218H

Notes: 1. This registers can be accessed in 8-bit or 16-bit units during read and can be accessed in 16-bit units only during write

2. CAN2 and CAN3 are available only in µPD703129 (16 Kbytes RAM)

Remark: n = xx00b

3.6 Specific Registers

Specific registers are registers that are protected from being written with illegal data due to erroneous program execution, etc. The write access of these specific registers is executed in a specific sequence, and if abnormal store operations occur, it is notified by the peripheral status register (PHS).
The V850E/CA2 Jupiter has five specific registers, the clock control register (CKC), the watchdog timer clock control register (WCC), the processor clock control register (PCC) and the power save control register (PSC).
For details of the CKC, WCC and PCC register please refer to the chapter **9.3.1 "Clock Control Register (CKC)" on page 241**.
For details of the PSC register please refer to the chapter **9.4.3 "Power Saving Mode Functions" on page 254**.

The access sequence to the specified registers is shown below.
The following sequence shows the data setting of the specific registers.

- Store instruction (ST/SST instruction)
- Bit operation instruction (SET1/CLR1/NOT1 instruction)

Please see the following example for initialization of a power save mode. The PSC register is a specific register and therefore the PRCMD register has to be written first. The following 5 NOPs are necessary for waken from the STOP mode.

Example

```
<1>  MOV   0x02,r10
<2>  ST.B  r10,PRCMD[r0]
<3>  ST.B  r10,PSC[r0]
<4>  NOP   dummy instruction (5 times NOP required)
```

No special sequence is required when reading the specific registers.

Remarks:

1. A store instruction to a command register will not be received with an interrupt.
   This presupposes that this is done with the continuous store instructions in <1> and <2> above in the program. If another instruction is placed between <1> and <2>, when an interrupt is received by that instruction, the above sequence may not be established, and cause a malfunction, so caution is necessary.
2. The data written in the PRCMD register is dummy data, but use the same general purpose register for writing to the PRCMD register (<2> in the example above) as was used in setting data in the specified register (<3> in the example above). Addressing is the same in the case where a general purpose register is used.
3. In a store instruction to the PSC register for setting it in the software STOP mode or IDLE mode, it is necessary to insert 1 or more NOP instructions just after. When clearing each power save mode by interrupt, or when resetting after executing interrupt processing, start executing from the next instruction without executing 1 instruction just after the store instruction.

3.6.1 Command Register (PRCMD)

This command register (PRCMD) is to protect the registers that may have a significant influence on the application system (PSC, PSM) from an inadvertent write access, so that the system does not stop in case of a program hang-up.
This register can only be written in 8-bit units (undefined data is used when this register is read).
Only the first write access to a specific on-chip register (hereafter referred to as a “specific register”) after data has been written to the PRCMD register is valid.
In this way, the value of the specific register can be rewritten only in a specified sequence, and an illegal write access is inhibited.

Figure 3-17: Command Register (PRCMD) Format

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PRCMD	REG7	REG6	REG5	REG4	REG3	REG2	REG1	REG0	FFFFF1FCH	R/W	xxH

REG7 to REG0: registration code (**any** 8-bit data)

Caution: The register must be written with store instruction execution by CPU. DMA transfer is prohibited.

3.6.2 Peripheral Command Register (PHCMD)

This command register (PHCMD) is to protect the registers that may have a significant influence on the application system (CKC, WCC, PCC) from an inadvertent write access, so that the system does not stop in case of a program hang-up.
This register can be only written in 8-bit units (undefined data is used when this register is read).

Only the first write access to a specific on-chip register (hereafter referred to as a “specific register”) after data has been written to the PHCMD register is valid.
In this way, the value of the specific register can be rewritten only in a specified sequence, and an illegal write access is inhibited.

Figure 3-18: Peripheral Command Register (PHCMD) Format

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PHCMD	REG7	REG6	REG5	REG4	REG3	REG2	REG1	REG0	FFFFF800H	R/W	xxH

REG7-0: registration code (**any** 8-bit data)

Caution: The register must be written with store instruction execution by CPU. DMA transfer is prohibited.

If an illegal store operation takes place, it can be checked by the PRERR flag of the peripheral status register (PHS).

Caution: Write to this register by DMA transfer is prohibited!

3.6.3 Peripheral Status Register (PHS)

The flag PRERR in the peripheral status register PHS indicates protection error occurrence.
This register can be read/written in 8-bit units or bit-wise.

Figure 3-19: Peripheral Status Register (PHS) Format

	7	6	5	4	3	2	1	0	Address	R/W	At Reset
PHS	0	0	0	0	0	0	0	PRERR	FFFFF802H	R/W	00H

Protection error detection:
If an incorrect write operation in a sequence without accessing the command register is performed to a protected internal register, the register is not written to, causing a protection error.
Writing "0" to the PRERR flag after the value is checked clears the error.

Operation conditions of PRERR flag:

Set condition:
<1> If the most recent store instruction for peripheral I/O register operation is not an operation to write the PHCMD register and if data is written to the specific register.
<2> If the first store instruction operation after data has been written to the PHCMD register is to memory or peripheral I/Os other than those of a specified register.

Reset condition:
<1> When "0" is written to the PRERR flag of the PHS register.
<2> On system reset.

3.6.4 Internal peripheral function wait control register (VSWC)

This register inserts wait states to the internal access of peripheral SFRs.
This register can be read or written in 1-bit and 8-bit units.

Figure 3-20: Internal peripheral function wait control register (VSWC) Format

	7	6	5	4	3	2	1	0	Address	R/W	Reset Value
VSWC	0	SUWL2	SUWL1	SUWL0	0	VSWL2	VSWL1	VSWL0	FFFFF06EH	R/W	77H
	0	1	1	1	0	1	1	1			

Bit Name	Description			
SUWL2, SUWL1, SUWL0	Setup wait for internal peripheral bus length			
	SUWL2	SUWL1	SUWL0	Number of data wait states (n = 7 - 0)
	0	0	0	0
	0	0	1	1 system clock
	0	1	0	2 system clock
	0	1	1	3 system clock
	1	0	0	4 system clock
	1	0	1	5 system clock
	1	1	0	6 system clock
	1	1	1	7 system clock (default)
VSWL2, VSWL1, VSWL0	Internal peripheral bus wait length			
	VSWL2	VSWL1	VSWL0	Number of data wait states (n = 7 - 0)
	0	0	0	0
	0	0	1	1 system clock
	0	1	0	2 system clock
	0	1	1	3 system clock
	1	0	0	4 system clock
	1	0	1	5 system clock
	1	1	0	6 system clock
	1	1	1	7 system clock (default)

Caution: With respect to the specified operation frequency the following register settings for VSWC are recommended.

Table 3-8: The Values of VSWC Register depending on System Clock

System Clock	Setup Wait	Strobe Wait	VSWC
4.0 MHz < f_{CPU} < 16.6 MHz	0	0	00H
16.6 MHz < f_{CPU} < 25.0 MHz	0	1	01H
25.0 MHz < f_{CPU} < 32.0 MHz	1	1	11H

Chapter 4 Bus Control Function

The V850E/CA2 Jupiter is provided with an external bus interface function by which external memories such as ROM and RAM, and I/O can be connected.

4.1 Features

- 16-bit/8-bit data bus sizing function
- 8 chip areas select function
  - 3 chip area select signals externally available ($\overline{\text{CS0}}$,$\overline{\text{CS3}}$ and $\overline{\text{CS4}}$)
- Wait function
  - Programmable wait function, capable of inserting up to 7 wait states for each memory block
  - External wait function through $\overline{\text{WAIT}}$ pin
- Idle state insertion function
- External device connection can be enabled via bus control/port alternate function pins

4.2 Bus Control Pins

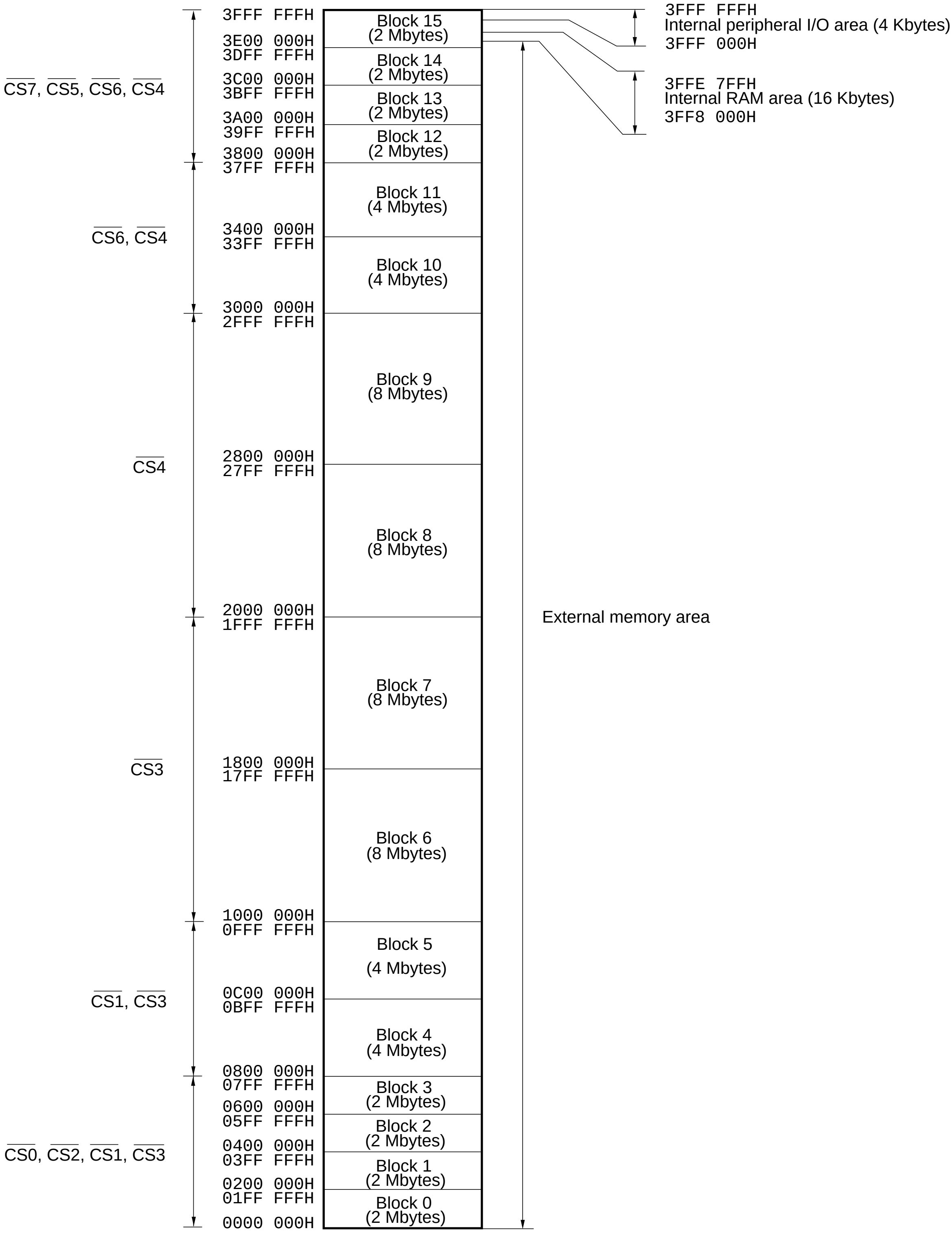
The following pins are used for connecting to external devices.

Bus Control Pin (Function when in Control Mode)	Function when in Port Mode	Register for Port/Control Mode Switching
Address data Data bus (D0 to D15)	-	-
Address bus (A0 to A15)	-	-
Address bus (A16 to A23)	PAH0 to PAH7 (Port AH)	PMCAH
Chip select ($\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$)	PCS0, PCS3 and PCS4 (Port CS)	PMCCS
Read/write control ($\overline{\text{LWR}}$, $\overline{\text{UWR}}$, $\overline{\text{RD}}$)	PCT0, PCT1, PCT4 (Port CT)	PMCCT
Internal system clock ($\overline{\text{WAIT}}$)	PCM0 (Port CM)	PMCCM

4.3 Memory Block Function

The 64 MB memory space is divided into memory blocks of 2 MB, 4 MB, and 8 MB units.

Figure 4-1: Memory Block Function



4.3.1 Chip Select Control Function

The 64 MB memory area can be divided into 2 MB, 4 MB and 8 MB memory blocks by the chip area selection control registers 0 and 1 (CSC0, CSC1) to control the chip select signals.
The memory area can be effectively used by dividing the memory area into memory blocks using the chip select control function. The priority order is described below.

(1) Chip area selection control registers 0, 1(CSC0, CSC1)

These registers can be read/written in 16-bit units. Valid by setting each bit (to 1). If different chip area select signals are set to the same block, the priority order is controlled as follows.

CSC0: Peripheral I/O area > $\overline{\text{CS0}}$ > $\overline{\text{CS2}}$ > $\overline{\text{CS1}}$ > $\overline{\text{CS3}}$ **Note**
CSC1: Peripheral I/O area > $\overline{\text{CS7}}$ > $\overline{\text{CS5}}$ > $\overline{\text{CS6}}$ > $\overline{\text{CS4}}$ **Note**

Notes: 1. Not all the chip area select signals are externally available on output pins. Even so, enabling chip area select signals other than $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ or $\overline{\text{CS4}}$, the setting for the corresponding memory blocks will be effective too, regardless of an external chip select output pin.

2. After reset Jupiter fetches the boot vector from CS0 area.

Figure 4-2: Chip Area Select Control Registers 0, 1 (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CSC0	CS33	CS32	CS31	CS30	CS23	CS22	CS21	CS20	CS13	CS12	CS11	CS10	CS03	CS02	CS01	CS00	3FFFFF060H	2C11H
	$\overline{\text{CS3}}$				$\overline{\text{CS2}}$				$\overline{\text{CS1}}$				$\overline{\text{CS0}}$					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CSC1	CS43	CS42	CS41	CS40	CS53	CS52	CS51	CS50	CS63	CS62	CS61	CS60	CS73	CS72	CS71	CS70	3FFFFF062H	2C11H
	$\overline{\text{CS4}}$				$\overline{\text{CS5}}$				$\overline{\text{CS6}}$				$\overline{\text{CS7}}$					

Figure 4-2: Chip Area Select Control Registers 0, 1 (2/2)

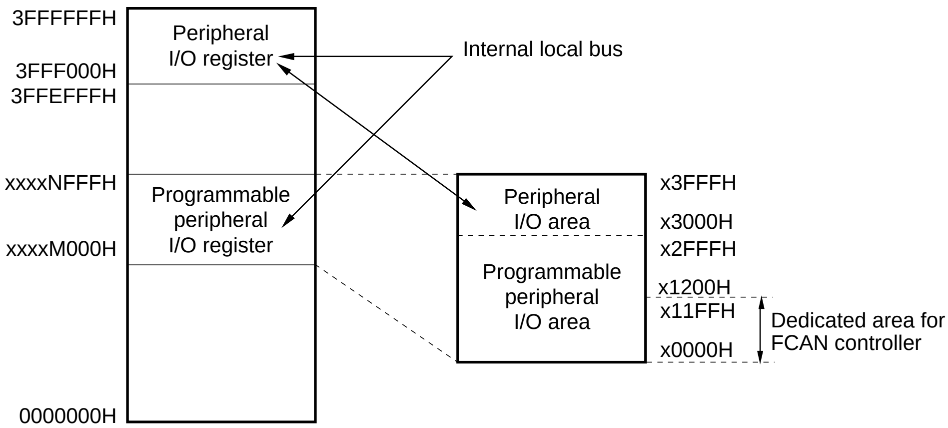
Bit Position	Bit Name	Function
15 to 0	CSn0 to CSn3 (n = 0 to 7)	Chip Select Enables chip select.

CSnm	CS Operation
CS00	$\overline{CS0}$ active during block 0 access
CS01	$\overline{CS0}$ active during block 1 access.
CS02	$\overline{CS0}$ active during block 2 access.
CS03	$\overline{CS0}$ active during block 3 access.
CS10	$\overline{CS1}$ active during block 0 or 1 access.
CS11	$\overline{CS1}$ active during block 2 or 3 access.
CS12	$\overline{CS1}$ active during block 4 access.
CS13	$\overline{CS1}$ active during block 5 access.
CS20	$\overline{CS2}$ active during block 0 access.
CS21	$\overline{CS2}$ active during block 1 access.
CS22	$\overline{CS2}$ active during block 2 access.
CS23	$\overline{CS2}$ active during block 3 access.
CS30	$\overline{CS3}$ active during block 0, 1, 2, or 3 access.
CS31	$\overline{CS3}$ active during block 4 or 5 access.
CS32	$\overline{CS3}$ active during block 6 access.
CS33	$\overline{CS3}$ active during block 7 access.
CS40	$\overline{CS4}$ active during block 12, 13, 14, or 15 access.
CS41	$\overline{CS4}$ active during block 10 or 11 access.
CS42	$\overline{CS4}$ active during block 9 access.
CS43	$\overline{CS4}$ active during block 8 access.
CS50	$\overline{CS5}$ active during block 15 access.
CS51	$\overline{CS5}$ active during block 14 access.
CS52	$\overline{CS5}$ active during block 13 access.
CS53	$\overline{CS5}$ active during block 12 access.
CS60	$\overline{CS6}$ active during block 14 or 15 access.
CS61	$\overline{CS6}$ active during block 12 or 13 access.
CS62	CS6 active during block 11 access.
CS63	$\overline{CS6}$ active during block 10 access.
CS70	$\overline{CS7}$ active during block 15 access.
CS71	$\overline{CS7}$ active during block 14 access.
CS72	$\overline{CS7}$ active during block 13 access.
CS73	$\overline{CS7}$ active during block 12 access.

4.4 Programmable peripheral I/O registers

In the V850E/CA2, the 16 KB area of x0000H to x3FFFH is provided as a programmable peripheral I/O area. In this area, the area between x0000H and x11FFH is used exclusively for the FCAN controller. The internal bus of the V850E/CA2 becomes active when the peripheral I/O register area (FFFF000H to FFFFFFFH) or the programmable peripheral I/O register area (xxxxm000H to xxxxnFFFH) is accessed (m = xx00B, n = xx11B). Note that when data is written to the peripheral I/O register area, the written contents is reflected on the peripheral I/O register since peripheral I/O register area is allocated to the last 4 KB of the programmable peripheral I/O register area.

Figure 4-3: Programmable Peripheral I/O Register (Outline)



Cautions:

1. **The programmable peripheral area must not be located above the address x1FFFFFFH.**
2. **Once the address of the programmable peripheral area is se, it cannot be changed.**
3. **If the programmable peripheral I/O area overlaps the following areas, the programmable peripheral I/O area becomes ineffective.**
   - **Peripheral I/O area**
   - **ROM area**
   - **RAM area**
4. **Programmable peripheral I/O area address setting is enabled only once. Do not change address in the middle of a program.**

Remark: M = xx00B
N = xx11B

(1) Peripheral area selection control register (BPC)

This register can be read/written in 16-bit units.

Figure 4-4: Peripheral Area Selection Control Register (BPC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BPC	PA15	0	PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	FFFFF064H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15</td><td>PA15</td><td>Enables/disables usage of programmable peripheral I/O area
<table>
<tr><th>PA15</th><th>Usage of Programmable Peripheral I/O Area</th></tr>
<tr><td>0</td><td>Disables usage of programmable peripheral I/O area</td></tr>
<tr><td>1</td><td>Enables usage of programmable peripheral I/O area</td></tr>
</table></td></tr>
<tr><td>13 to 0</td><td>PA13 to PA0</td><td>Specifies the bit 27 to bit 14 of the starting address of the programmable peripheral I/O area (The other bits are fixed at zero).</td></tr>
</table>

Remark: The recommended setting for the peripheral area selection control register (BPC) is x8600H. With this configuration the effective start address of the programmable peripheral area is mapped to x1800000H.

4.5 Bus Cycle Type Control Function

In the V850E/CA2 Jupiter, the following external devices can be connected directly to each memory block.

- SRAM, external ROM, external I/O
- Page ROM

Connected external devices are specified by the bus cycle type configuration registers 0, 1 (BCT0, BCT1).

4.5.1 Bus cycle type configuration

(1) Bus cycle configuration registers 0, 1(BCT0, BCT1)

These registers can be read/written in 16-bit units

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BCT0	ME3	0	0	BT30	ME2	0	0	BT20	ME1	0	0	BT10	ME0	0	0	BT00	FFFFF480H	8888H
	$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$					

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BCT1	ME7	0	0	BT70	ME6	0	0	BT60	ME5	0	0	BT50	ME4	0	0	BT40	FFFFF482H	8888H
	$\overline{CS7}$				$\overline{CS6}$				$\overline{CS5}$				$\overline{CS4}$					

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 11, 7, 3
(BCT0)
15, 11, 7, 3
(BCT1)</td><td>MEn
(n = 0 to 7)</td><td>Memory Controller Enable
Sets memory controller operation enable for each chip select signal $\overline{CSn}$.
<table>
<tr><th>ME</th><th>Memory Controller Operation Enable</th></tr>
<tr><td>0</td><td>Operation disable</td></tr>
<tr><td>1</td><td>Operation enable</td></tr>
</table></td></tr>
<tr><td>12, 8, 4, 0
(BCT0),
(BCT1)</td><td>BTn0
(n = 0 to 7)</td><td>Bus Cycle Type
Specifies the device to be connected to the $\overline{CSn}$ signal...
<table>
<tr><th>BTn0</th><th>External Device Connected Directly to $\overline{CSn}$ signal</th></tr>
<tr><td>0</td><td>SRAM, external I/O</td></tr>
<tr><td>1</td><td>Page ROM</td></tr>
</table></td></tr>
</table>

Cautions: 1. Write to the BCT0 and BCT1 registers after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the BCT0 and BCT1 registers is finished. However, it is possible to access external memory areas whose initialization has been finished.

2. The bits marked as 0 are reserved. They have to leave to 0.

4.6 Bus Access

4.6.1 Number of access clocks

The number of basic clocks necessary for accessing each resource is as follows.

Table 4-1: Number of Bus Access Clocks

Bus Cycle Configuration \ Resources (Bus width)		Internal Instruction Cache (32 bits)	Internal RAM (32 bits)	Peripheral I/O (16 bits)	External memory (16 bits)
Instruction fetch	Normal access	1 Note 1	1 Note 1	-	2 Note 2
	Branch	2	1	-	2 Note 2
Operand data access		-	1	3 Note 2	2 Note 2

Notes: 1. The instruction fetch becomes 2 clocks, in case of contention with data access.

2. This is the minimum value.

4.6.2 Bus sizing function

The bus sizing function controls data bus width for each CS area. The data bus width is specified by using the bus size configuration register (BSC).

(1) Bus size configuration register (BSC)

This register can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BSC	0	BS70	0	BS60	0	BS50	0	BS40	0	BS30	0	BS20	0	BS10	0	BS00	FFFFF066H	5555H
	$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$			

Bit Position	Bit Name	Function
15 to 0	BSn1, BSn0 (n = 0 to 7)	Data Bus Width Sets the data bus width of CSn area. BSn0 | Data Bus Width of CSn area 0 | 8 bits 1 | 16 bits

Cautions: 1. Write to the BSC register after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the BSC register is finished. However, it is possible to access external memory areas whose initialization has been finished.

2. When the data bus width is specified as 8 bits, only the $\overline{\text{LWR}}$ signal becomes active.

4.6.3 Endian control function

The endian control function can be used to set processing of word data in memory either by the Big Endian method or the Little Endian method for each CS area selected with the chip select signal ($\overline{\text{CS0}}$ to $\overline{\text{CS7}}$). Switching of the endian method is specified with the endian configuration register (BEC).

(1) Endian configuration register (BEC)

This register can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BEC	0	BE70	0	BE60	0	BE50	0	BE40	0	BE30	0	BE20	0	BE10	0	BE00	FFFFF068H	0000H
	$\overline{\text{CS7}}$		$\overline{\text{CS6}}$		$\overline{\text{CS5}}$		$\overline{\text{CS4}}$		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$			

Bit Position	Bit Name	Function
14, 12, 10, 8, 6, 4, 2, 0	BEn0 (n = 0 to 7)	Big Endian Specifies the endian method.

BEn0	Endian Control
0	Little Endian method
1	Big Endian method

Cautions: 1. Bits 15, 13, 11, 9, 7, 5, 3, and 1 of the BEC register must be cleared (0). If these bits are set to 1, the operation is not guaranteed.

2. Set the CSn area specified as the programmable peripheral I/O area to Little Endian format (n = 0 to 7).

3. In the following areas, the data processing method is fixed to Little Endian method. Any setting of Big Endian method for these areas according to the BEC register is invalid.
- On-chip peripheral I/O area
- Internal RAM area
- Fetch area of external memory

Figure 4-5: Big Endian Addresses within Word

31 24	23 16	17 8	7 0
0008H	0009H	000AH	000BH
0004H	0005H	0006H	0007H
0000H	0001H	0002H	0003H

Figure 4-6: Little Endian Addresses within Word

31 24	23 16	17 8	7 0
000BH	000AH	0009H	0008H
0007H	0006H	0005H	0004H
0003H	0002H	0001H	0000H

4.7 Cache Configuration

The cache configuration register (BHC) is used to set the cache memory configuration for each CS area selected by the chip select signals ($\overline{\text{CS0}}$ to $\overline{\text{CS7}}$).

(1) Cache configuration register (BHC)

This register can be read or written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BEC	0	BH70	0	BH60	0	BH50	0	BH40	0	BH30	0	BH20	0	BH10	0	BH00	FFFFF06AH	0000H
	$\overline{\text{CS7}}$		$\overline{\text{CS6}}$		$\overline{\text{CS5}}$		$\overline{\text{CS4}}$		$\overline{\text{CS3}}$		$\overline{\text{CS2}}$		$\overline{\text{CS1}}$		$\overline{\text{CS0}}$			

Bit Position	Bit Name	Function
14, 12, 10, 8, 6, 4, 2, 0	BHn0 (n = 0 to 7)	Sets whether or not the instruction cache located in the block n area can be used.

BHn0	Instruction Cache Setting
0	Cache not available
1	Cache available

Cautions: 1. Be sure to disable the cache for big endian format CS area and CS areas set as the following areas.

- **ROM area**
- **RAM area**
- **Peripheral I/O area**
- **Programmable peripheral I/O area**

2. The bits marked as 0 are reserved. They have to leave to 0

Note: n = 0 to 7

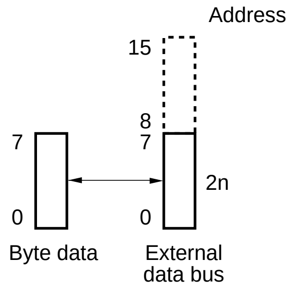
4.7.1 Bus width

The V850E/CA2 Jupiter accesses peripheral I/O and external memory in 8-bit, 16-bit, or 32-bit units. The following shows the operation for each type of access. Access all data in order starting from the lower order side.

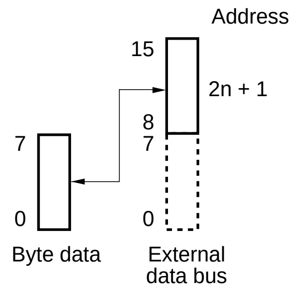
(1) Byte access (8 bits)

(a) When the data bus width is 16 bits (Little Endian)

<1> Access to even address (2n)

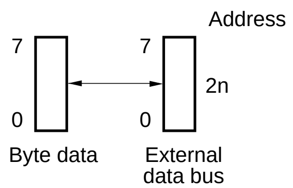


<2> Access to odd address (2n + 1)

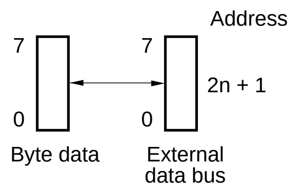


(b) When the data bus width is 8 bits (Little Endian)

<1> Access to even address (2n)

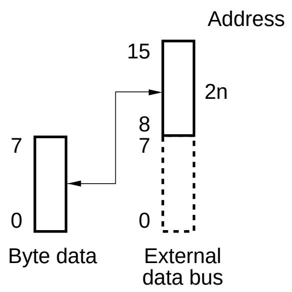


<2> Access to odd address (2n + 1)

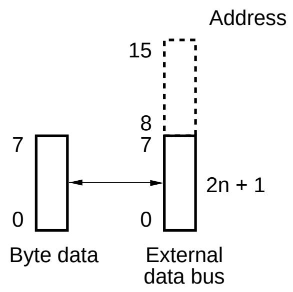


(c) When the data bus width is 16 bits (Big Endian)

<1> Access to even address (2n)

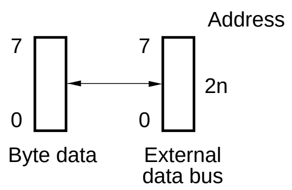


<2> Access to odd address (2n + 1)

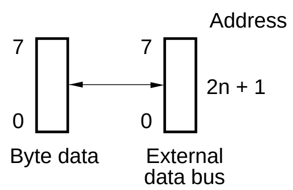


(d) When the data bus width is 8 bits (Big Endian)

<1> Access to even address (2n)



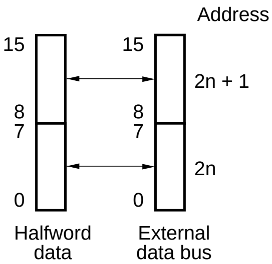
<2> Access to odd address (2n + 1)



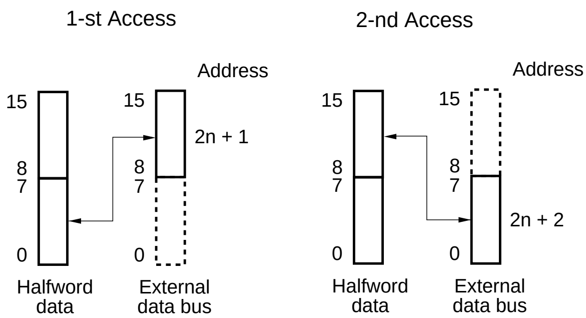
(2) Halfword access (16 bits)

(a) When the bus width is 16 bits (Little Endian)

<1> Access to even address (2n)

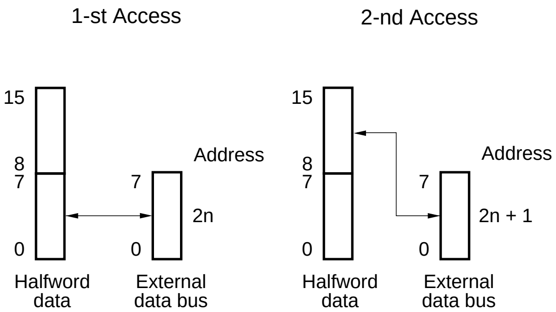


<2> Access to odd address (2n + 1)

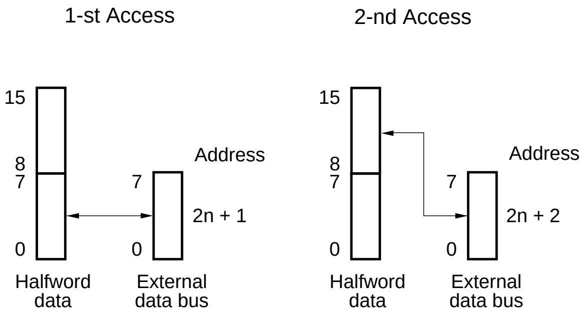


(b) When the data bus width is 8 bits (Little Endian)

<1> Access to even address (2n)

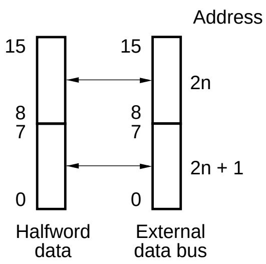


<2> Access to odd address (2n + 1)

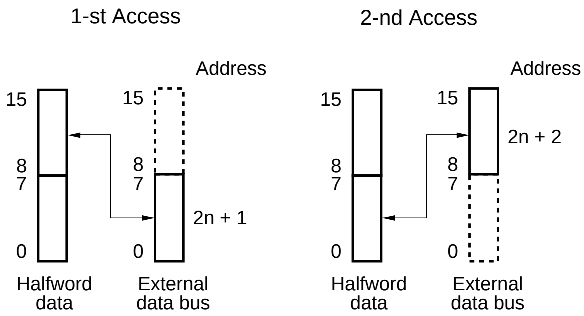


(c) When the data bus width is 16 bits (Big Endian)

<1> Access to even address (2n)

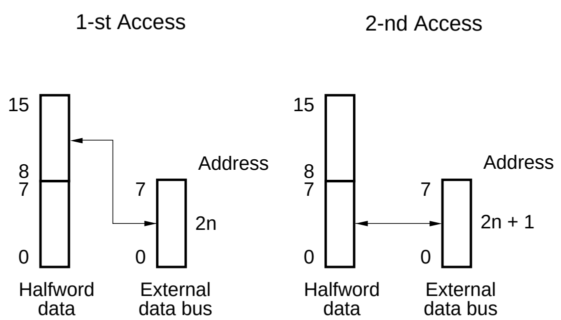


<2> Access to odd address (2n + 1)

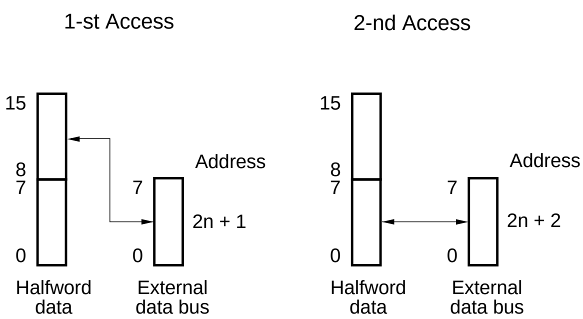


(d) When the data bus width is 8 bits (Big Endian)

<1> Access to even address (2n)



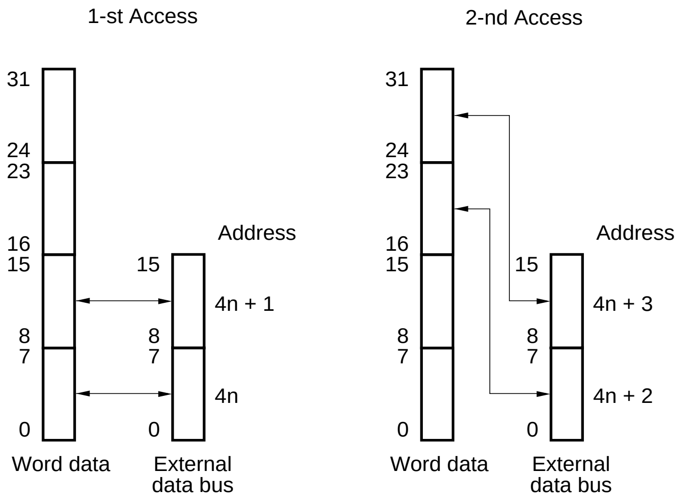
<2> Access to odd address (2n + 1)



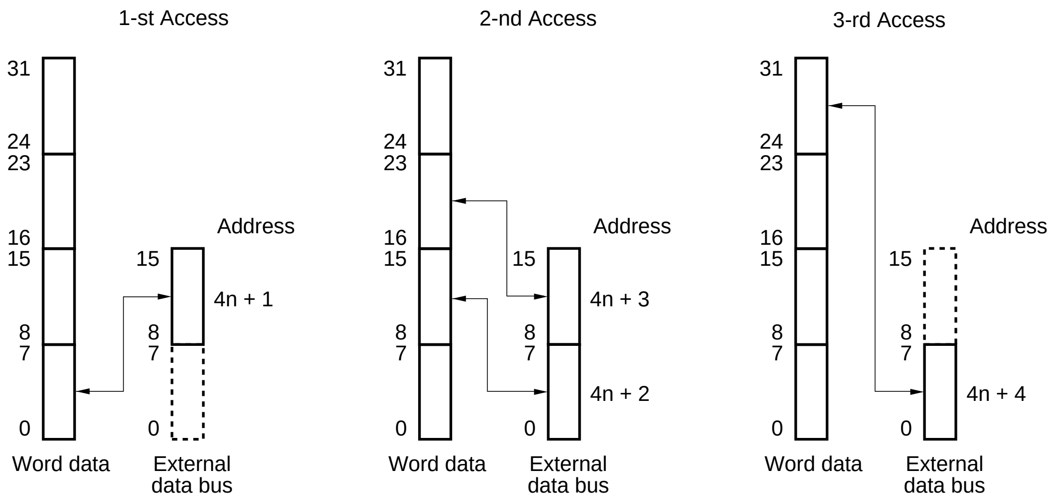
(3) Word access (32 bits)

(a) When the bus width is 16 bits (Little Endian)

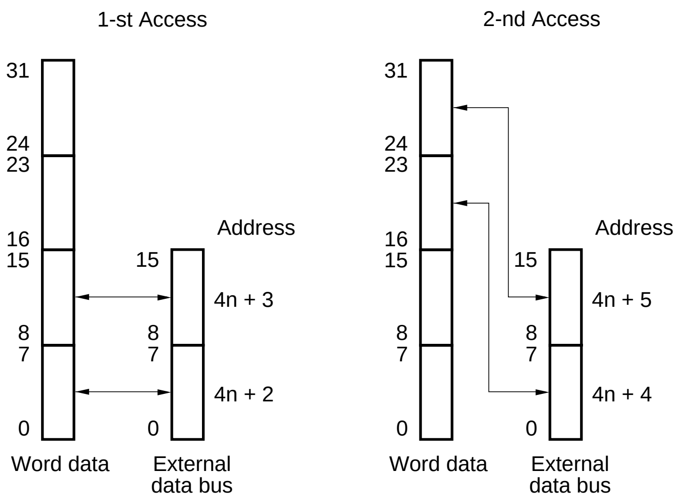
<1> Access to address 4n



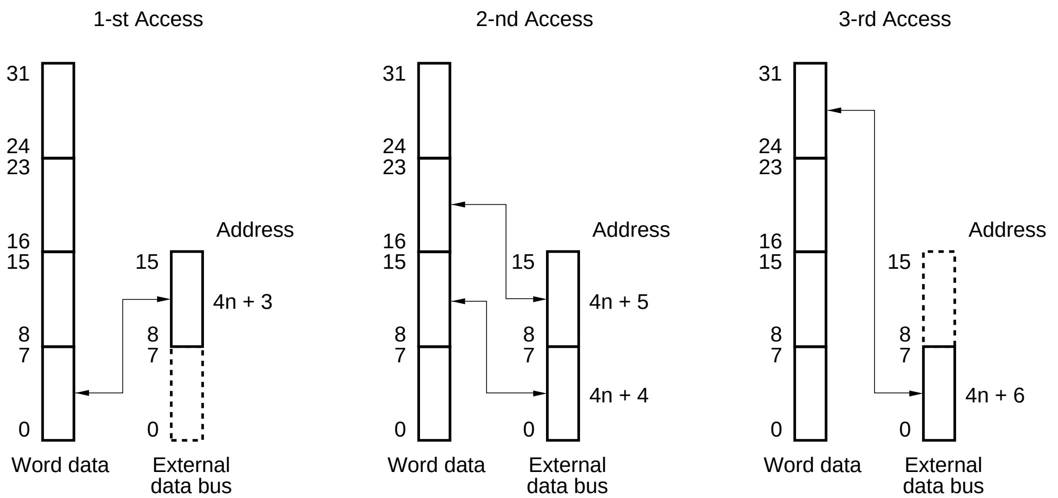
<2> Access to address 4n + 1



<3> Access to address 4n + 2

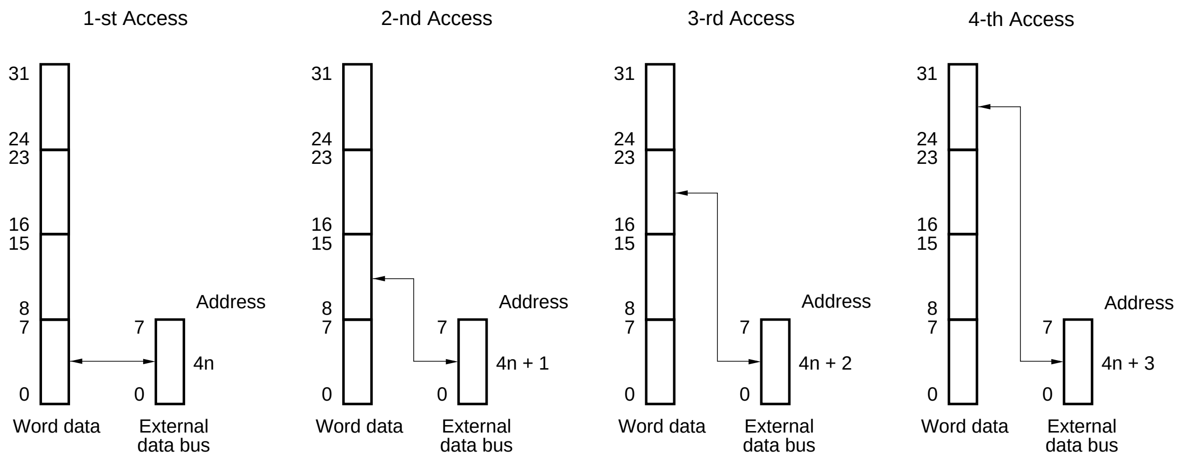


<4> Access to address 4n + 3

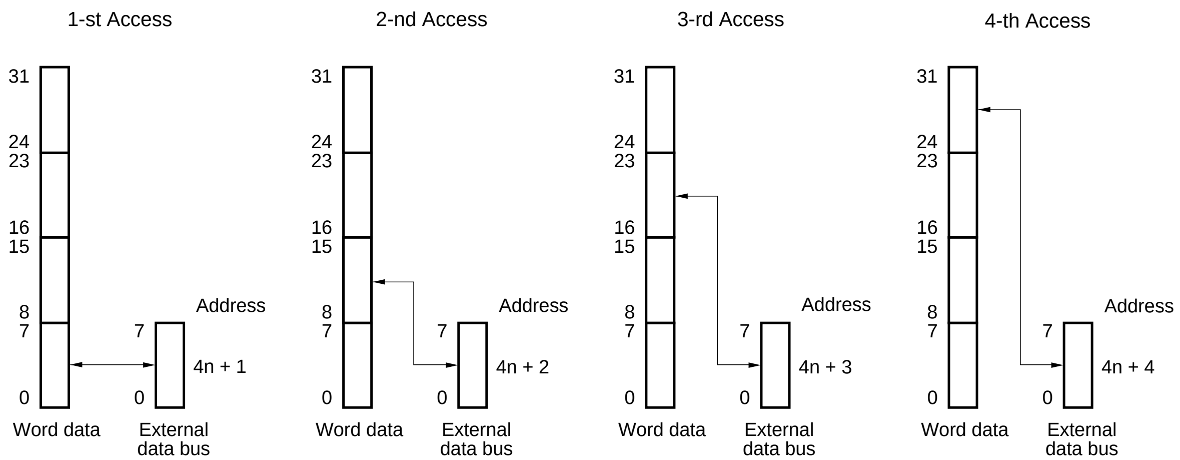


(b) When the bus width is 8 bits (Little Endian)

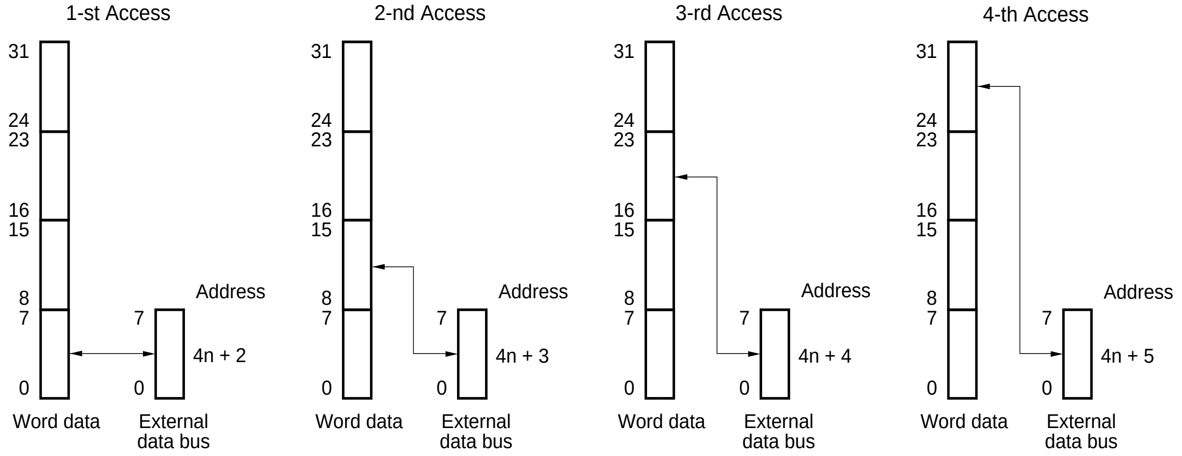
<1> Access to address 4n



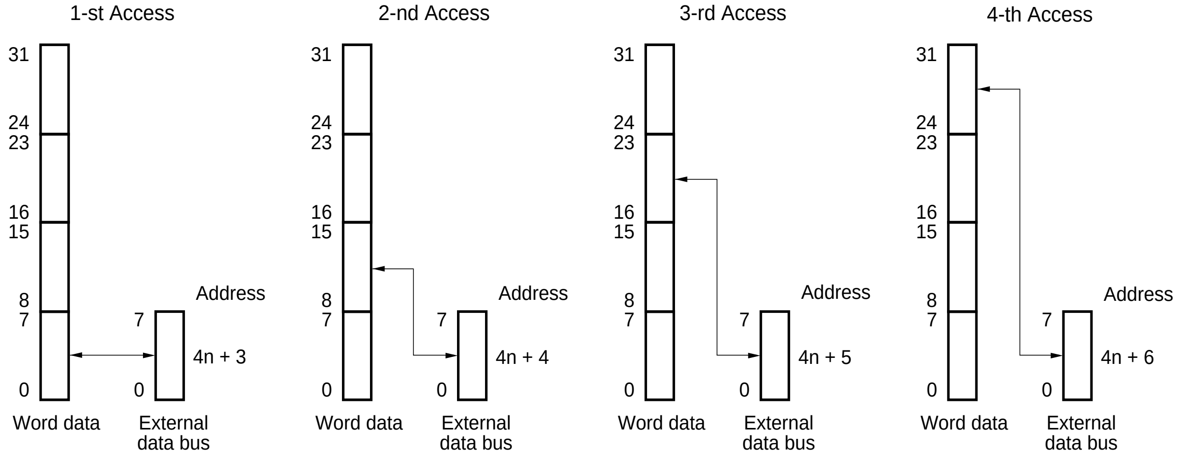
<2> Access to address 4n + 1



<3> Access to address 4n + 2

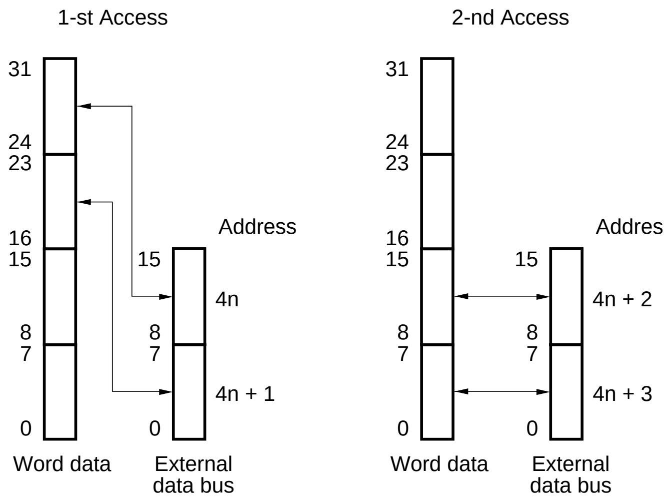


<4> Access to address 4n + 3

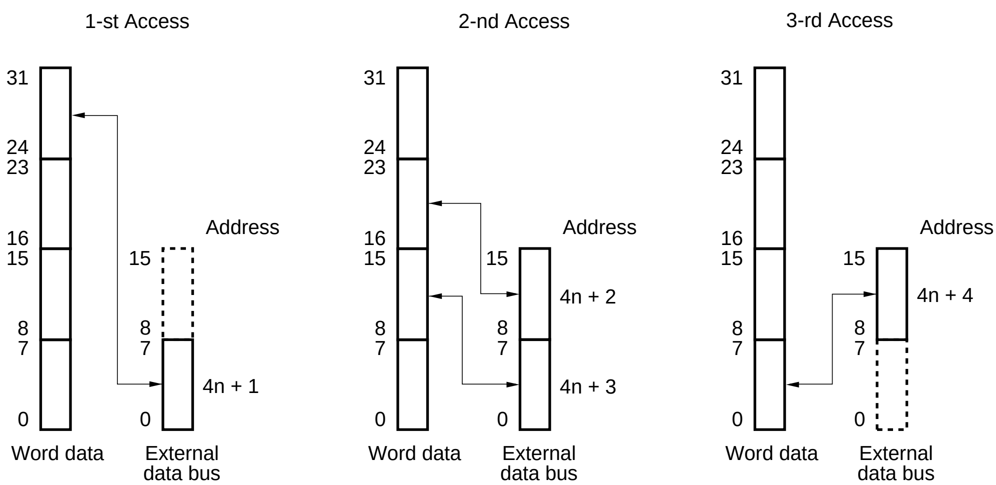


(c) When the data bus width is 16 bits (Big Endian)

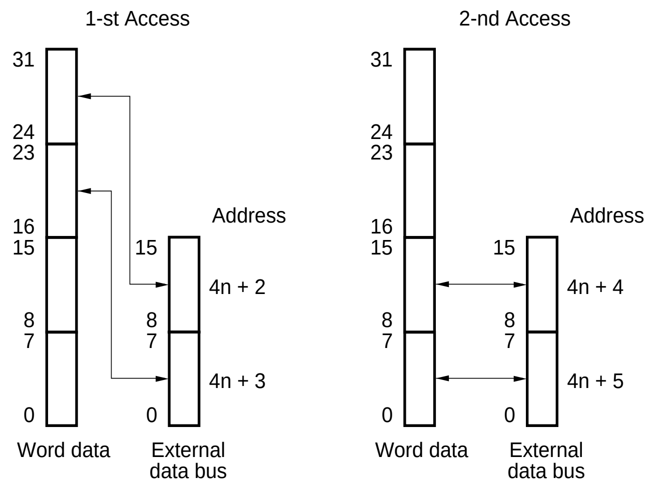
<1> Access to address 4n



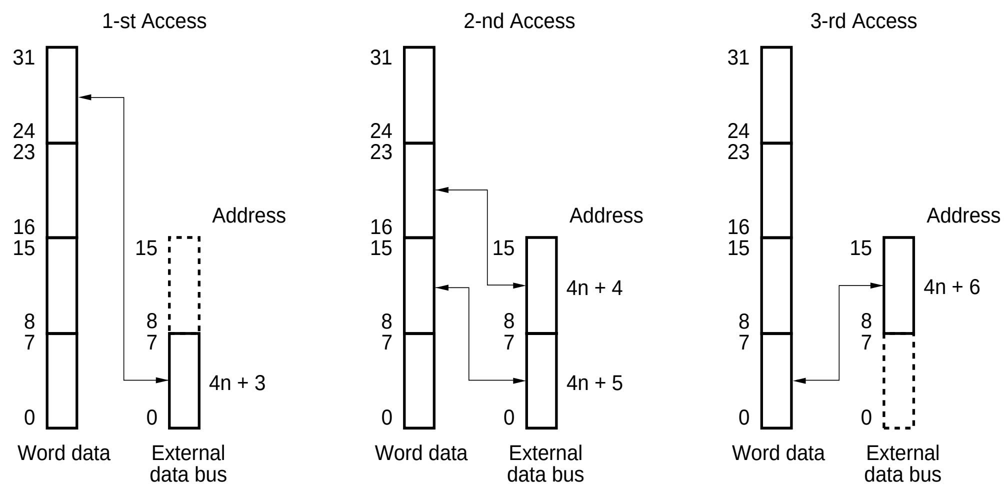
<2> Access to address 4n + 1



<3> Access to address 4n + 2

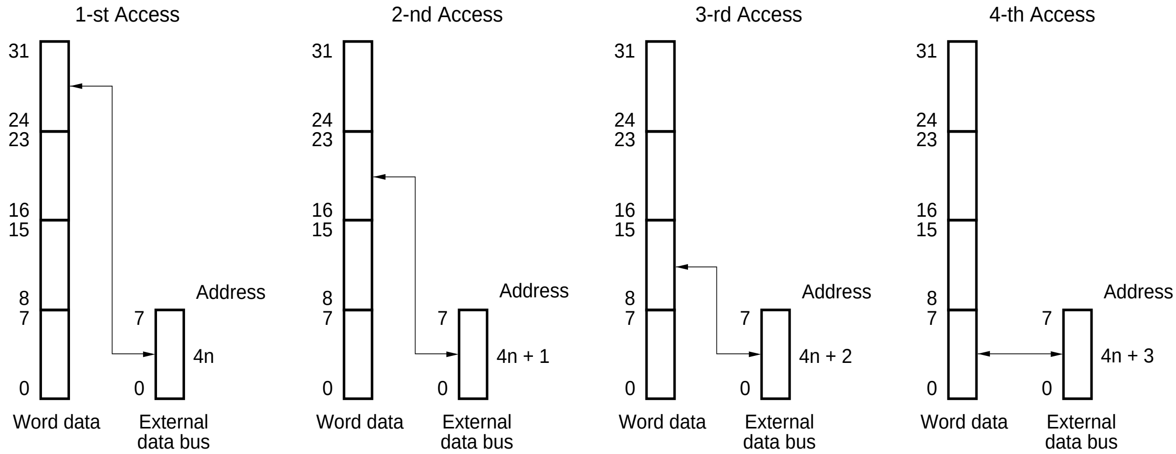


<4> Access to address 4n + 3

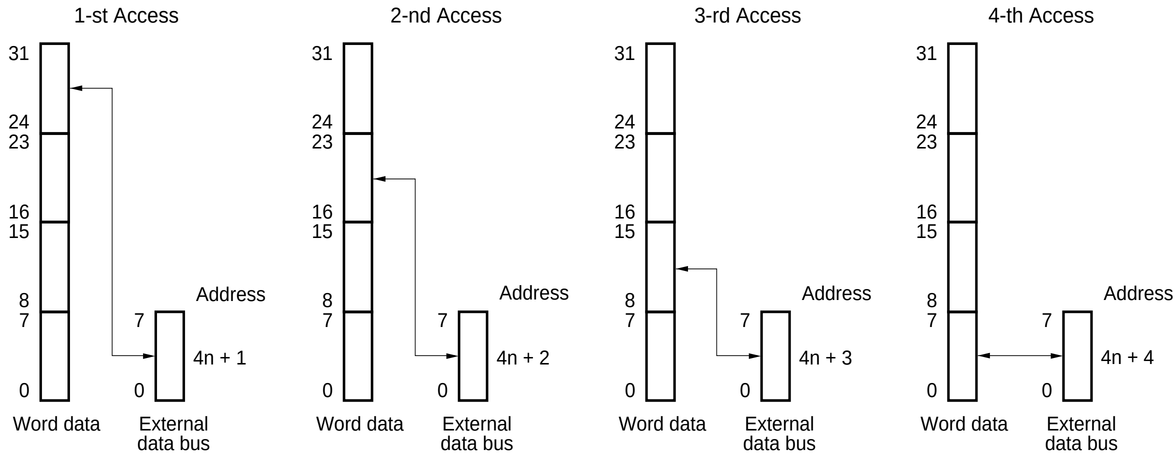


(d) When the data bus width is 8 bits (Big Endian)

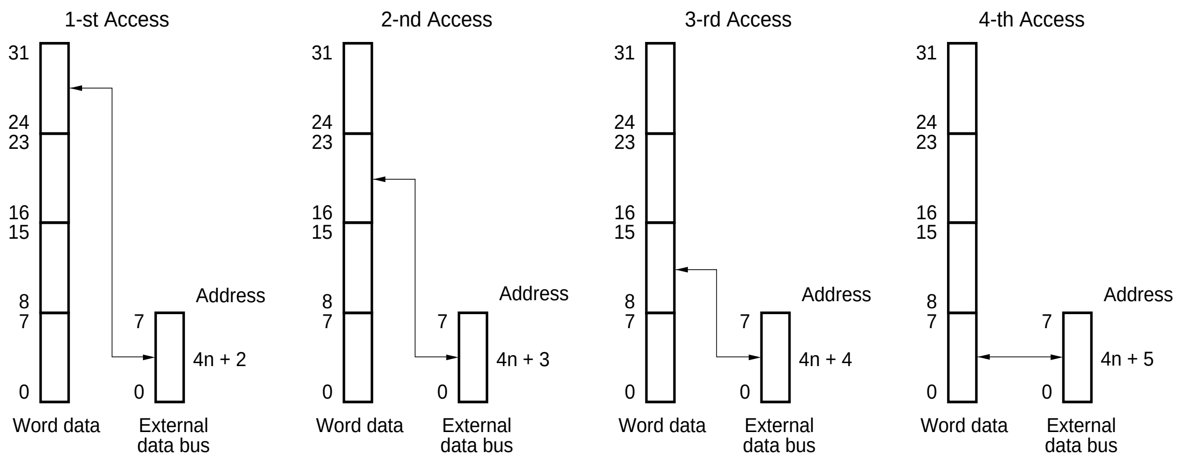
<1> Access to address 4n



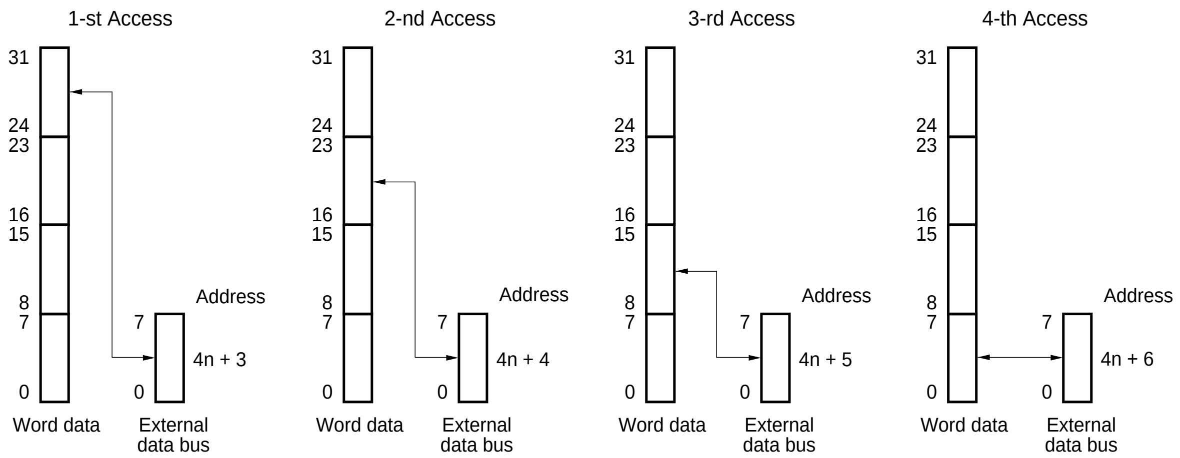
<2> Access to address 4n + 1



<3> Access to address 4n + 2



<4> Access to address 4n + 3



4.8 Wait Function

4.8.1 Programmable wait function

(1) Data wait control registers 0, 1 (DWC0, DWC1)

With the purpose of realizing easy interfacing with low-speed memory or with I/Os, it is possible to insert up to 7 data wait states with respect to the starting bus cycle for each CS area.
The number of wait states can be specified by data wait control registers 0 and 1 (DWC0, DWC1) in programming. Just after system reset, all blocks have 7 data wait states inserted.

These registers can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DWC0	0	DW32	DW31	DW30	0	DW22	DW21	DW20	0	DW12	DW11	DW10	0	DW02	DW01	DW00	FFFFF484H	7777H
		$\overline{CS3}$				$\overline{CS2}$				$\overline{CS1}$				$\overline{CS0}$				

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DWC1	0	DW72	DW71	DW70	0	DW62	DW61	DW60	0	DW52	DW51	DW50	0	DW42	DW41	DW40	FFFFF486H	7777H
		$\overline{CS7}$				$\overline{CS6}$				$\overline{CS5}$				$\overline{CS4}$				

Bit Position	Bit Name	Function
14 to 12, 10 to 8, 6 to 4, 2 to 0	DWn2 to DWn0 (n = 0 to 7)	Data Wait Specifies the number of wait states inserted in the $\overline{CSn}$ area.

DWn2	DWn1	DWn0	Number of Wait States Inserted in $\overline{CSn}$ Space
0	0	0	No wait states inserted
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

Cautions: 1. The internal Cache and the internal RAM area are not subject to programmable waits and ordinarily no wait access is carried out. The internal peripheral I/O area is also not subject to programmable wait states, with wait control performed only by each peripheral function.

2. In the following cases, the settings of registers DWC0 and DWC1 are invalid (wait control is performed by each memory controller).
- Page ROM on-page access

3. Write to the DWC0 and DWC1 registers after reset, and then do not change the set values. Also, do not access an external memory area other than that for this initialization routine until initial setting of the DWC0 and DWC1 registers is finished. However, it is possible to access external memory areas whose initialization has been finished.

(2) Address setup wait control register (ASC)

The V850E/CA2 Jupiter allows insertion of address setup wait states before the T1 cycle of the SRAM or page ROM cycle.
The number of address setup wait states can be set with the ASC register for each CS area.

This register can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ASC	AC71	AC70	AC61	AC60	AC51	AC50	AC41	AC40	AC31	AC30	AC21	AC20	AC11	AC10	AC01	AC00	FFFFF48AH	FFFFH
	$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$			

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>ACn1,
ACn0
(n = 0 to 7)</td><td>Address Cycle
Specifies the number of address setup wait states inserted before the T1 cycle of SRAM/page ROM cycle for each CS area.
<table>
<tr><th>ACn1</th><th>ACn0</th><th>Number of Wait States</th></tr>
<tr><td>0</td><td>0</td><td>Not inserted</td></tr>
<tr><td>0</td><td>1</td><td>1</td></tr>
<tr><td>1</td><td>0</td><td>2</td></tr>
<tr><td>1</td><td>1</td><td>3</td></tr>
</table></td></tr>
</table>

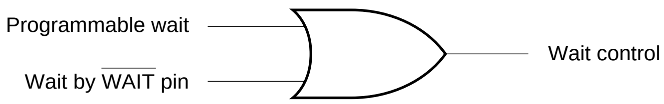
Remark: During address setup wait, the external wait function is disabled by the $\overline{\text{WAIT}}$ pin.

4.8.2 External wait function

When an extremely slow device, I/O, or asynchronous system is connected, any number of wait states can be inserted in a bus cycle by the external wait pin ($\overline{\text{WAIT}}$) to synchronize with the external device.
Just as with programmable waits, access to internal ROM, internal RAM, and internal peripheral I/O areas cannot be controlled by external waits.
Input of the external $\overline{\text{WAIT}}$ signal can be done asynchronously to the system clock and is sampled at the rising edge of the clock in the T1 and TW states of a bus cycle. If the setup/hold time at sampling timing is not satisfied, the wait state may or may not be inserted in the next state.

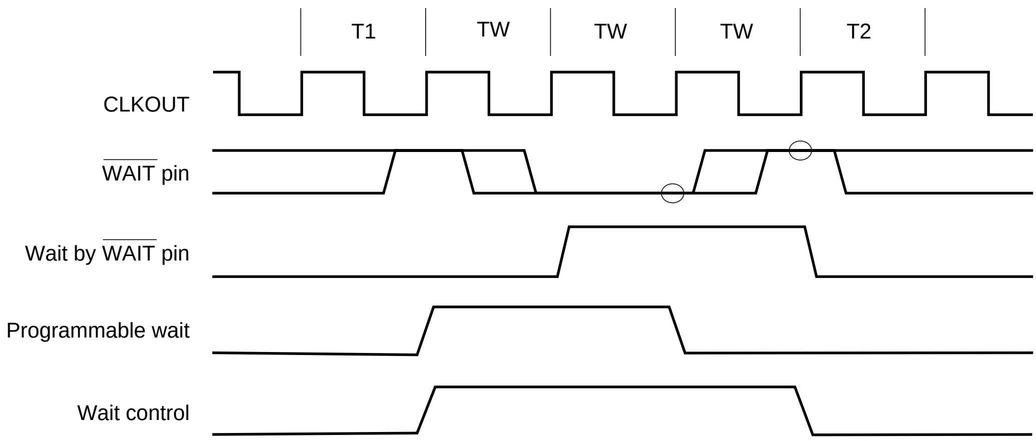
4.8.3 Relationship between programmable wait and external wait

A wait cycle is inserted as the result of an OR operation between the wait cycle specified by the set value of the programmable wait and the wait cycle controlled by the $\overline{\text{WAIT}}$ pin. In other words, the number of wait cycles is determined by the side with the greatest number of cycles.



For example, if the programmable wait and the timing of the $\overline{\text{WAIT}}$ pin signal are as illustrated below, three wait states will be inserted in the bus cycle.

Figure 4-7: Example of Wait Insertion



Remark: The circle ❍ indicates the sampling timing.

4.9 Idle State Insertion Function

To facilitate interfacing with low-speed memory devices, an idle state (TI) can be inserted into the current bus cycle after the T2 state to meet the data output float delay time (tdf) on memory read access for each CS space. The bus cycle following the T2 state starts after the idle state is inserted.

An idle state is inserted after read/write cycles for SRAM, external I/O, or external ROM.

In the following cases, an idle state is inserted in the timing.

- after read/write cycles for SRAM, external I/O, or external ROM

The idle state insertion setting can be specified by program using the bus cycle control register (BCC). Immediately after the system reset, idle state insertion is automatically programmed for all memory blocks.

(1) Bus cycle control register (BCC)

This register can be read/written in 16-bit units.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
BCC	BC71	BC70	BC61	BC60	BC51	BC50	BC41	BC40	BC31	BC30	BC21	BC20	BC11	BC10	BC01	BC00	FFFFF488H	FFFFH
	$\overline{CS7}$		$\overline{CS6}$		$\overline{CS5}$		$\overline{CS4}$		$\overline{CS3}$		$\overline{CS2}$		$\overline{CS1}$		$\overline{CS0}$			

Bit Position	Bit Name	Function		
15 to 0	BCn1, BCn0 (n = 0 to 7)	Data Cycle Specifies the insertion of an idle state when accessing corresponding $\overline{CSn}$ area.		
		BCn1	BCn0	Idle State in $\overline{CSn}$ Area
		0	0	Not inserted
		0	1	1
		1	0	2
		1	1	3

Cautions: 1. The internal iCache area, the internal RAM area and the internal peripheral I/O area are not subject to insertion of an idle state.

2. Write to the BCC register after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the BCC register is finished. However, it is possible to access external memory areas whose initialization has been finished.

4.10 Bus Priority Order

There are three external bus cycles: DMA cycle, operand data access and instruction fetch.
As for the priority order, the highest priority has the DMA cycle, instruction fetch, and operand data access, in this order.
An instruction fetch may be inserted between read access and write access during read modify write access.
Also, an instruction fetch may be inserted between bus access and bus access during CPU bus clock.

Table 4-2: Bus Priority Order

Priority Order	External Bus Cycle	Bus Master
High	DMA cycle	DMA controller
↕	Operand data access	CPU
Low	Instruction fetch	CPU

4.11 Boundary Operation Conditions

4.11.1 Program space

(1) Branching to the peripheral I/O area or successive fetch from the internal RAM area to the internal peripheral I/O area is inhibited. In terms of hardware, fetching the NOP opcode continues, and fetching from the external memory is not performed.

(2) If a branch instruction exists at the upper limit of the internal RAM area, a pre-fetch operation (invalid fetch) that straddles over the internal peripheral I/O area does not occur when instruction fetch is performed.

4.11.2 Data space

The V850E/CA2 Jupiter is provided with an address misalign function.
Through this function, regardless of the data format (word data, halfword data, or byte data), data can be placed in all addresses. However, in the case of word data and halfword data, if data are not subjected to boundary alignment, the bus cycle will be generated a minimum of 2 times and bus efficiency will drop.

(1) In the case of halfword length data access

When the address's LSB bit is 1, the byte length bus cycle will be generated 2 times.

(2) In the case of word length data access

(a) When the address's LSB is 1, bus cycles will be generated in the order of byte length bus cycle, halfword length bus cycle, and word length bus cycle.

(b) When the address's lowest 2 bits are 10, the halfword length bus cycle will be generated 2 times.

Chapter 5 Memory Access Control Function

5.1 SRAM, External ROM, External I/O Interface

5.1.1 Features

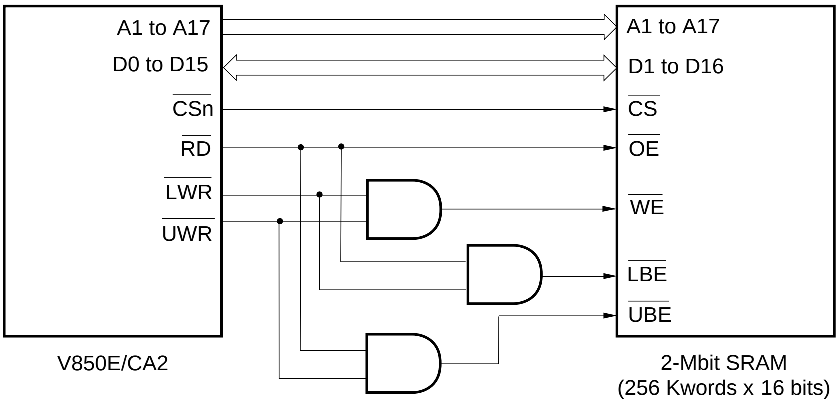
- Access to SRAM takes a minimum of 2 states.
- Up to 7 states of programmable data waits can be inserted through setting of the DWC0 and DWC1 registers.
- Data wait can be controlled with input pin ($\overline{\text{WAIT}}$).
- Up to 3 idle states can be inserted after the read/write cycle through setting of the BCC register.
- Up to 3 address set up wait states can be inserted through setting of the ASC register.

5.1.2 SRAM connections

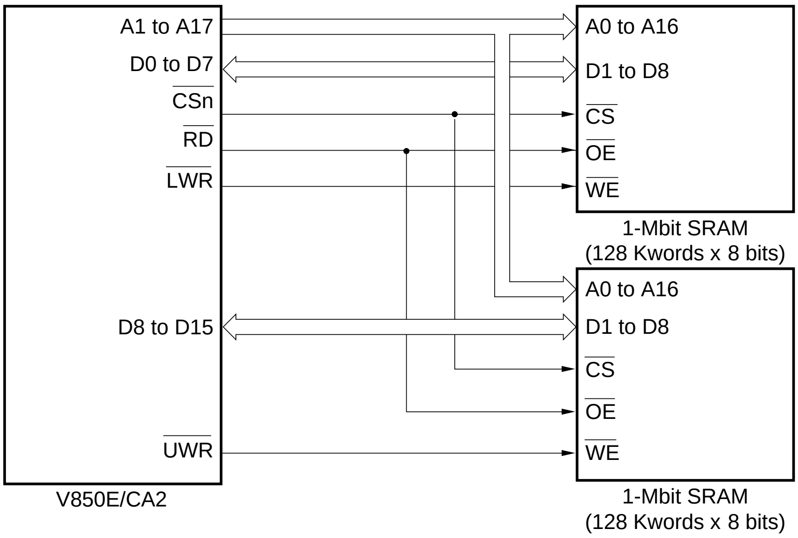
An example of connection to SRAM is shown below.

Figure 5-1: Example of Connection to SRAM

(a) When data bus width is 16 bits



(b) When data bus width is 8 bits

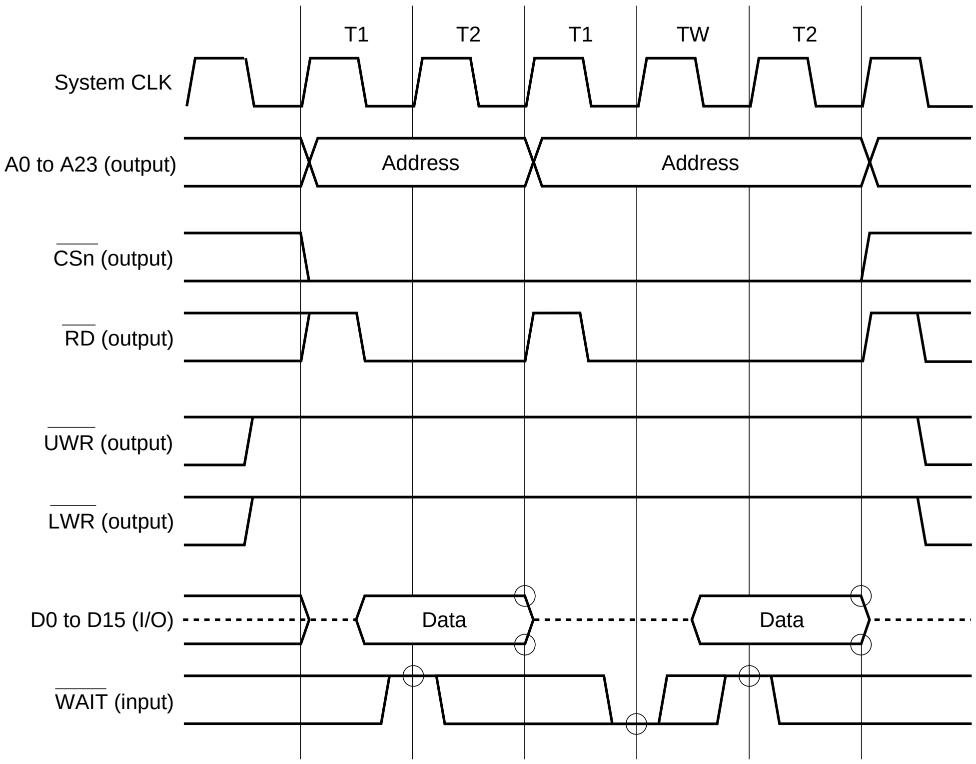


Remark: $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

5.1.3 SRAM, external ROM, external I/O access

Figure 5-2: SRAM, External ROM, External I/O Access Timing (1/6)

(a) During read



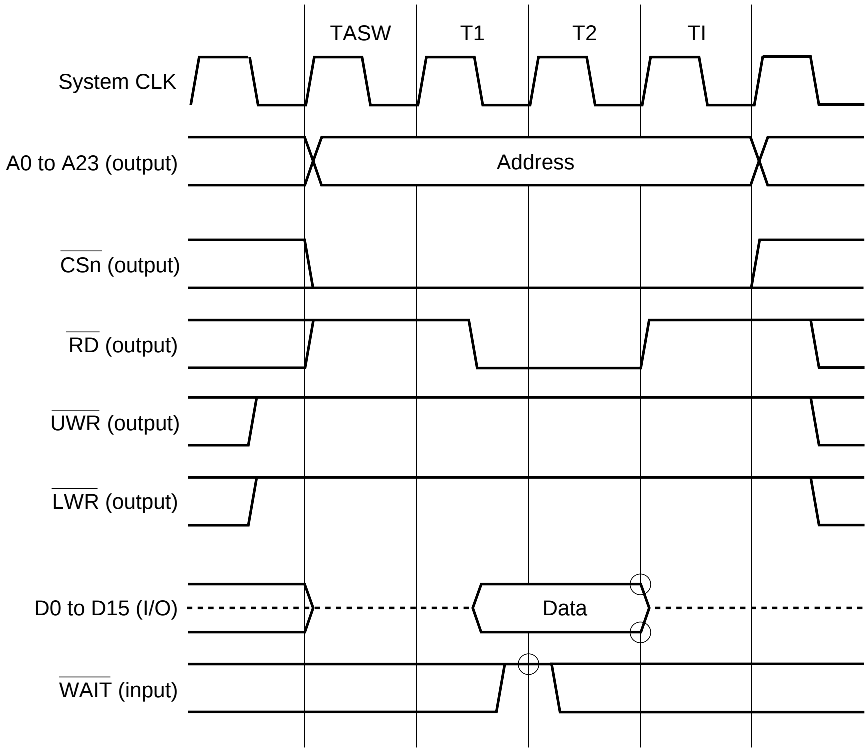
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (2/6)

(b) During read (address setup wait, idle state insertion)



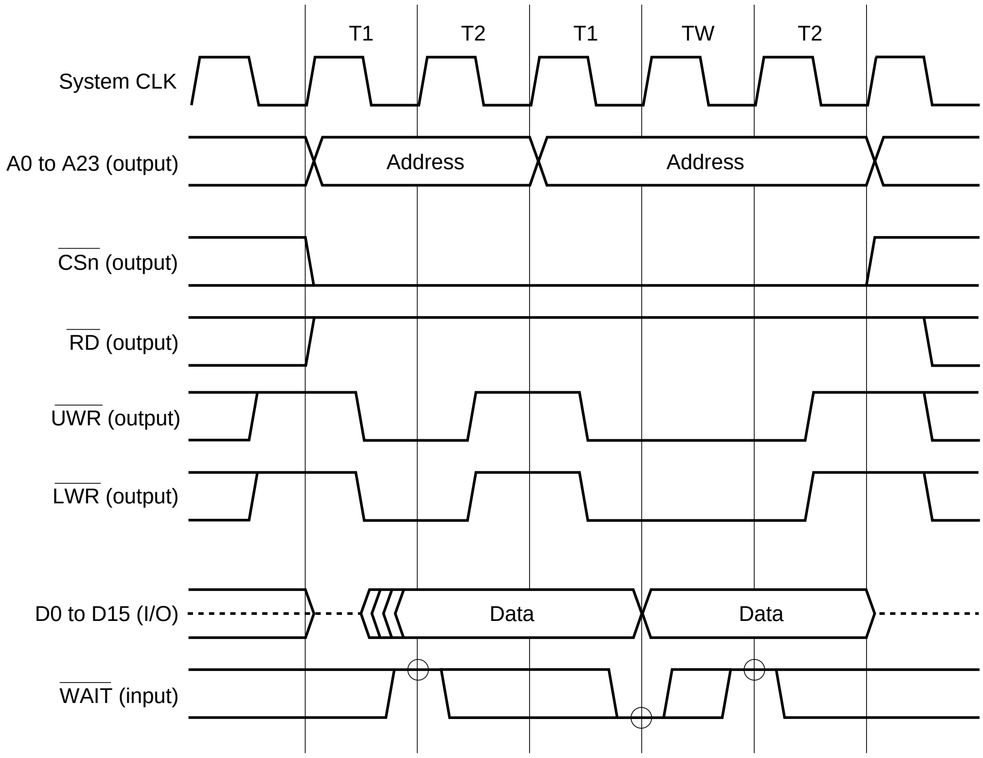
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (3/6)

(c) During write



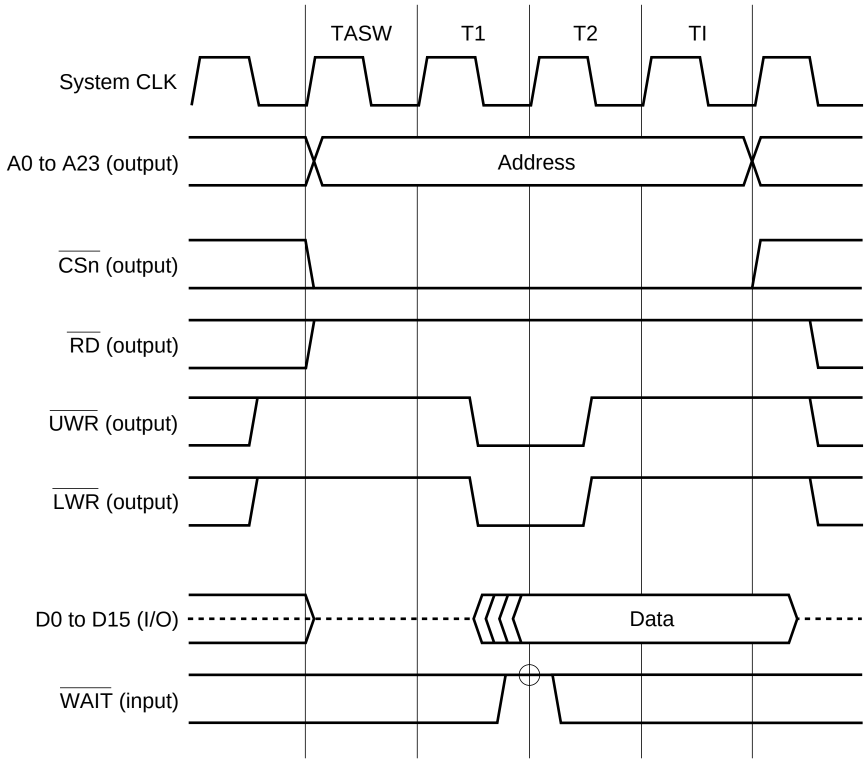
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (4/6)

(d) During write (address setup wait, idle state insertion)



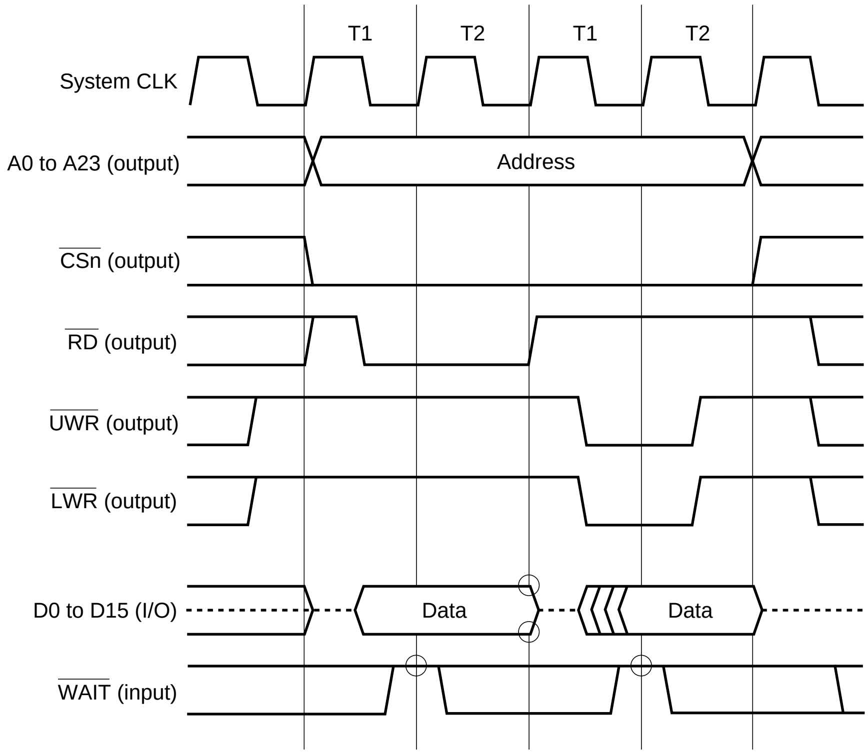
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{CSn}$ = $\overline{CS0}$, $\overline{CS3}$ and $\overline{CS4}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (5/6)

(e) When read → write operation



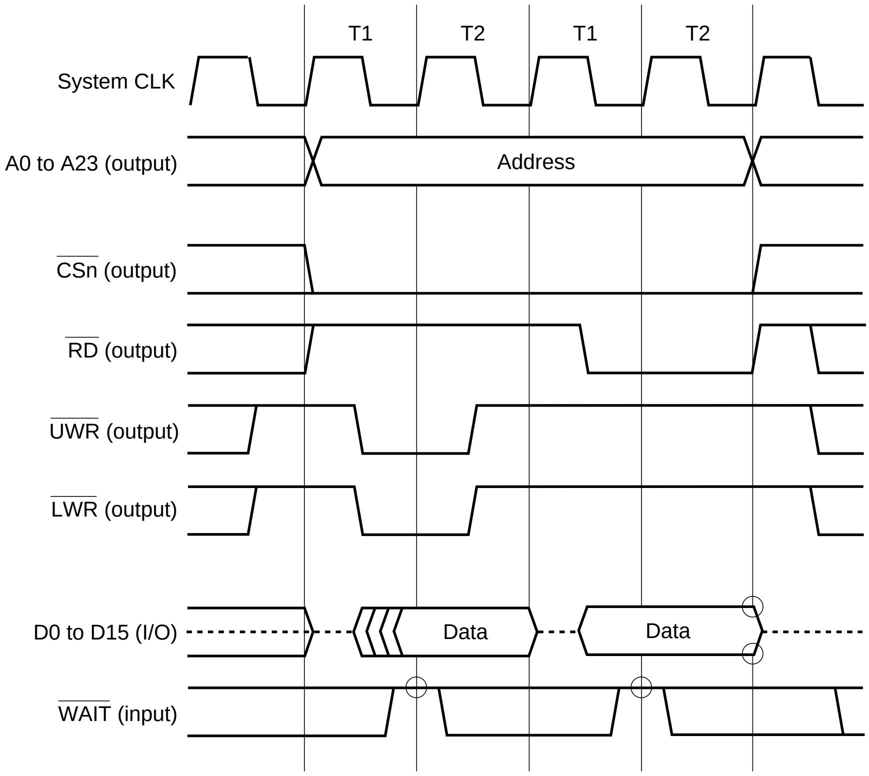
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

Figure 5-2: SRAM, External ROM, External I/O Access Timing (6/6)

(f) When write → read operation



Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

5.2 Page ROM Controller (ROMC)

The page ROM controller (ROMC) is provided for access to ROM (page ROM) with the page access function.
Comparison of addresses with the immediately preceding bus cycle is carried out and wait control for normal access (off-page) and page access (on-page) is executed. This controller can handle page widths from 8 to 128 bytes.

5.2.1 Features

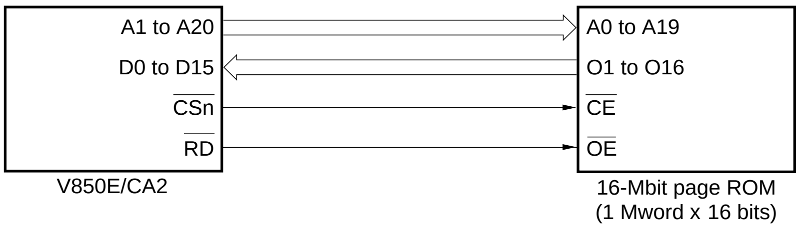
- Direct connection to 8-bit/16-bit page ROM supported
- In case of 16-bit bus width: 4/8/16/32/64 word page access supported
- In case of 8-bit bus width: 8/16/32/64/128 word page access supported
- Page ROM access a minimum of 2 states.
- On-page judgment function
- Addresses to be compared can be changed through setting of the PRC register.
- Up to 7 states of programmable data waits can be inserted during the on-page cycle through setting of the PRC register.
- Up to 7 states of programmable data wait can be inserted during the off-page cycle through setting of the DWC0 and DWC1 registers.
- Waits can be controlled with pin input.

5.2.2 Page ROM connections

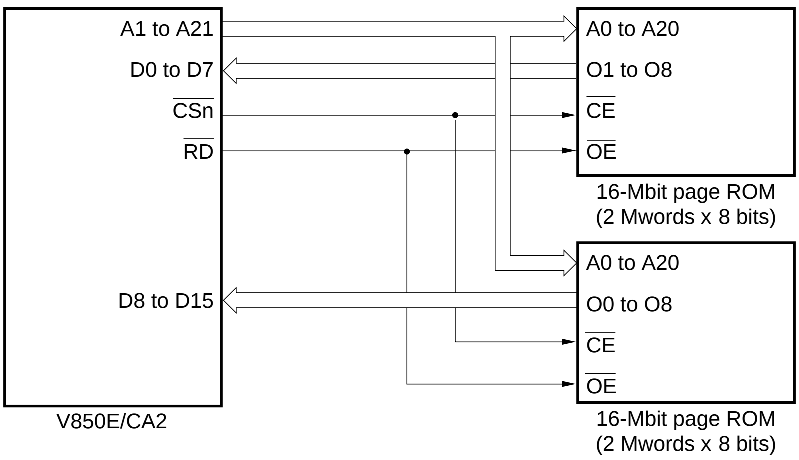
Examples of page ROM connections are shown below.

Figure 5-3: Example of Page ROM Connections

(a) In case of 16-bit data bus width



(b) In case of 8-bit data bus width



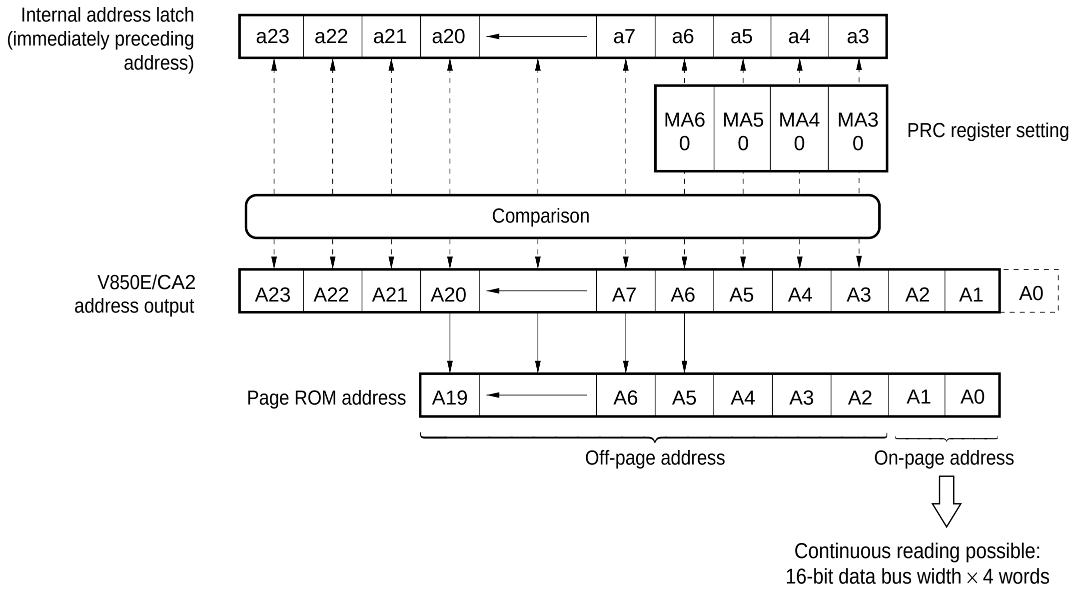
Remark: $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

5.2.3 On-page/off-page judgment

Whether a page ROM cycle is on-page or off-page is judged by latching the address of the previous cycle and comparing it with the address of the current cycle.
Through the page ROM configuration register (PRC), according to the configuration of the connected page ROM and the number of continuously readable bits, one of the addresses (A3 to A6) is set as the masking address (no comparison is made).

Figure 5-4: On-Page/Off-Page Judgment during Page ROM Connection (1/2)

(a) In case of 16-Mbit (1 M × 16 bits) page ROM (4-word page access)



(b) In case of 16-Mbit (1 M × 16 bits) page ROM (8-word page access)

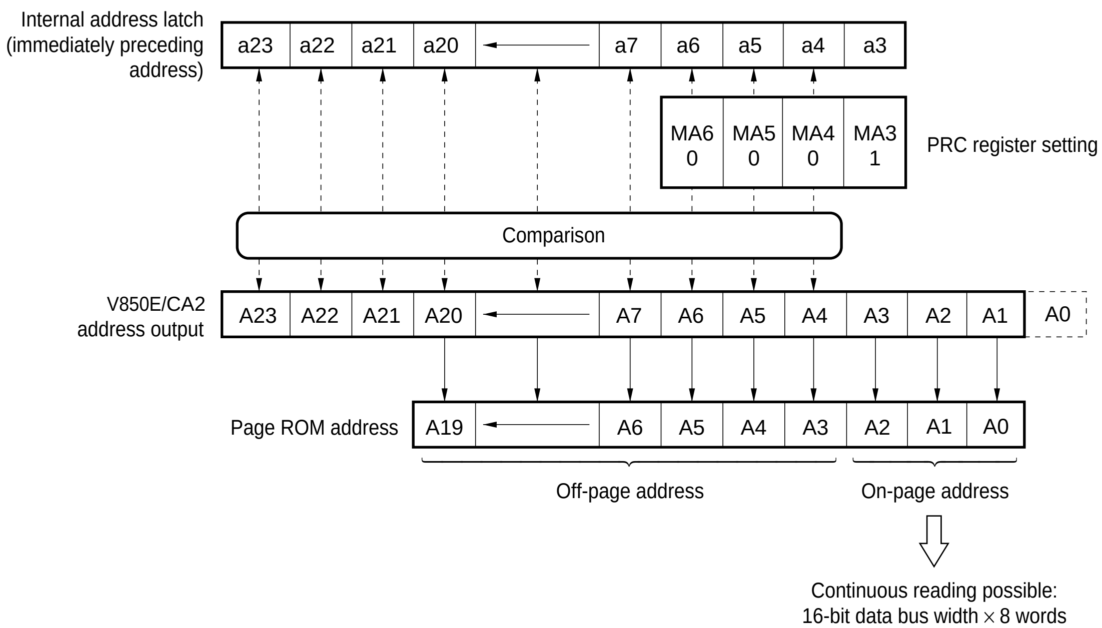
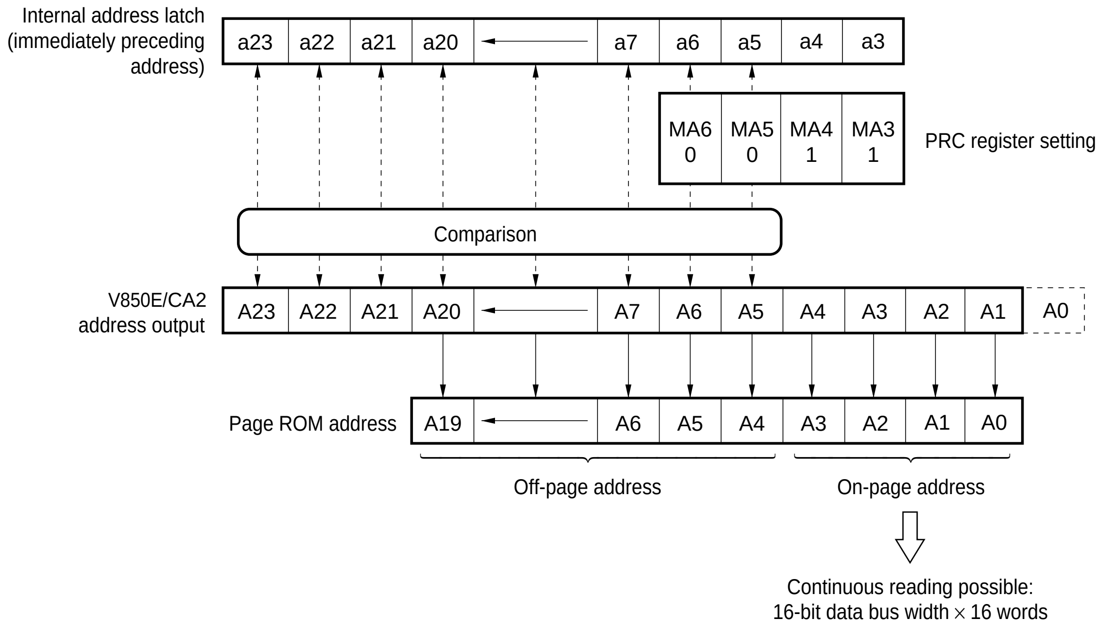


Figure 5-4: On-Page/Off-Page Judgment during Page ROM Connection (2/2)

(c) In case of 32-Mbit (2 M × 16 bits) page ROM (16-word page access)



5.2.4 Page ROM configuration register (PRC)

This register specifies whether page ROM on-page access is enabled or disabled. If on-page access is enabled, the masking address (no comparison is made) out of the addresses (A3 to A6) corresponding to the configuration of the page ROM being connected to and the number of bits that can be read continuously, as well as the number of waits corresponding to the internal system clock, are set.
This register can be read/written in 16-bit units.

Figure 5-5: Page ROM Configuration Register (PRC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
PRC	0	PRW2	PRW1	PRW0	0	0	0	0	0	0	0	0	MA6	MA5	MA4	MA3	FFFFF49AH	7000H

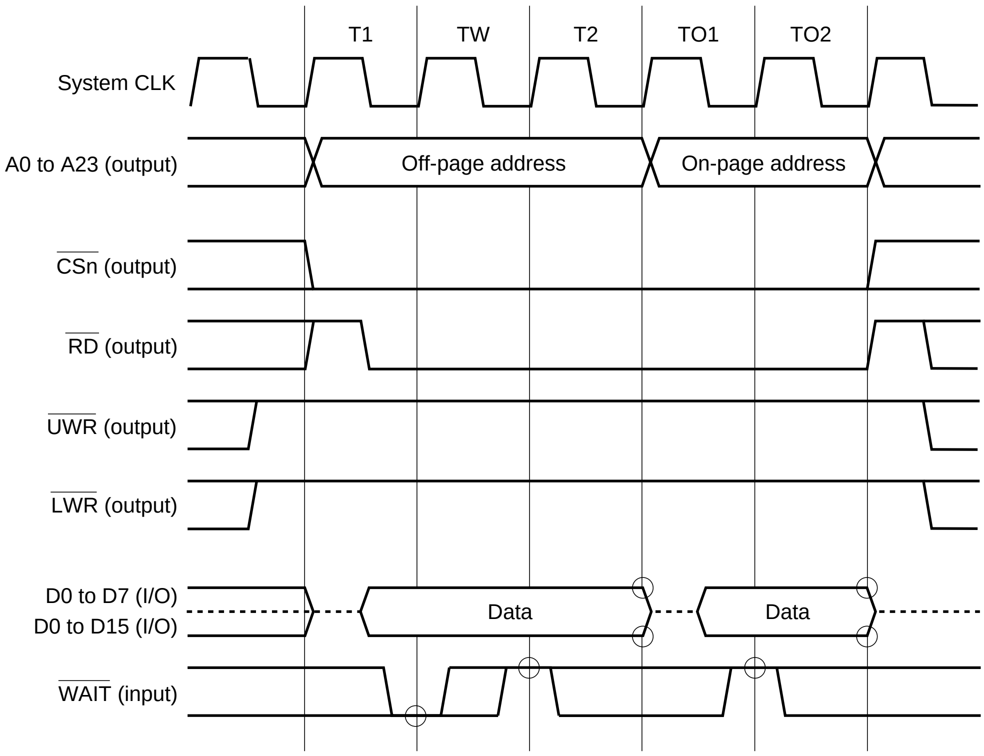
<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14 to 12</td><td>PRW2 to PRW0</td><td>Page-ROM On-page Wait Control
Sets the number of waits corresponding to the internal system clock.
The number of waits set by this bit are inserted only when on-page. When off-page, the waits set by registers DWC0 and DWC1 are inserted.
<table>
<tr><th>PRW2</th><th>PRW1</th><th>PRW0</th><th>Number of Inserted Wait Cycles</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>7</td></tr>
</table></td></tr>
<tr><td>3 to 0</td><td>MA6 to MA3</td><td>Mask Address
Each respective address (A6 to A3) corresponding to MA6 to MA3 is masked (masked by 1). The masked address is not subject to comparison during on/off-page judgment.
It is set according to the number of continuously readable bits.
<table>
<tr><th>MA6</th><th>MA5</th><th>MA4</th><th>MA3</th><th>Number of Continuously Readable Bits</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>4 words × 16 bits (8 words × 8 bits)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>8 words × 16 bits (16 words × 8 bits)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>16 words × 16 bits (32 words × 8 bits)</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>32 words × 16 bits (64 words × 8 bits)</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>64 words × 16 bits (128 words × 8 bits)</td></tr>
</table></td></tr>
</table>

Caution: Write to the PRC register after reset, and then do not change the set value. Also, do not access an external memory area other than that for this initialization routine until initial setting of the PRC register is finished. However, it is possible to access external memory areas whose initialization has been finished.

5.2.5 Page ROM access

Figure 5-6: Page ROM Access Timing (1/4)

(a) During read (when half word/word access with 8-bit bus width or when word access with 16-bit bus width)



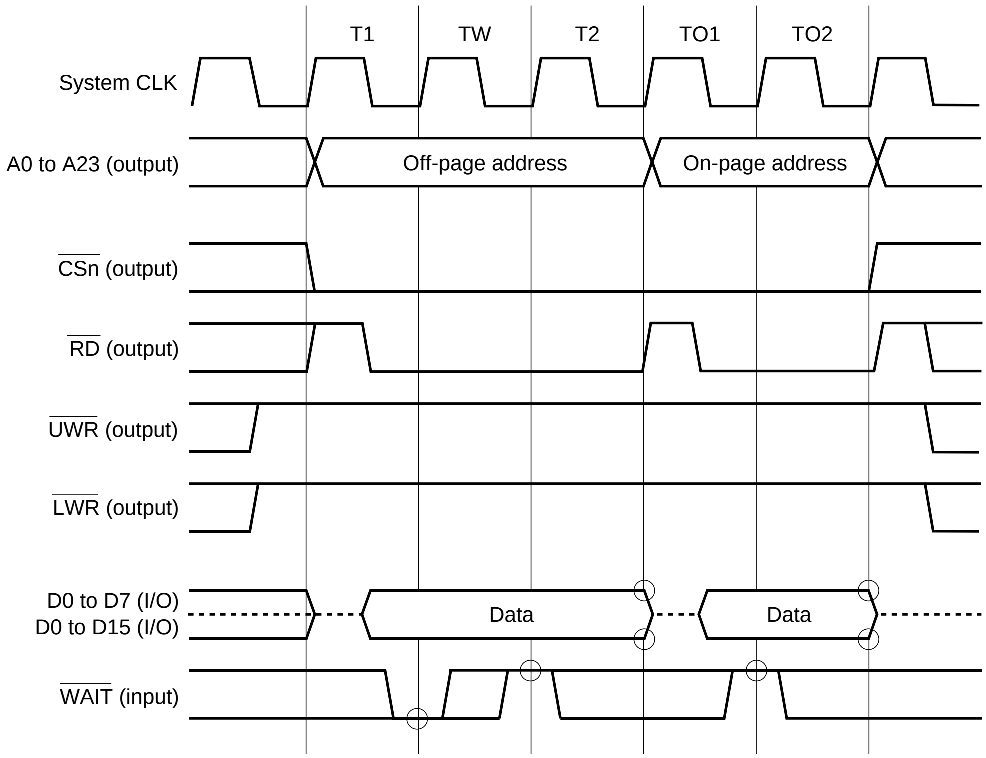
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

Figure 5-6: Page ROM Access Timing (2/4)

(b) During read (when byte access with 8-bit bus width or when byte/half word access with 16-bit bus width)



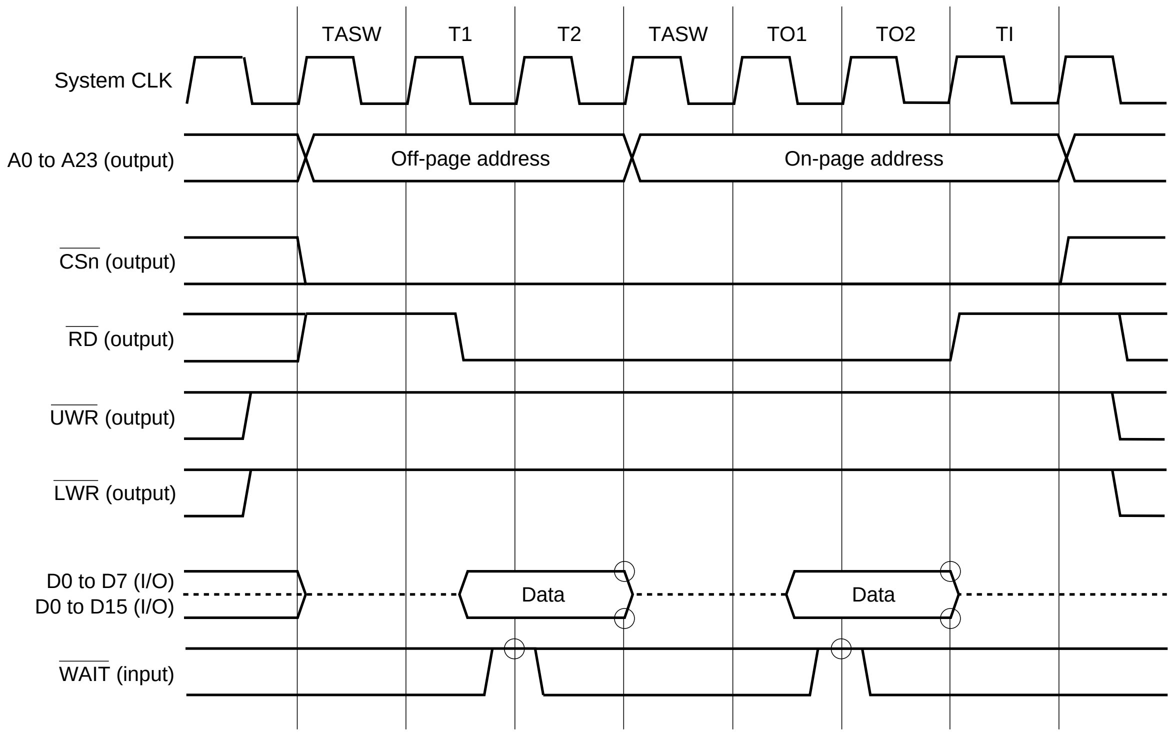
Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

Figure 5-6: Page ROM Access Timing (3/4)

(c) During read (address setup wait, idle state insertion)
(when half word/word access with 8-bit bus width or when word access with 16-bit bus width)



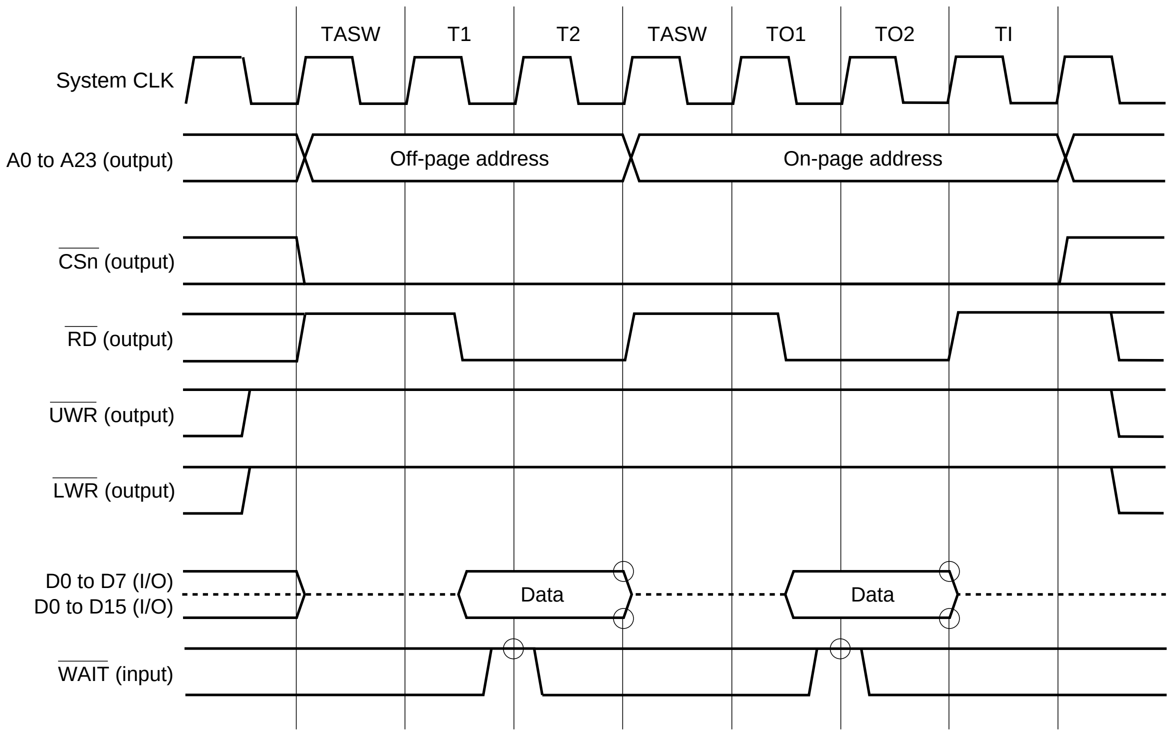
Remarks: **1.** The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

Figure 5-6: Page ROM Access Timing (4/4)

(d) During read (address setup wait, idle state insertion)
(when byte access with 8-bit bus width or when byte/half word access with 16-bit bus width)



Remarks: 1. The circles ❍ indicate the sampling timing.

2. The broken line indicates the high-impedance state.

3. $\overline{\text{CSn}}$ = $\overline{\text{CS0}}$, $\overline{\text{CS3}}$ and $\overline{\text{CS4}}$

[MEMO]

Chapter 6 Instruction Cache

The V850E/CA2 Jupiter device contains a 4 KByte 2-way-associative instruction memory (iCache) to improve the system's instruction execution speed and performance.

6.1 Features

- Use of Least Recently Used (LRU) algorithm.
  The LRU algorithm replaces the cache line, that has not been used since the longest time.
  The probability of a instruction cache hit is mostly higher compared to the direct mapped type.

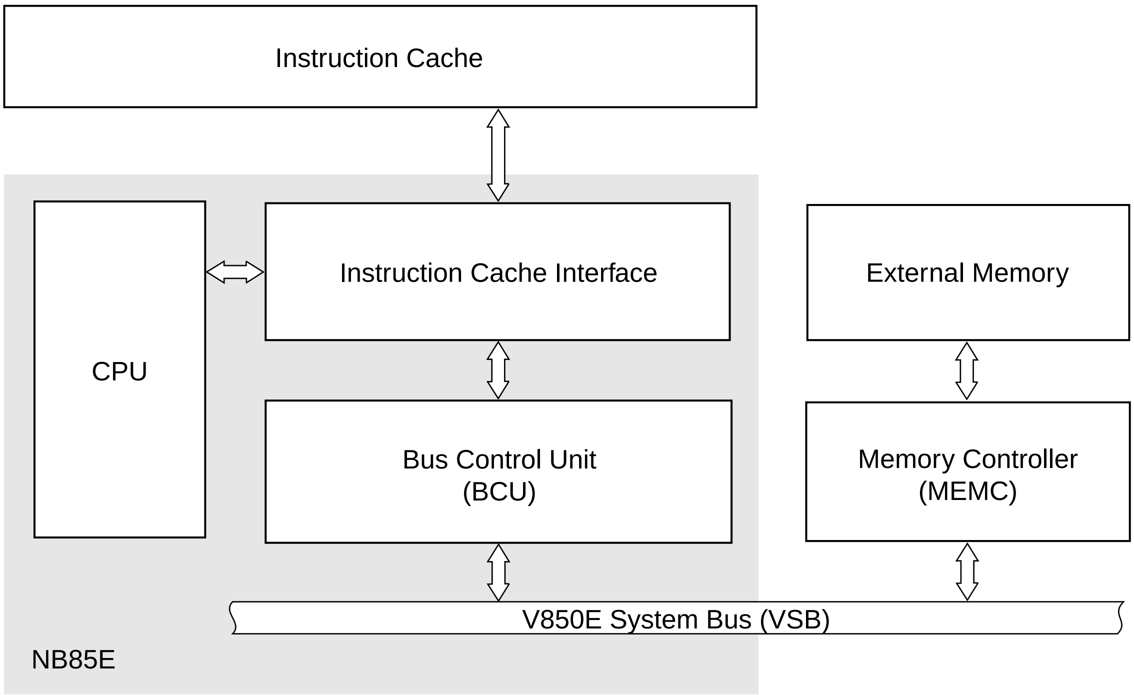
- Using the tag clear function, the contents of all tags can be cleared (invalidated).

- Using the autofill function, instructions for one way can be filled automatically (way 0 only). A filled way is locked automatically, and replacing data in the way or writing to tags is disabled. Thus, it also can be used as a ROM that can operate in one cycle.

6.2 Configuration

To improve the instruction execution speed and the total system's performance, the V850E/CA2 Jupiter device provides a 4 KByte 2-way associative instruction cache memory. The instruction cache is organized as 4 words x 128 entries x 2 ways.

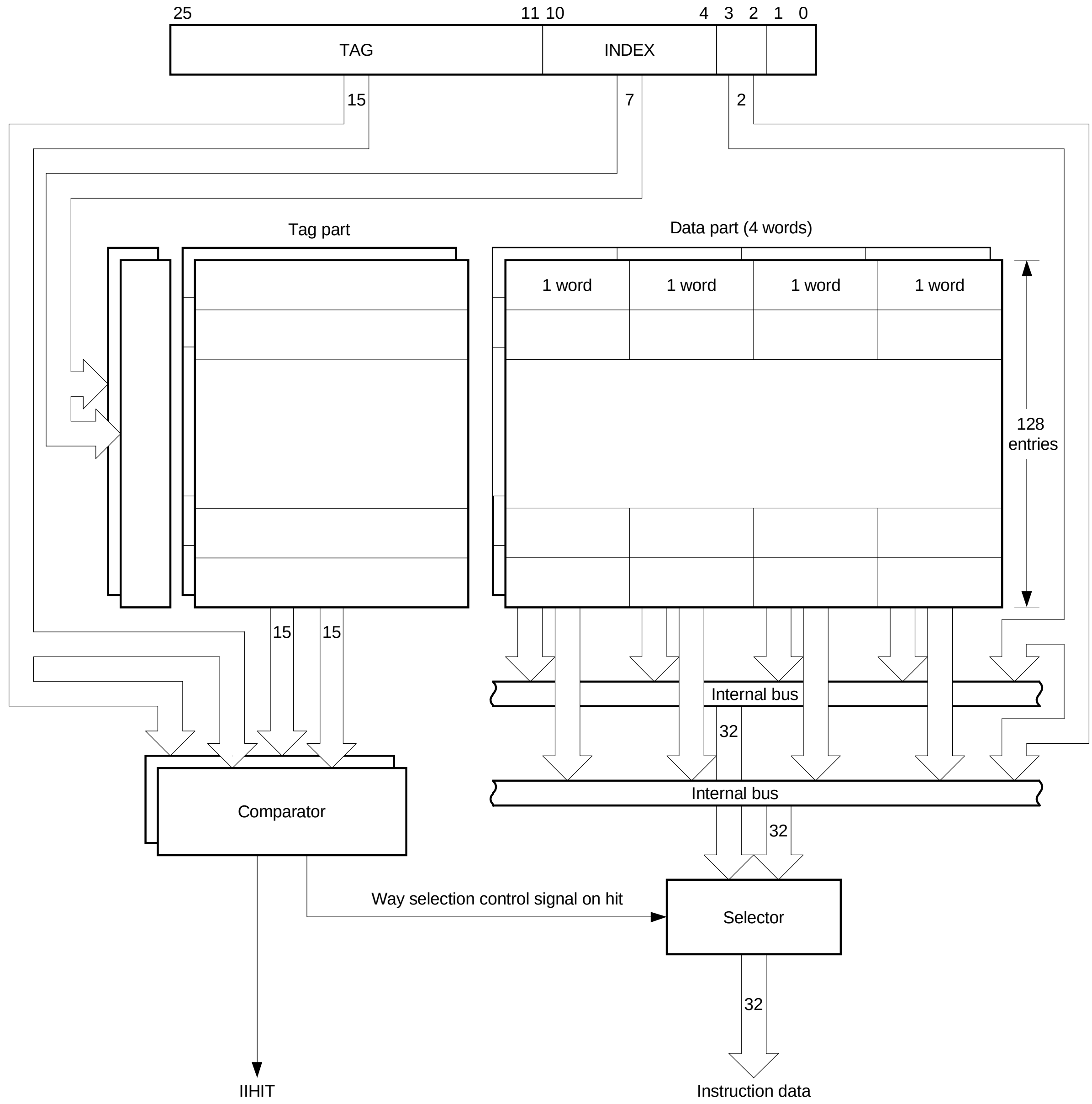
Figure 6-1: Instruction Cache Configuration



6.2.1 Four Kbytes 2-way set-associative Instruction Cache

The data memory of a 4 KBytes 2-way set-associative instruction cache has two ways, each consisting of a block of 128 entries of 4 words per line, for a total capacity of 4 KB.

Figure 6-2: Configuration of 4 KB 2-Way Set-Associative Instruction Cache



6.3 Control Registers

(1) Instruction Cache Control Register (ICC)

The ICC register is the register that sets two types of functions, tag clear and autofill. The ICC register can be read or written in 16-bit, 8-bit or 1-bit units.

Figure 6-3: Instruction Cache Control Register (ICC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ICC	0	0	0	LOCK0	0	0	0	0	0	0	0	FILL0	0	0	TCLR1	TCLR0	FFFFF070H	03H

Bit Position	Bit Name	Function
12	LOCK0	This bit shows the cache lock status of way 0. When way 0 is filled, the cache is locked and this bit is set (1) automatically. Clearing (0) this bit releases the cache lock of way 0. 0: Way 0 is not locked 1: Way 0 is locked
4	FILL0	This bit sets way 0 autofill. Setting (1) this bit autofills way 0. When autofill is complete, this bit is cleared (0) automatically. 0: Way 0 fill complete 1: Way 0 fill operating
1	TCLR1	This bit sets way 1 tag clear. Setting (1) this bit clears (invalidates) way 1 tags. When tag clear is complete, this bit is cleared (0) automatically. 0: Way 1 tag clear complete 1: Way 1 tag clear operating
0	TCLR0	This bit sets way 0 tag clear. Setting (1) this bit clears (invalidates) way 0 tags. When tag clear is complete, this bit is cleared (0) automatically. 0: Way 0 tag clear complete 1: Way 0 tag clear operating

Note: During reset active, the value of this register becomes 0003H, and tag initialization begins automatically. Upon completion of tag initialization, the value changes to 0000H.

Cautions:
1. **If any of bits 0, 1, or 4 is set, do not forcibly clear that bit.**
2. **Do not set bit 4 at the same time as the other bits.**
3. **Do not set bit 12. Bit 12 can only be cleared.**
4. **Make ICC register settings running code from an uncacheable area (except for setting bit 4).**

(2) Instruction Cache Data Configuration Register (ICD)

The ICD register sets the address of the memory area to be autofilled when using the autofill function. The ICD register can be read or written in 16-bit units.

Figure 6-4: Instruction Cache Data Configuration Register (ICD)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ICD	0	DATA14	DATA13	DATA12	DATA11	DATA10	DATA9	DATA8	DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0	FFFFF074H	Undef.

Bit Position	Bit Name	Function
0 to 14	DATA0 to DATA14	Address of the memory area to be autofilled. These bits specify the address bits 25 to 11 of the start address of the memory area to be autofilled.

Cautions: 1. Do not overwrite the ICD register while autofill is operating.

2. Since the initial value of the ICD register is undefined, when using the autofill function, be sure to set a value in the ICD register prior to setting the FILL0 bit of the ICC register. If the FILL0 bit of the ICC register is set without setting a value in the ICD register, the operation cannot be guaranteed.

(3) Instruction Cache Initial Register (ICI)

The ICI register controls the iCache operation by the MODE bit. The ICI register can be accessed by 16-bit access.

Figure 6-5: Instruction Cache Initial Register (ICI)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ICI	0	MODE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	FFFFF072H	FFH

Bit Position	Bit Name	Function
14	MODE	Instruction Cache Operation Mode Control bit.

Caution: All bits of the ICI register must be cleared immediately following a system reset. The following procedure must be executed to clear the ICI register:

Example: st.h r0, 0xfffff072[r0]

6.4 Instruction Cache Operation

(1) Instruction Cache Basic Operation

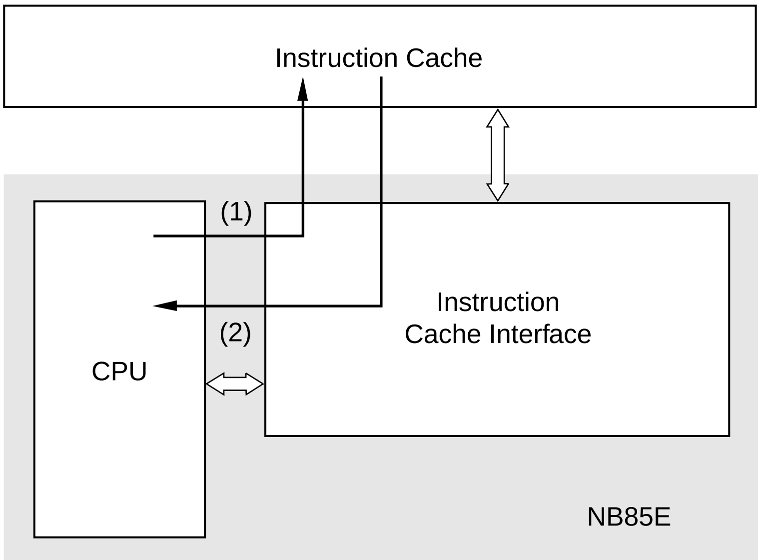
The instruction cache automatically performs a caching operation whenever there is a fetch access to a cacheable area set using the cache configuration register BHC.

(2) Operation on Instruction Cache Hit

(1) On a fetch access from memory, the CPU outputs the instruction fetch request and the concerned address to the instruction cache.

(2) If a hit occurs due to the address existing in the instruction cache, the instruction data is read from the instruction cache and passed through to the CPU.

Figure 6-6: Operation on Instruction Cache Hit



(3) Operation on Instruction Cache Miss

(1) On a fetch access from memory, the CPU outputs the instruction fetch request and the concerned address to the instruction cache.

(2) If an instruction cache miss occurs due to the address not existing in the instruction cache, the fetch request and the address will be output from the instruction cache to the BCU.

(3) The BCU then outputs the address to external memory via the VSB and refills the instruction cache with one line (4 words) at the address to be read.

(4) The instruction cache then transfers the data to be read among the 4 words of refill data to the CPU.

Caution: The miss penalty time when a miss occurs varies depending on such things as memory controller specifications for external memory and VSB bus cycle wait insertion time.

Figure 6-7: Operation on Instruction Cache Miss

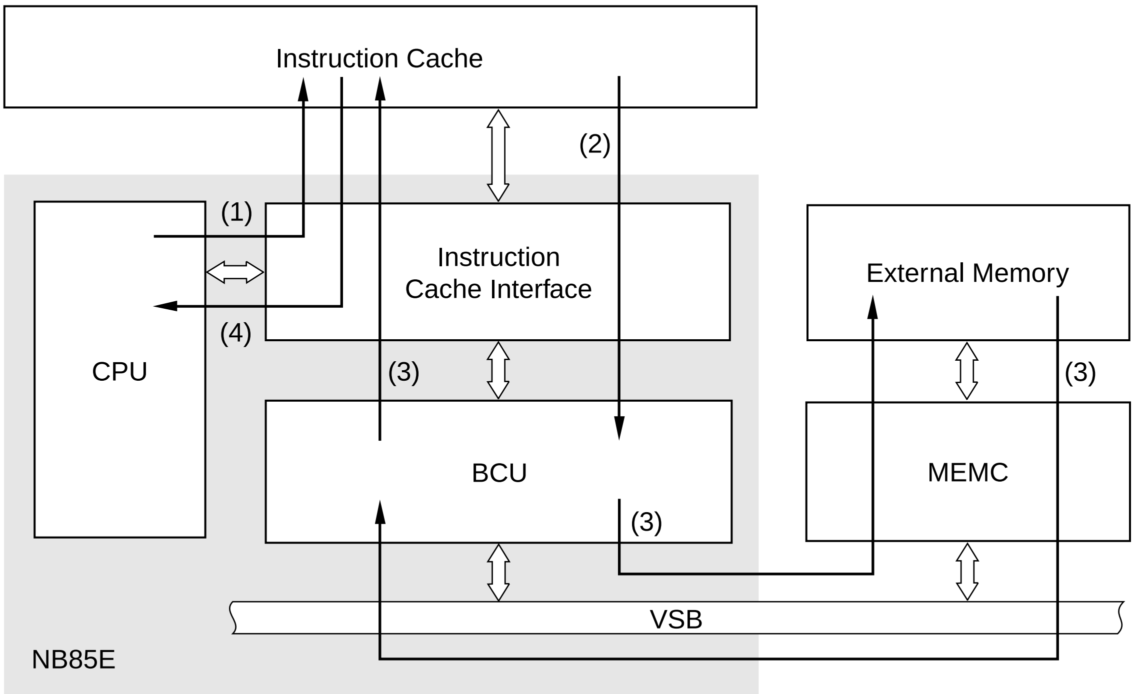
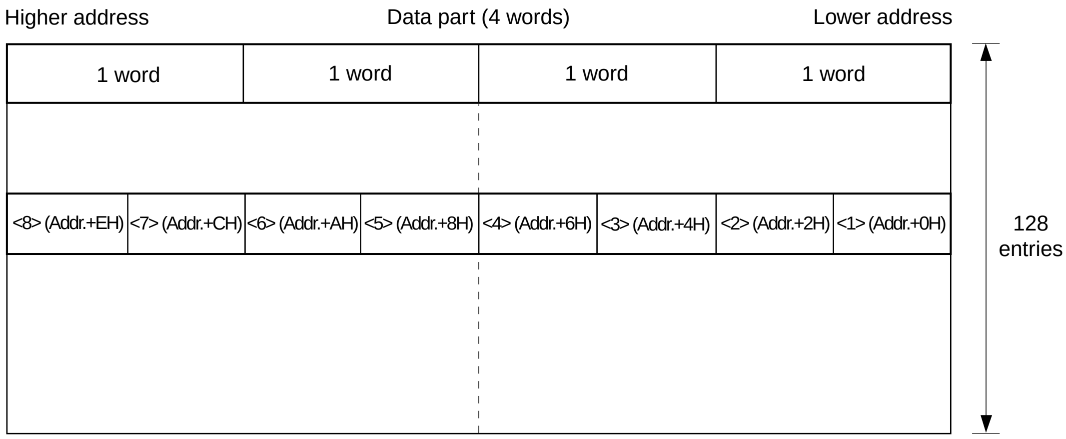


Figure 6-8: Refill Sequence to Instruction Cache (16-bit Data Bus)



Remarks: 1. The numbers within pointed brackets (< >) indicate the refill sequence.

2. (Adrs. +n): Data of address in () (n = 0H to FH)

(4) Tag Clear Function

The tag clear function clears (invalidates) the tags of one way. In addition, it automatically clears (invalidates) the tags of all ways on a system reset. Instruction cache tag clear performs the following procedure:

(1) Read the instruction cache control register (ICC) and confirm that bits 0 and 1 (TCLR0, TCLR1) are all cleared.

(2) Read the ICC register and confirm that bit 12 (LOCK0) is cleared

(3) Set bit TCLR0 or bit TCLR1 of the ICC register as follows:

Cautions: 1. To clear the instruction cache tags, the tag clear operation by setting the bits TCLR0 or TCLR1 of the ICC register must be executed twice.

2. Perform all of (1) to (3) above (tag clear) executing the code in an uncacheable area (tags are not cleared if the above processing is performed by code in a cacheable area).

- **When clearing way 0 and way 1 at the same time:**

(a) Set the TCLR0 and TCLR1 bits.
(b) Read the TCLR0 and TCLR1 bits to confirm that these bits are cleared.
(c) Perform (a) and (b) above again.

- **When clearing way 0 and way 1 individually[Note]:**

(a) Set the TCLR0 bit.
(b) Read the TCLR0 bit to confirm that this bit is cleared.
(c) Perform (a) and (b) above again.
(d) Set the TCLR1 bit.
(e) Read the TCLR1 bit to confirm that this bit is cleared.
(f) Perform (d) and (e) above again.

Note: The setting can also be made in order of (d)-(e)-(f)-(a)-(b)-(c).

3. Way 0 shares the counter to clear tags with way 1.
Therefore a clear tag operation must not be started (set the TCLR0 bit or TCLR1 bit of the ICC register), even if the other way is currently being cleared. When clearing the tags of way 0 and way 1 individually, if tag clearing for either way is executed during tag clear execution for the other way (TCLR0 or TCLR1 = "1"), the counter stops in the middle of tag clearing. Consequently, normal tag clearing cannot be performed because the counter switches to perform the other tag clear operation still indicating the value it had when stopped halfway. Be sure to confirm that tag clearing for one way is completed (TCLR0 or TCLR1 = "0") before performing tag clearing for the other way.
When setting both bits at the same time as shown below, normal tag clearing will be performed properly.

4. Be sure not to perform other processing simultaneously with tag clearing before reading the TCLR0 and TCLR1 bits of the ICC register and confirming that these bits are cleared "0".

Sample Coding:

<1>	mov	0x3, r2	
<2>	LOP0:		
<3>	ld.h	ICC[r0], r1	
<4>	cmp	r0, r1	
<5>	bnz	LOP0	
<6>	st.h	r2, ICC[r0]	
<7>	LOP1:		-- First TAG clear
<8>	ld.h	ICC[r0], r1	
<9>	cmp	r0, r1	
<10>	bnz	LOP1	
<11>	st.h	r2, ICC[r0]	
<12>	LOP2:		-- Second TAG clear
<13>	ld.h	ICC[r0], r1	
<14>	cmp	r0, r1	
<15>	bnz	LOP2	

Remark: The clock count required for a tag clear operation is 256 clocks (To actually clear tags, the required clock count is doubled because a tag clear operation is performed twice sequentially).

Note: During reset active, the value of the bits TCLR0 and TCLR1 becomes set “1” and tag initialization begins automatically. Upon completion of tag initialization, the value of these bits changes to “0”.

(5) Autofill Function (Way 0 only)

The autofill function automatically fills instructions for one way. Once autofilled, a way is automatically locked and write is disabled and it operates the same as a ROM that is accessible in one cycle. When the lock is released, it again operates as an instruction cache.
Instruction cache autofill performs the following procedure:

(1) Clear (invalidate) the tags of way 0 (see “Tag Clear Function” on page 163).

(2) Set the address corresponding to the memory area to be autofilled in the instruction cache data configuration register (ICD).

(3) Branch to the cacheable area corresponding to the tag information set in the ICD register.

(4) Set bit 4 (FILL0) of the instruction cache control register (ICC).

(5) When autofill is complete, bit 12 (LOCK0) of the ICC register is automatically set “1” and the way 0 is locked. At that same time, read bit FILL0 of the ICC register and confirm that bit is cleared “0”.

Remarks: 1. A lock is released by clearing bit LOCK0 of the ICC register.

2. While the iCache autofill operation is ongoing, neither interrupt nor NMI will be served by CPU until the iCache autofill procedure is finished. Even for the situation that a required interrupt service function is by chance already available in the second way of iCache, CPU can not access these opcodes until the autofill operation of the iCache way 0 is completed. Direct opcode execution from the VSB is also not possible in general during the processing time of the autofill operation.

3. Since the autofill operation is performed from the external memory to the instruction cache via the VSB, other processing can be performed at the same time, but only if the processing involves operations within the CPU (processing without any VSB and NPB accesses).

Caution: Run the code to perform the above operations from the memory areas shown below:

- **(1), (2), (3)......... Uncacheable area**
- **(4)..................... Cacheable area. If bit 4 (FILL0) of the ICC register is set using an uncacheable area, autofill cannot be performed (invalid operation).**
- **(5)..................... Either a cacheable area or an uncacheable area**

6.5 Instruction Cache Initialisation

The instruction cache settings must be performed using the following procedure with the initial settings of the user program immediately following a system reset.

(1) Wait until the contents of the ICC register becomes 0000H (TAG initialization is completed).

(2) Clear all bits of the ICI register using the following instruction:

```
st.h    r0, 0xfffff072[r0]
```

(3) Set the ICC and ICD registers

(4) Make the instruction cache settings of the cache configuration register BHC

Caution: Be sure to make the BHC register settings running the code from an uncacheable area (an instruction is not correctly fetched if settings are made using a cacheable area).

6.6 Operating Precautions

(1) Operation on Reset:

At the time of a reset, tags are automatically cleared (invalidated), which puts the next data replacement in a state of being performed from way 0. Therefore, if there is an access to the instruction cache within a period of as many clock cycles as the number of lines after a reset, the CPU stops until the tags are cleared (become valid).

(2) Setting registers:

Be sure to set the registers shown below running the code from an uncacheable area. However, set bit 4 of the instruction cache control register (ICC) using a cacheable area.

- Chip area select control registers (CSC0, CSC1)
- Peripheral I/O area select control register (BPC)
- Bus size configuration register (BSC)
- Endian configuration register (BEC)
- Cache configuration register (BHC)
- Instruction cache control register (ICC[Note])
- Instruction cache data configuration register (ICD)

Note: Excluding bit 4

(3) Initial program settings:

Always execute the following instruction before setting the cache configuration register BHC with the initial settings of the user program immediately following system reset.

```
st.h r0, 0xfffff072[r0]
```

Following execution of this instruction, the cache is enabled by setting “cache enable” (BHn0 bit = “1”) as the instruction cache setting with the BHC register (n = 7 to 0).

(4) Setting BHC register:

In the case of CSn areas for which an instruction to set the BHC register exists in the same CSn area, cache enable/disable settings for the instruction cache using this instruction cannot be performed (n = 7 to 0). Instruction cache enable/disable settings are possible only for those CSn areas in which no instruction for setting the BHC register exists.

For example, if a BHC register setting instruction exists in the CS0 area, the instruction cache of the CS0 area cannot be set (cache enable/disable settings). In this case, only the instruction cache settings for areas CS1 to CS7 are possible.

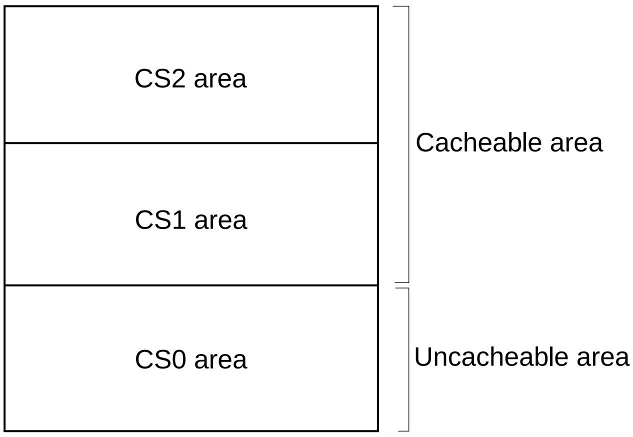
(5) Access to memory boundary:

If adjacent chip select (CSn) areas are a cacheable area and an uncacheable area, continuous access across the memory boundary is possible only by using a branch instruction. Operation is not guaranteed if the memory boundary is continuously accessed by instruction other than a branch instruction. An example is shown below:

Suppose that the cache area settings are as shown in figure “iCache Area Setting Example”. In this case, access to the memory areas is as follows:

- From CS0 area to CS1 area, access is possible only by using a branch instruction.
- From CS1 area to CS2 area, continuous access is possible.

Figure 6-9: iCache Area Setting Example



Chapter 7 DMA Functions (DMA Controller)

The V850E/CA2 includes a direct memory access (DMA) controller (DMAC) that executes and controls DMA transfer.
The DMAC controls data transfer between memory and I/O or among I/Os, based on DMA requests issued by the on-chip peripheral I/O, or software triggers (memory refers to internal RAM).

7.1 Features

- Four independent DMA channels
- Transfer units: 8, 16 and 32 bits
- Maximum transfer count: 65,536 (2^{16})
- Two types of transfer
  - two cycle transfer
- Four transfer modes
  - Single transfer mode
  - Single-step transfer mode
  - Line transfer mode (Four bus cycle transfer mode)
  - Block transfer mode
- Transfer requests
  - Request by interrupts from on-chip peripheral I/O
  - Requests by software trigger
- Transfer objects
  - Between internal RAM and I/O
  - Between internal RAM and external I/O
  - Between internal RAM and internal RAM
  - Between external memory and I/O
  - Between external memory and external memory
  - Between I/O and I/O
- DMA transfer completion flag
- Next address setting function

7.2 Control Registers

7.2.1 DMA source address registers H0 to H3 (DSAH0 to DSAH3)

These registers are used to set the DMA source addresses (28 bits each) for DMA channel n (n = 0 to 3). They are divided into two 16-bit registers, DSAHn and DSALn.
Since these registers are configured as 2-stage FIFO buffer registers, a new source address for DMA transfer can be specified during DMA transfer (refer to **7.3 Next Address Setting Function**).

(1) DMA source address registers DSAH0 to DSAH3 (DSAH0 to DSAH3)

These registers can be read/written in 16-bit units.

Caution: When setting an address of a peripheral I/O register for the source address, be sure to specify an address between FFFF000H and FFFFFFFH. An address of the peripheral I/O register image (3FFF000H to 3FFFFFFH) must not be specified.

Figure 7-1: DMA Source Address Registers DSAH0 to DSAH3 (DSAH0 to DSAH3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH0	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF082H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH1	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF08AH	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH2	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF092H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAH3	IR	0	0	0	SA26	SA26	SA25	SA24	SA23	SA22	SA21	SA20	SA19	SA18	SA17	SA16	FFFFF09AH	undef.

Bit Position	Bit Name	Function
15	IR	Specifies the DMA source address. 0: External memory or On-chip peripheral I/O 1: Internal RAM
11 to 0	SA27 to SA16	Sets the DMA source addresses (A27 to A16). During DMA transfer, it stores the next DMA transfer source address.

(2) DMA source address registers L0 to L3 (DSAL0 to DSAL3)

These registers can be read/written in 16-bit units.

Figure 7-2: DMA Source Address Registers DSAL0 to DSAL3 (DSAL0 to DSAL3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL0	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF080H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL1	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF088H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL2	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF090H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSAL3	SA15	SA14	SA13	SA12	SA11	SA10	SA9	SA8	SA7	SA6	SA5	SA4	SA3	SA2	SA1	SA0	FFFFF098H	undef.

Bit Position	Bit Name	Function
15 to 0	SA15 to SA0	Sets the DMA source address (A15 to A0). During DMA transfer, it stores the next DMA transfer source address.

7.2.2 DMA destination address registers H0 to H3 (DDAH0 to DDAH3)

These registers are used to set the DMA destination address (28 bits each) for DMA channel n (n = 0 to 3). They are divided into two 16-bit registers, DDAHn and DDALn.
Since these registers are configured as 2-stage FIFO buffer registers, a new destination address for DMA transfer can be specified during DMA transfer (refer to **7.3 Next Address Setting Function**).

(1) DMA destination address registers DDAH0 to DDAH3 (DDAH0 to DDAH3)

These registers can be read/written in 16-bit units.

Caution: When setting an address of a peripheral I/O register for the destination address, be sure to specify an address between FFFF000H and FFFFFFFH. An address of the peripheral I/O register image (3FFF000H to 3FFFFFFH) must not be specified.

Figure 7-3: DMA Destination Address Registers 0H to 3H (DDA0H to DDA3H)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH0	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF086H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH1	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF08EH	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH2	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF096H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAH3	IR	0	0	0	DA27	DA26	DA25	DA24	DA23	DA22	DA21	DA20	DA19	DA18	DA17	DA16	FFFFF09EH	undef.

Bit Position	Bit Name	Function
15	IR	Specifies the DMA destination address. 0: External memory or On-chip peripheral I/O 1: Internal RAM
11 to 0	DA27 to DA16	Sets the DMA destination addresses (A27 to A16). During DMA transfer, it stores the next DMA transfer destination address.

(2) DMA destination address registers L0 to L3 (DDAL0 to DDAL3)

These registers can be read/written in 16-bit units.

Figure 7-4: DMA Destination Address Registers L0 to L3 (DDAL0 to DDAL3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL0	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF084H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL1	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF08CH	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL2	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF094H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDAL3	DA15	DA14	DA13	DA12	DA11	DA10	DA9	DA8	DA7	DA6	DA5	DA4	DA3	DA2	DA1	DA0	FFFFF09CH	undef.

Bit Position	Bit Name	Function
15 to 0	DA15 to DA0	Sets the DMA destination address (A15 to A0). During DMA transfer, it stores the next DMA transfer destination address.

7.2.3 DMA transfer count registers 0 to 3 (DBC0 to DBC3)

These 16-bit registers are used to set the transfer counts for DMA channels n. They store the remaining transfer counts during DMA transfer.
Since these registers are configured as 2-stage FIFO buffer registers, a new DMA transfer count for DMA transfer can be specified during DMA transfer (refer to **7.3 Next Address Setting Function**).
During DMA transfer these registers are decremented by 1 for each transfer that is performed. DMA transfer is terminated when an underflow occurs (from 0 to FFFFH). On terminal count these registers are rewritten with the value that was set immediately before.
These registers can be read/written in 16-bit units.

Figure 7-5: DMA Transfer Count Registers 0 to 3 (DBC0 to DBC3)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC0	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C0H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC1	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C2H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC2	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C4H	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DBC3	BC15	BC14	BC13	BC12	BC11	BC10	BC9	BC8	BC7	BC6	BC5	BC4	BC3	B2C	BC1	BC0	FFFFF0C6H	undef.

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 0</td><td>BC15 to BC0</td><td>Sets the transfer count. It stores the remaining transfer count during DMA transfer.
<table>
<tr><th>DBCn</th><th>States</th></tr>
<tr><td>0000H</td><td>Transfer count 1 or remaining transfer count</td></tr>
<tr><td>0001H</td><td>Transfer count 2 or remaining transfer count</td></tr>
<tr><td>:</td><td>:</td></tr>
<tr><td>FFFFH</td><td>Transfer count 65,536 (2^{16}) or remaining transfer count</td></tr>
</table></td></tr>
</table>

Cautions: 1. In case of a line transfer and the setting of the DBCn register is 0003H (four transfers), one line transfer is applied.

2. For a setting of the DBCn register in which the transfer count cannot be divided by 4, the sections that can be line transferred are (line) transferred first, then the remaining indivisible sections are transferred as single transfer.

Remark: n = 0 to 3

7.2.4 DMA addressing control registers 0 to 3 (DADC0 to DADC3)

These 16-bit registers are used to control the DMA transfer modes for DMA channel n.
They can be read/written in 16-bit units.

Figure 7-6: DMA Addressing Control Registers 0 to 3 (DADC0 to DADC3) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC0	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D0H	0000H
DADC1	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D2H	0000H
DADC2	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D4H	0000H
DADC3	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	TM1	TM0	0	0	FFFFF0D6H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 14</td><td>DS1, DS0</td><td>Sets the transfer data size for DMA transfer.
<table>
<tr><th>DS1</th><th>DS0</th><th>Transfer Data Size</th></tr>
<tr><td>0</td><td>0</td><td>8 bits</td></tr>
<tr><td>0</td><td>1</td><td>16 bits</td></tr>
<tr><td>1</td><td>0</td><td>32 bits</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table>
For the peripheral I/O and programmable peripheral I/O registers, ensure the transfer size matches the access size.</td></tr>
<tr><td>7, 6</td><td>SAD1, SAD0</td><td>Sets the count direction of the source address for DMA channel n (n = 0 to 3).
<table>
<tr><th>SAD1</th><th>SAD0</th><th>Count Direction</th></tr>
<tr><td>0</td><td>0</td><td>Increment</td></tr>
<tr><td>0</td><td>1</td><td>Decrement</td></tr>
<tr><td>1</td><td>0</td><td>Fixed</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
</table></td></tr>
</table>

Figure 7-6: DMA Addressing Control Registers 0 to 3 (DADC0 to DADC3) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th colspan="3">Function</th></tr>
<tr><td rowspan="6">5, 4</td><td rowspan="6">DAD1,
DAD0</td><td colspan="3">Sets the count direction of the destination address for DMA channel n (n = 0 to 3).</td></tr>
<tr><td>DAD1</td><td>DAD0</td><td>Count Direction</td></tr>
<tr><td>0</td><td>0</td><td>Increment</td></tr>
<tr><td>0</td><td>1</td><td>Decrement</td></tr>
<tr><td>1</td><td>0</td><td>Fixed</td></tr>
<tr><td>1</td><td>1</td><td>Setting prohibited</td></tr>
<tr><td rowspan="6">3, 2</td><td rowspan="6">TM1, TM0</td><td colspan="3">Sets the transfer mode during DMA transfer.</td></tr>
<tr><td>TM1</td><td>TM0</td><td>Transfer Mode</td></tr>
<tr><td>0</td><td>0</td><td>Single transfer mode</td></tr>
<tr><td>0</td><td>1</td><td>Single-step transfer mode</td></tr>
<tr><td>1</td><td>0</td><td>Line transfer mode</td></tr>
<tr><td>1</td><td>1</td><td>Block transfer mode</td></tr>
</table>

Caution: These registers cannot be accessed during DMA operation.

Remark: n = 0 to 3

7.2.5 DMA channel control registers 0 to 3 (DCHC0 to DCHC3)

These 8-bit registers are used to control the DMA transfer operating mode for DMA channel n.
These registers can be read/written in 8-bit or 1-bit units. (However, bit 7 is read only and bits 2 and 1 are write only. If bits 2 and 1 are read, the read value is always 0.)

Figure 7-7: DMA Channel Control Registers 0 to 3 (DCHC0 to DCHC3)

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC0	TC0	0	0	0	MLE0	INIT0	STG0	EN0	FFFFF0E0H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC1	TC1	0	0	0	MLE1	INIT1	STG1	EN1	FFFFF0E2H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC2	TC2	0	0	0	MLE	INIT	STG	EN2	FFFFF0E4H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
DCHC3	TC3	0	0	0	MLE	INIT	STG	EN3	FFFFF0E6H	00H

Bit Position	Bit Name	Function
7	TCn	This status bit indicates whether DMA transfer through DMA channel n has ended or not. It is read-only, and is set to 1 when DMA transfer ends and cleared (0) when it is read. 0: DMA transfer had not ended. 1: DMA transfer had ended.
3	MLEn	When this bit is set to 1 at terminal count output, the Enn bit is not cleared to 0 and the DMA transfer enable state is retained. Moreover, the next DMA transfer request can be accepted even when the TCn bit is not read. When this bit is cleared to 0 at terminal count output, the Enn bit is cleared to 0 and the DMA transfer disable state is entered. At the next DMA request, the setting of the Enn bit to 1 and the reading of the TCn bit are required.
2	INITn	When this bit is set to 1, DMA transfer is forcibly terminated.
1	STGn	If this bit is set to 1 in the DMA transfer enable state (TCn bit = 0, Enn bit = 1), DMA transfer is started.
0	ENn	Specifies whether DMA transfer through DMA channel n is to be enabled or disabled. This bit is cleared to 0 when DMA transfer ends. It is also cleared to 0 when DMA transfer is forcibly terminated by means of setting the INITn bit to 1 or by NMI input. 0: DMA transfer disabled 1: DMA transfer enabled

Remark: n = 0 to 3

7.2.6 DMA disable status register (DDIS)

This register holds the contents of the ENn bit of the DCHCn register during NMI input.
This register is read-only in 8-bit or 1-bit units.

Figure 7-8: DMA Disable Status Register (DDIS)

	7	6	5	4	3	2	1	0	Address	Initial value
DDIS	0	0	0	0	CH3	CH2	CH1	CH0	FFFFF0F0H	00H

Bit Position	Bit Name	Function
3 to 0	CH3 to CH0	Reflects the contents of the ENn bit of the DCHCn register during NMI input. The contents of this register are held until the next NMI input or until the system is reset.

Remark: n = 0 to 3

7.2.7 DMA restart register (DRST)

This register is used to restart DMA transfer that has been forcibly interrupted by a non-maskable interrupt (NMI). The ENn bit of this register and the ENn bit of the DCHCn register are linked to each other. Following forcible interrupt by NMI input, the DMA channel that was interrupted is confirmed from the contents of the DDIS register, and DMA transfer is restarted by setting the ENn bit of the corresponding channel to 1.
This register can be read/written in 8-bit or 1-bit units.

Figure 7-9: DMA Restart Register (DRST)

	7	6	5	4	3	2	1	0	Address	Initial value
DRST	0	0	0	0	EN3	EN2	EN1	EN0	FFFFF0F2H	00H

Bit Position	Bit Name	Function
3 to 0	EN3 to EN0	Specifies whether DMA transfer through DMA channel n is to be enabled or disabled. This bit is cleared to 0 when DMA transfer is completed in accordance with the terminal count output. It is also cleared to 0 when DMA transfer is forcibly terminated by setting the INITn bit to 1 or by NMI input. 0: DMA transfer disabled 1: DMA transfer enabled

Remark: n = 0 to 3

7.2.8 DMA trigger factor register 0 (DTFR0)

This 8-bit registers is used to control the DMA transfer start trigger of DMA channel 0 through interrupt requests from on-chip peripheral I/O. The interrupt requests set with these registers serve as DMA transfer start factors.
This register can be read/written in 8-bit/1-bit units.

Figure 7-10: DMA Trigger Factor Registers 0 (DTFR0)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR0	DRQ	DOFL	0	0	0	IFC2	IFC1	IFC0	FFFFF840H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7</td><td>DRQ[Note]</td><td></td></tr>
<tr><td>6</td><td>DOFL[Note]</td><td></td></tr>
<tr><td>3 to 0</td><td>IFC3 to IFC0</td><td>Sets the interrupt source that serves as the DMA start factor.
<table>
<tr><th>IFC2</th><th>IFC1</th><th>IFC0</th><th>Interrupt Source</th><th>Peripheral Source</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>INTIN1</td><td>CSI0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>INTIN2</td><td>CSI1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>INTIN3</td><td>CSI2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>INTIN4</td><td>UART0, Reception</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>INTIN6</td><td>UART1, Reception</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>INTIN8</td><td>ADC</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>INTIN9</td><td>TMG0.1</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>INTIN10</td><td>TMG1.1</td></tr>
</table>
</td></tr>
</table>

Note: DRQ and DOFL are set by hardware and reset by software. Setting these bits by software is not possible. A “0” must be written to the respective bit location to reset DRQ or DOFL.

Cautions: 1. Be sure to stop DMA operation before making changes to DTFR0 register settings.

2. An interrupt request input in a standby mode (IDLE or software STOP mode) cannot be used as a DMA transfer start factor.

7.2.9 DMA trigger factor register 1 (DTFR1)

This 8-bit registers is used to control the DMA transfer start trigger of DMA channel 1 through interrupt requests from on-chip peripheral I/O. The interrupt requests set with these registers serve as DMA transfer start factors.
This register can be read/written in 8-bit/1-bit units.

Figure 7-11: DMA Trigger Factor Registers 1 (DTFR1)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR1	DRQ	DOFL	0	0	0	IFC2	IFC1	IFC0	FFFFF842H	00H

Bit Position	Bit Name	Function
7	DRQ Note	
6	DOFL Note	
3 to 0	IFC3 to IFC0	Sets the interrupt source that serves as the DMA start factor.

IFC2	IFC1	IFC0	Interrupt Source	Peripheral Source
0	0	0	INTIN1	CSI0
0	0	1	INTIN2	CSI1
0	1	0	INTIN3	CSI2
0	1	1	INTIN4	UART0, Reception
1	0	0	INTIN6	UART1, Reception
1	0	1	INTIN8	ADC
1	1	0	INTIN11	TMG0.2
1	1	1	INTIN12	TMC.0

Note: DRQ and DOFL are set by hardware and reset by software. Setting these bits by software is not possible. A “0” must be written to the respective bit location to reset DRQ or DOFL.

Cautions: 1. Be sure to stop DMA operation before making changes to DTFR1 register settings.

2. An interrupt request input in a standby mode (IDLE or software STOP mode) cannot be used as a DMA transfer start factor.

7.2.10 DMA trigger factor register 2 (DTFR2)

This 8-bit registers is used to control the DMA transfer start trigger of DMA channel 2 through interrupt requests from on-chip peripheral I/O. The interrupt requests set with these registers serve as DMA transfer start factors.
This register can be read/written in 8-bit/1-bit units.

Figure 7-12: DMA Trigger Factor Registers 2 (DTFR2)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR2	DRQ	DOFL	0	0	0	IFC2	IFC1	IFC0	FFFFF844H	00H

Bit Position	Bit Name	Function
7	DRQ[Note]	
6	DOFL[Note]	
3 to 0	IFC3 to IFC0	Sets the interrupt source that serves as the DMA start factor.

IFC2	IFC1	IFC0	Interrupt Source	Peripheral Source
0	0	0	INTIN1	CSI0
0	0	1	INTIN2	CSI1
0	1	0	INTIN3	CSI2
0	1	1	INTIN5	UART0, Transmission
1	0	0	INTIN7	UART1, Transmission
1	0	1	INTIN8	ADC
1	1	0	INTIN13	TMG0.3
1	1	1	INTIN14	TMG1.3

Note: DRQ and DOFL are set by hardware and reset by software. Setting these bits by software is not possible. A “0” must be written to the respective bit location to reset DRQ or DOFL.

Cautions: 1. Be sure to stop DMA operation before making changes to DTFR2 register settings.

2. An interrupt request input in a standby mode (IDLE or software STOP mode) cannot be used as a DMA transfer start factor.

7.2.11 DMA trigger factor register 3 (DTFR3)

This 8-bit registers is used to control the DMA transfer start trigger of DMA channel 3 through interrupt requests from on-chip peripheral I/O. The interrupt requests set with these registers serve as DMA transfer start factors.
This register can be read/written in 8-bit/1-bit units.

Figure 7-13: DMA Trigger Factor Registers 3 (DTFR3)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR3	DRQ	DOFL	0	0	0	IFC2	IFC1	IFC0	FFFFF846H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7</td><td>DRQ[Note]</td><td></td></tr>
<tr><td>6</td><td>DOFL[Note]</td><td></td></tr>
<tr><td>3 to 0</td><td>IFC3 to IFC0</td><td>Sets the interrupt source that serves as the DMA start factor.
<table>
<tr><th>IFC2</th><th>IFC1</th><th>IFC0</th><th>Interrupt Source</th><th>Peripheral Source</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>INTIN1</td><td>CSI0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>INTIN2</td><td>CSI1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>INTIN3</td><td>CSI2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>INTIN5</td><td>UART0, Transmission</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>INTIN7</td><td>UART1, Transmission</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>INTIN8</td><td>ADC</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>INTIN15</td><td>TMG0.4</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>INTIN16</td><td>TMC1</td></tr>
</table></td></tr>
</table>

Note: DRQ and DOFL are set by hardware and reset by software. Setting these bits by software is not possible. A “0” must be written to the respective bit location to reset DRQ or DOFL.

Cautions: 1. Be sure to stop DMA operation before making changes to DTFR3 register settings.

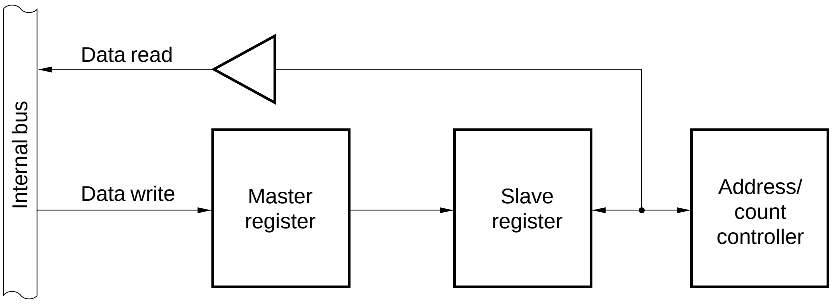
2. An interrupt request input in a standby mode (IDLE or software STOP mode) cannot be used as a DMA transfer start factor.

7.3 Next Address Setting Function

The DMA source address registers (DSAHn, DSALn), DMA destination address registers (DDAHn, DDALn), and DMA transfer count register (DBCn) are buffer registers with a 2-stage FIFO configuration. When the terminal count is issued, these registers are automatically rewritten with the value that was set immediately before.
Therefore, during DMA transfer, transfer is automatically started when a new DMA transfer setting is made for these registers and the MLEn bit of the DCHCn register is set (however, the DMA transfer end interrupt may be issued even if DMA transfer is automatically started).

Figure 7-14, “Buffer Register Configuration,” on page 183 shows the configuration of the buffer register.

Figure 7-14: Buffer Register Configuration



Remark: n = 0 to 3

7.4 DMA Bus States

7.4.1 Types of bus states

The DMAC bus states consist of the following 13 states.

(1) TI state

The TI state is an idle state, during which no access request is issued.

(2) T0 state

This is the DMA transfer ready state (state in which a DMA transfer request has been issued and the bus mastership is acquired for the first DMA transfer).

(3) T1R state

The bus enters the T1R state at the beginning of a read operation in the two-cycle transfer mode. Address driving starts. After entering the T1R state, the bus invariably enters the T2R state.

(4) T1RI state

This is a state in which the DMAC is awaiting an acknowledge signal for an external memory read request. After the last T1RI state, the DMAC always transitions to the T2R state.

(5) T2R state

The T2R state corresponds to the last state of a read operation in the two-cycle transfer mode, or to a wait state.
In the last T2R state, read data is sampled. After entering the last T2R state, the bus invariably enters the T1W state.

(6) T2RI state

State in which the bus is ready for DMA transfer to on-chip peripheral I/O or internal RAM (state in which the bus mastership is acquired for DMA transfer to on-chip peripheral I/O or internal RAM). After entering the last T2RI state, the bus invariably enters the T1W state.

(7) T1W state

The bus enters the T1W state at the beginning of a write operation in the two-cycle transfer mode. Address driving starts. After entering the T1W state, the bus invariably enters the T2W state.

(8) T1WI state

This is a state in which the DMAC is awaiting an acknowledge signal for an external memory write request. After the last T1WI state, the DMAC always transitions to the T2W state.

(9) T2W state

The T2W state corresponds to the last state of a write operation in the two-cycle transfer mode, or to a wait state.
In the last T2W state, the write strobe signal is made inactive.

(10) T1FH state

This is the basic state of a flyby[Note] transfer and is the execution cycle of that transfer. After the T1FH state, the DMAC transitions to the T2FH state.

(11) T1FHI state

This is the last state of a flyby[Note] transfer and the DMAC is awaiting the end of the transfer. After the T1FHI start, the bus is released and the DMAC transitions to the TE state

(12) T2FH state

This is the state in which the DMAC judges whether or not to continue flyby[Note] transfers.
If the next transfer is executed in block transfer mode, the DMAC moves to the T1FH state after the T2FH state. In other modes, if a wait has occurred, the DMAC transitions to the T1FHI state. If no wait has occurred, the bus is released and the DMAC transitions to the TE state.

(13) TE state

The TE state corresponds to DMA transfer completion. The DMAC generates the internal DMA transfer completion signal and various internal signals are initialized. After entering the TE state, the bus invariably enters the TI state.

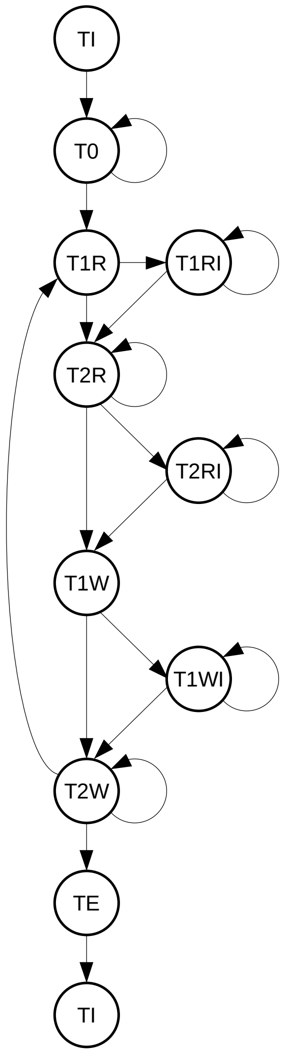
Note: The Flyby transfer mode is not supported on Jupiter.

7.4.2 DMAC bus cycle state transition

Except for the block transfer mode, each time the processing for a DMA transfer is completed, the bus mastership is released.

Figure 7-15: DMAC Bus Cycle State Transition Diagram

(a) Two-cycle transfer



7.5 Transfer Mode

7.5.1 Single transfer mode

In single transfer mode, the DMAC releases the bus at each byte/halfword/word transfer. If there is a subsequent DMA transfer request, transfer is performed again once. This operation continues until a terminal count occurs.
When the DMAC has released the bus, if another higher priority DMA transfer request is issued, the higher priority DMA request always takes precedence. However, if a lower priority DMA transfer request is generated within one clock after the end of a single transfer, even if the previous higher priority DMA transfer request signal stays active, this request is not prioritized and the next DMA transfer after the bus is released for the CPU is a transfer based on the newly generated, lower priority DMA transfer request.

Figure 7-16, “Single Transfer Example 1,” on page 187 shows a DMAC transfer in single transfer mode. In this example the DMA channel 3 is used for a single transfer.

Figure 7-16: Single Transfer Example 1

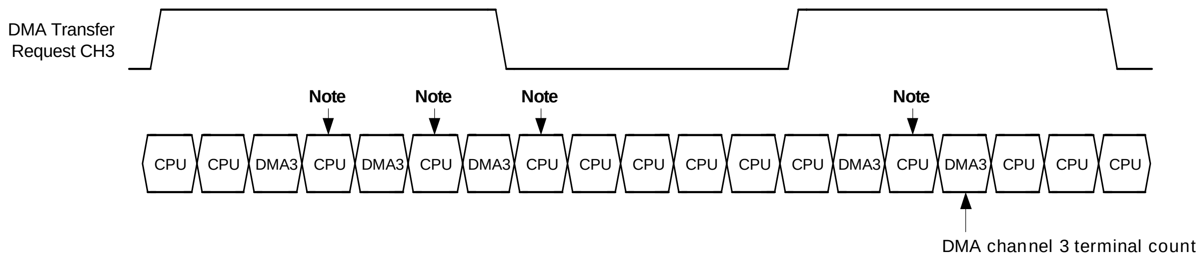
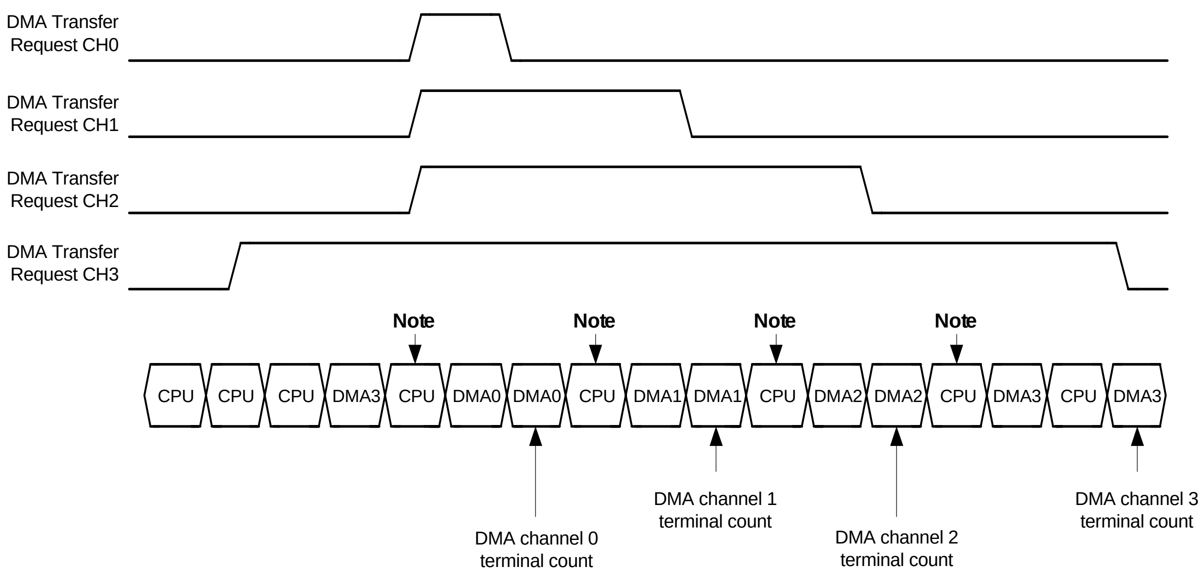


Figure 7-17, “Single Transfer Example 2,” on page 187 shows DMAC transfers in single transfer mode in which a higher priority DMA transfer request is generated. DMA channels 0 to 2 are used for a block transfer and channel 3 is used for a single transfer.

Figure 7-17: Single Transfer Example 2



Note: The bus is always released

Figure 7-18, “Single Transfer Example 3,” on page 188 shows a DMA transfer example in single transfer mode in which a lower priority DMA transfer request is generated within one clock after the end of a single transfer. DMA channels 0 and 3 are used for the single transfer example. When two DMA transfer request signals are activated at the same time, the two DMA transfers are performed alternately.

Figure 7-18: Single Transfer Example 3

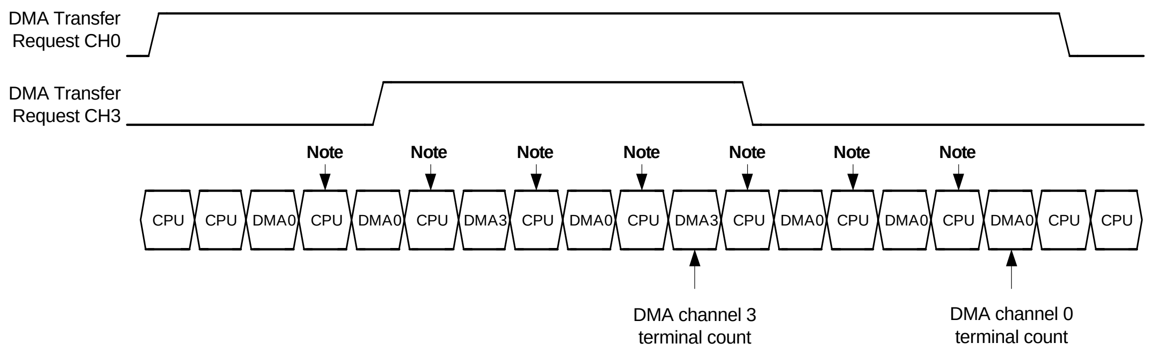
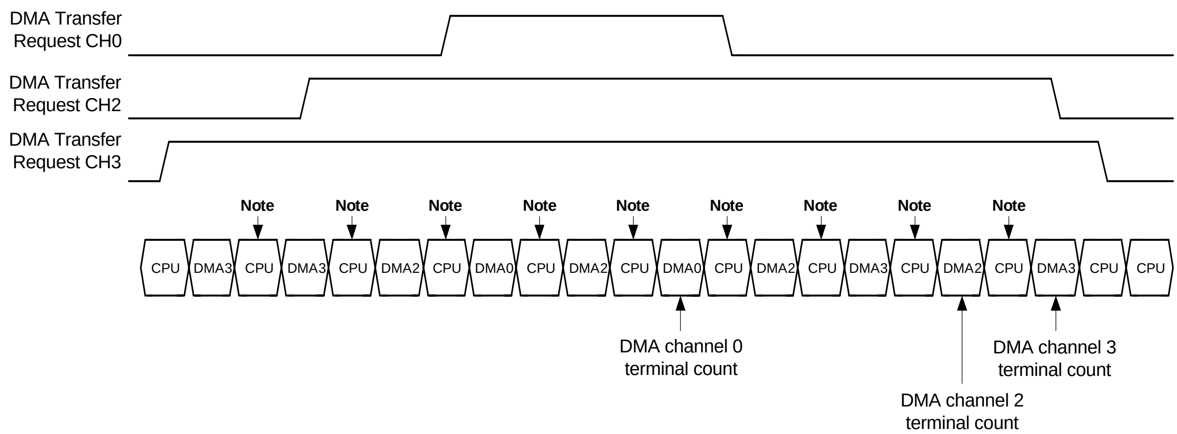


Figure 7-19, “Single Transfer Example 4,” on page 188 shows a single transfer mode example in which two or more lower priority DMA transfer requests are generated within one clock after the end of a single transfer. DMA channels 0, 2 and 3 are used for this single transfer example. When three or more DMA transfer request signals are activated at the same time always the two highest priority DMA transfers are performed alternately.

Figure 7-19: Single Transfer Example 4



Note: The bus is always released

7.5.2 Single-step transfer mode

In single-step transfer mode, the DMAC releases the bus at each byte/halfword/word transfer. Once a DMA transfer request signal is received, transfer is performed again. This operation continues until a terminal count occurs.
When the DMAC has released the bus, if another higher priority DMA transfer request is issued, the higher priority DMA request always takes precedence.

Figure 7-20, “Single-Step Transfer Example 1,” on page 189 shows a DMA transfer example in single-step transfer mode.

Figure 7-20: Single-Step Transfer Example 1

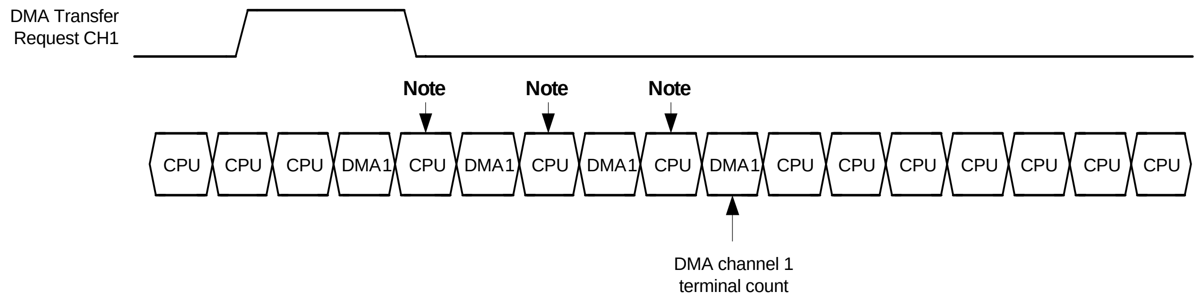
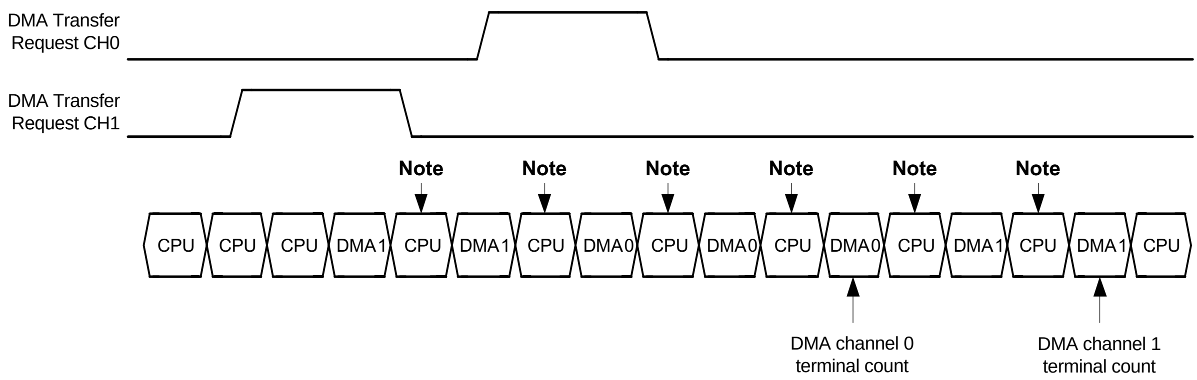


Figure 7-21, “Single-Step Transfer Example 2,” on page 189 shows a DMA transfer example in single-step transfer mode in which a higher priority DMA transfer request is generated while the lower DMA channel has released the bus.

Figure 7-21: Single-Step Transfer Example 2



Note: The bus is always released

7.5.3 Line Transfer Mode

In line transfer mode, the DMAC releases the bus after every four byte, halfword or word transfer. If there is a subsequent DMA transfer request, four transfers are performed again. This operation continues until a terminal count occurs. In two-cycle transfer, the operation from read to write is repeated four times.
If a higher priority DMA transfer request is generated while the DMAC has released the bus, the higher priority DMA transfer request always takes precedence. However, if a lower priority DMA transfer request is generated within one clock after the end of a line transfer, even if the previous higher priority DMA transfer request signal stays active, this request is not prioritized and the next DMA transfer after the bus is released for the CPU is a transfer based on the newly generated, lower priority DMA transfer request.

Figure 7-22, “Line Transfer Example 1,” on page 190 shows a DMA transfer example in line transfer mode.

Figure 7-22: Line Transfer Example 1

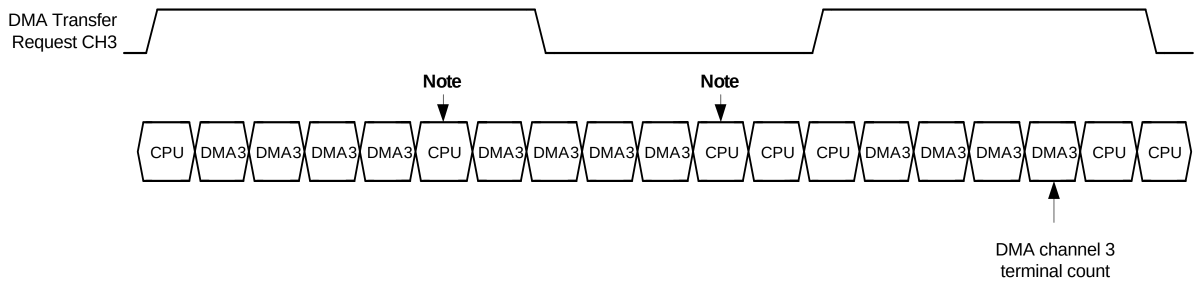
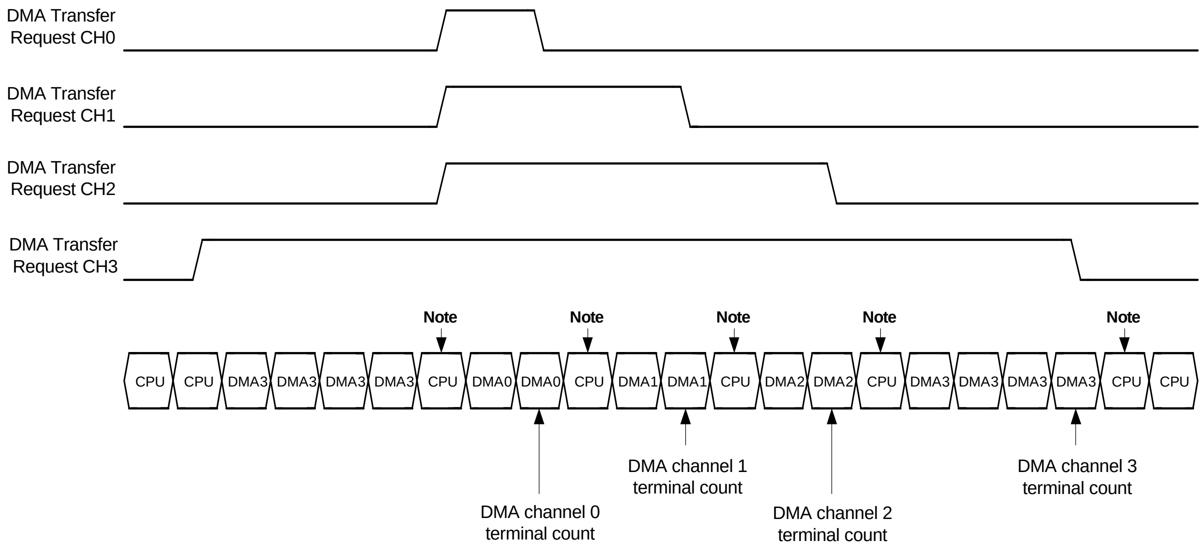


Figure 7-23, “Line Transfer Example 2,” on page 190 shows DMAC transfers in line transfer mode in which a higher priority DMA transfer request is generated. DMA channels 0 to 2 are used for a block transfer and channel 3 is used for a line transfer.

Figure 7-23: Line Transfer Example 2

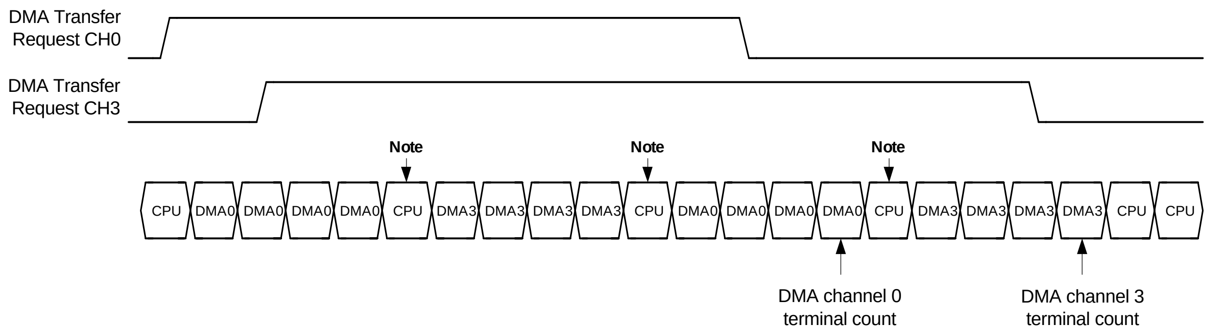


Note: The bus is always released

Figure 7-24, “Line Transfer Example 3,” on page 191 and Figure 7-25, “Line Transfer Example 4,” on page 191 shows DMAC transfers in line transfer mode in which a lower priority DMA transfer request is generated within one clock after the end of a line transfer. When two DMA transfer requests are activated at the same time, the two DMA transfers are performed alternately.

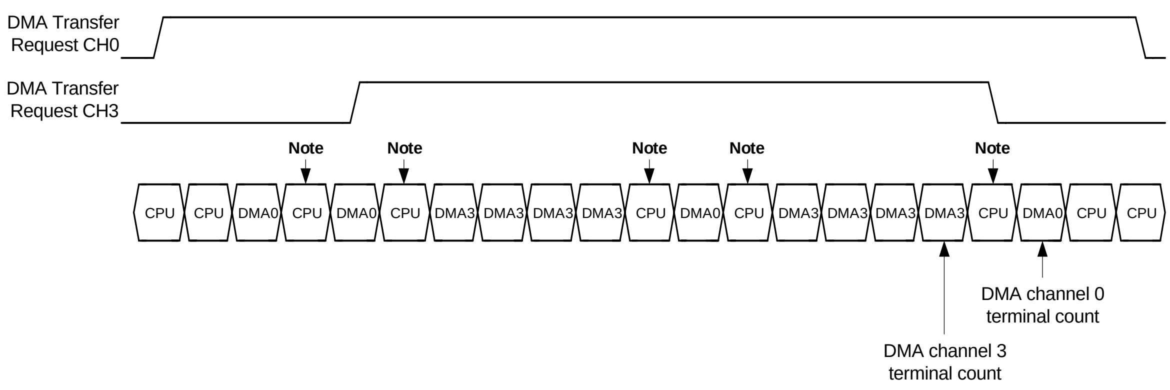
DMA channel 0 and 3 in Figure 7-24, “Line Transfer Example 3,” on page 191 are used for the line transfer example.

Figure 7-24: Line Transfer Example 3



DMA channel 0 in Figure 7-25, “Line Transfer Example 4,” on page 191 is used for a single transfer and channel 3 is used for the line transfer.

Figure 7-25: Line Transfer Example 4



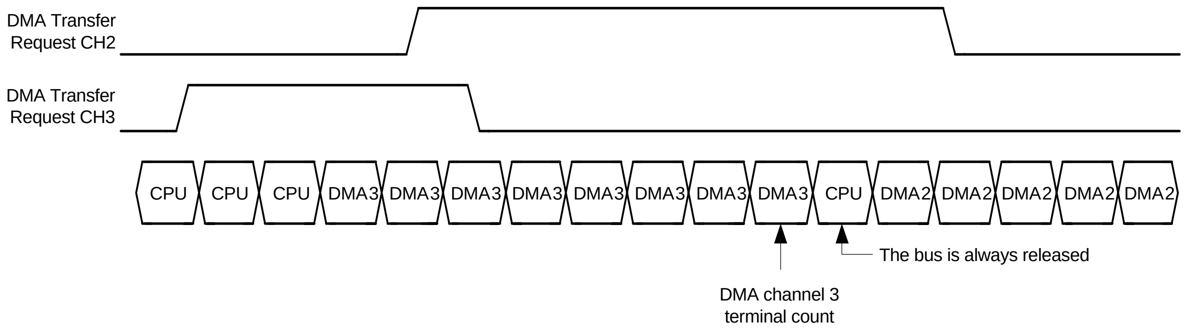
Note: The bus is always released

7.5.4 Block transfer mode

In the block transfer mode, once transfer begins, the DMAC continues the transfer operation without releasing the bus until a terminal count occurs. No other DMA requests are acknowledged during block transfer.
After the block transfer ends and the DMAC releases the bus and another DMA transfer can be acknowledged.

Figure 7-26, “Block Transfer Example,” on page 192 shows a block transfer mode example. It is a block transfer mode example in which a higher priority DMA transfer request is generated. DMA channels 2 and 3 are used for the block transfer example.

Figure 7-26: Block Transfer Example



7.6 Transfer Types

7.6.1 Two-cycle transfer

In two-cycle transfer, data transfer is performed in two cycles, a read cycle (source to DMAC) and a write cycle (DMAC to destination).
In the first cycle, the source address is output and reading is performed from the source to the DMAC. In the second cycle, the transfer destination address is output and writing is performed from the DMAC to the transfer destination.

Caution: A one-clock idle period is always inserted between a read cycle and a write cycle.

7.7 Transfer Object

7.7.1 Transfer type and transfer object

Table 7-1, “Relationship Between Transfer Type and Transfer Object,” on page 193 lists the relationships between transfer type and transfer object.

Table 7-1: Relationship Between Transfer Type and Transfer Object

		Destination			
		Two-Cycle Transfer			
		On-Chip Peripheral I/O	External I/O	Internal RAM	External Memory
Source	On-chip peripheral I/O	√	√	√	√
	External I/O	√	√	√	√
	Internal RAM	√	√	√	√
	External memory	√	√	√	√

Caution: Addresses between 3FFF000H and 3FFFFFFH cannot be specified for the source and destination address of DMA transfer. Be sure to specify an address between FFFF000H and FFFFFFFH.

7.8 DMA Channel Priorities

The DMA channel priorities are fixed as follows.

DMA channel 0 > DMA channel 1 > DMA channel 2 > DMA channel 3

These priorities are valid in the TI state only. In the block transfer mode, the channel used for transfer is never switched.
In the single-step transfer mode, if a higher priority DMA transfer request is issued while the bus is released (in the TI state), the higher priority DMA transfer request is acknowledged.

7.9 DMA Transfer Start Factors

There are two types of DMA transfer start factors, as shown below.

(1) Request from on-chip peripheral I/O

If the ENn and the TCn bits of the DCHCn register are set as shown below, and an interrupt request is issued from the on-chip peripheral I/O that is set in the DTFRn register, the DMA transfer starts.

- ENn bit = 1
- TCn bit = 0

(2) Request from software

If the STGn, the ENn and the TCn bits of the DCHCn register are set as follows, the DMA transfer starts.

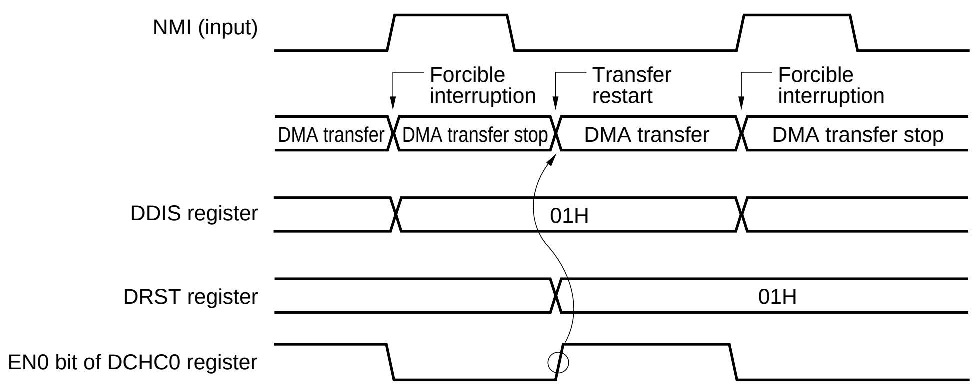
- STGn bit = 1
- ENn bit = 1
- TCn bit = 0

Remark: n = 0 to 3

7.10 Forcible Interruption

DMA transfer can be forcibly interrupted by NMI input during DMA transfer. At such a time, the DMAC clears the ENn bit of the DCHCn register of all channels and the DMA transfer disabled state is entered. An NMI request can then be acknowledged after the DMA transfer executed during NMI input is terminated.
In the single-step transfer mode, block transfer mode or line transfer mode the DMA transfer request is held in the DMAC. If the ENn bit is set the DMA transfer restarts from the point where it was interrupted. In the single transfer mode, if the ENn bit is set, the next DMA transfer request is acknowledged and DMA transfer begins.

Figure 7-27: Example of Forcible Interruption of DMA Transfer



Caution: To forcibly interrupt DMA transfer and stop the next transfer from occurring, the NMI signal must be made active before the end of the DMA transfer currently under execution. Moreover, although it is possible to restart DMA transfer following an interruption, this transfer cannot be executed under new settings (new conditions). Execute DMA transfer under new settings either after the end of the current transfer or after transfer has been forcibly terminated by setting the INITn bit of the DCHCn register.

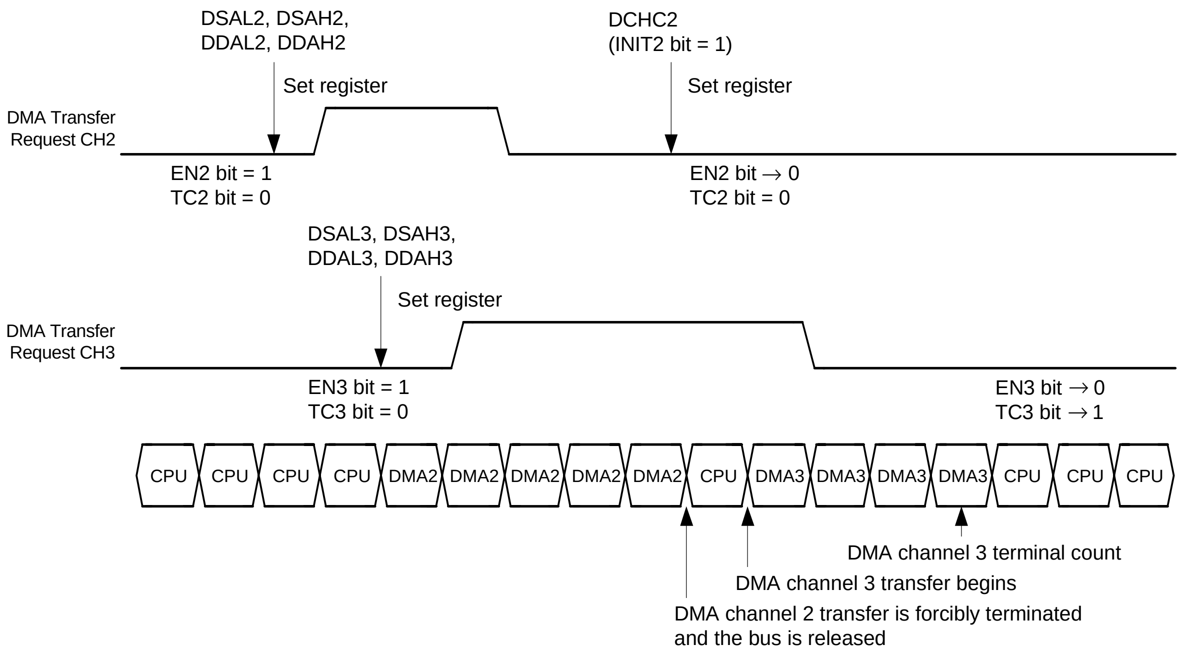
Remark: n = 0 to 3

7.11 Forcible Termination

In addition to the forcible interruption operation by means of the NMI input, DMA transfer can be forcibly terminated by the INITn bit of the DCHCn register. The following is an example of the operation of a forcible termination.

Figure 7-28, "DMA Transfer Forcible Termination Example 1," on page 196 shows a block transfer of channel 3 which begins during the DMA block transfer of DMA channel 2. The block transfer of DMA channel 2 is forcibly terminated by setting the INIT2 bit of its DCHC2 control register.

Figure 7-28: DMA Transfer Forcible Termination Example 1

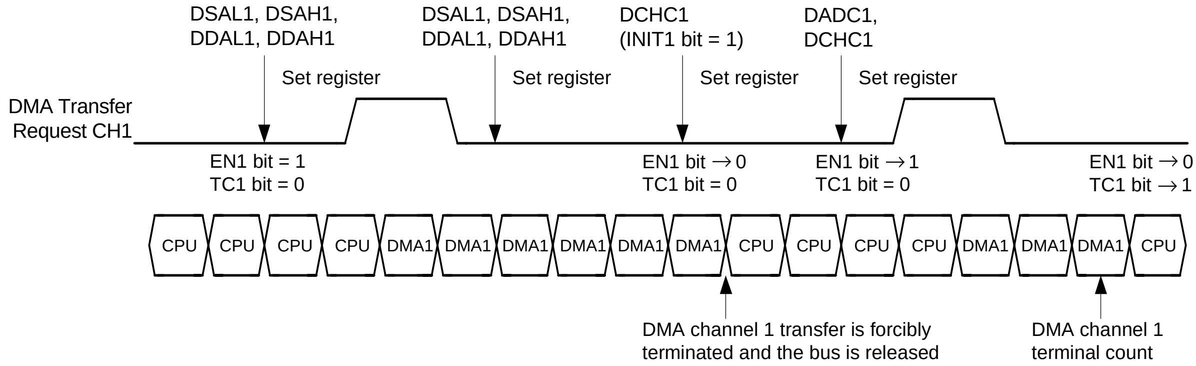


Remarks: **1.** The next condition can be set even during DMA transfer because the DSAn, DDAn, and DBCn registers are buffered registers. However, the setting to the DADCn register is invalid (refer to **7.3 Next Address Setting Function** and **7.2.4 DMA addressing control registers 0 to 3 (DADC0 to DADC3)**).

2. n = 0 to 3

Figure 7-29, "DMA Transfer Forcible Termination Example 2," on page 197 shows a forcible termination of a block transfer operation of DMA channel 1. A transfer containing a new configuration is executed.

Figure 7-29: DMA Transfer Forcible Termination Example 2



Remarks: 1. Since the DSALn, DSAHn, DDALn, DDAHn and DBCn registers are buffered registers, the next transfer condition can be set even during a DMA transfer. However, a setting in the DADCn register is ignored (refer to **7.3 Next Address Setting Function**)

2. n = 0 to 3

7.12 DMA Transfer Completion

7.12.1 DMA transfer end interrupt

When DMA transfer ends and the TCn bit of the DCHCn register is set, a DMA transfer end interrupt (INTDMAn) is issued to the interrupt controller (INTC).

Remark: n = 0 to 3

7.12.2 Terminal count output upon DMA transfer end

The terminal count signal becomes active for one clock during the last DMA transfer cycle.

7.13 Precautions

(1) Memory boundary

The transfer operation is not guaranteed if the source or the destination address exceeds the area of DMA objects (internal RAM, or peripheral I/O) during DMA transfer.

(2) Transfer of misaligned data

DMA transfer of 16-bit/32-bit bus width misaligned data is not supported.

(3) Times related to DMA transfer

The overhead before and after DMA transfer and the minimum execution clock for DMA transfer are shown below.

- Internal RAM access: 2 clocks

(4) Bus arbitration for CPU

The CPU can access on-chip peripheral I/O, and internal RAM not undergoing DMA transfer.
While data transfer is being executed between internal RAMs, the CPU can access external memory and peripheral I/O.

(5) Interrupt factors

DMA transfer is interrupted if a bus hold is issued.
If the factor (bus hold) interrupting DMA transfer disappears, DMA transfer promptly restarts.

Chapter 8 Interrupt/Exception Processing Function

The V850E/CA2 is provided with a dedicated interrupt controller (INTC) for interrupt servicing and can process a total of 64 maskable and three non-maskable interrupt requests.
An interrupt is an event that occurs independently of program execution, and an exception is an event whose occurrence is dependent on program execution. Generally, an exception takes precedence over an interrupt.
The V850E/CA2 can process interrupt requests from the on-chip peripheral hardware and external sources. Moreover, exception processing can be started by the TRAP instruction (software exception) or by generation of an exception event (i.e. fetching of an illegal opcode) (exception trap).
Eight levels of software-programmable priorities can be specified for each interrupt request. Interrupt servicing starts after no fewer than 11 system clocks (343 ns (@ 32 MHz)) following the generation of an interrupt request.

8.1 Features

- Interrupts
  - Non-maskable interrupts: 2 sources
  - Maskable interrupts: 63 sources
  - 8 levels of programmable priorities (maskable interrupts)
  - Multiple interrupt control according to priority
  - Masks can be specified for each maskable interrupt request.
  - Noise elimination, edge detection, and valid edge specification for external interrupt request signals.

- Exceptions
  - Software exceptions: 32 sources
  - Exception traps: 2 sources (illegal opcode exception and debug trap)

Interrupt/exception sources are listed in Table 8-1, “Interrupt/Exception Source List,” on page 200.

Table 8-1: Interrupt/Exception Source List (1/3)

Type	Classifi-cation	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
		Name	Controlling Register	Generating Source	Generating Unit				
Reset	Interrupt	RESET	–	$\overline{\text{RESET}}$ input	Pin	–	0000H	00000000H	undef.
Non-maskable	Interrupt	NMI0	–	P60 NMI Input	Port Module	–	0010H	00000010H	nextPC
		NMIWDT	–	Watchdog timer	WDT	–	0020H	00000020H	nextPC
		NMI2	-	Unused	–	–	0030H	00000030H	nextPC
Software exception	Exception	TRAP0n Note 1	–	TRAP instruction	–	–	004nH Note 1	00000040H	nextPC
	Exception	TRAP1n Note 1	–	TRAP instruction	–	–	005nH Note 1	00000050H	nextPC
Excep-tion trap	Exception	ILGOP/ DBTRAP	–	Illegal opcode/ DBTRAP instruction	–	–	0060H	00000060H	nextPC
Maskable	Interrupt	INTWT	WTIC	Real Time Clock Divider Tick	Watch timer	0	0080H	00000080H	nextPC
	Interrupt	INTTMD0	TMD0IC	Compare Match	Timer D0	1	0090H	00000090H	nextPC
	Interrupt	INTTMD1	TMD1IC	Compare Match	Timer D1	2	00A0H	000000A0H	nextPC
	Interrupt	INTWTI	WTIIC	Interval time	Watch timer	3	00B0H	000000B0H	nextPC
	Interrupt	INTP0	P0IC	P61	Port Module	4	00C0H	000000C0H	nextPC
	Interrupt	INTP1	P1IC	P62	Port Module	5	00D0H	000000D0H	nextPC
	Interrupt	INTP2	P2IC	P63	Port Module	6	00E0H	000000E0H	nextPC
	Interrupt	INTP3	P3IC	P64	Port Module	7	00F0H	000000F0H	nextPC
	Interrupt	INTP4	P4IC	P52	Port Module	8	0100H	00000100H	nextPC
	Interrupt	INTP5	P5IC	P53	Port Module	9	0110H	00000110H	nextPC
	Interrupt	INTTMG00	TMG00IC	Time base 0 Overflow	Timer G0	10	0120H	00000120H	nextPC
	Interrupt	INTTMG01	TMG01IC	Time base 1 Overflow	Timer G0	11	0130H	00000130H	nextPC
	Interrupt	INTGCC00	GCC00IC	CC coincidence Channel 0	Timer G0	12	0140H	00000140H	nextPC
	Interrupt	INTGCC01	GCC01IC	CC coincidence Channel 1	Timer G0	13	0150H	00000150H	nextPC
	Interrupt	INTGCC02	GCC02IC	CC coincidence Channel 2	Timer G0	14	0160H	00000160H	nextPC
	Interrupt	INTGCC03	GCC03IC	CC coincidence Channel 3	Timer G0	15	0170H	00000170H	nextPC
	Interrupt	INTGCC04	GCC04IC	CC coincidence Channel 4	Timer G0	16	0180H	00000180H	nextPC
	Interrupt	INTGCC05	GCC05IC	CC coincidence Channel 5	Timer G0	17	0190H	00000190H	nextPC
	Interrupt	INTTMG10	TMG10IC	Time base 0 Overflow	Timer G1	18	01A0H	000001A0H	nextPC
	Interrupt	INTTMG11	TMG11IC	Time base 1 Overflow	Timer G1	19	01B0H	000001B0H	nextPC
	Interrupt	INTGCC10	GCC10IC	CC coincidence Channel 0	Timer G1	20	01C0H	000001C0H	nextPC
	Interrupt	INTGCC11	GCC11IC	CC coincidence Channel 1	Timer G1	21	01D0H	000001D0H	nextPC
	Interrupt	INTGCC12	GCC12IC	CC coincidence Channel 2	Timer G1	22	01E0H	000001E0H	next PC
	Interrupt	INTGCC13	GCC13IC	CC coincidence Channel 3	Timer G1	23	01F0H	000001F0H	next PC
	Interrupt	INTGCC14	GCC14IC	CC coincidence Channel 4	Timer G1	24	0200H	00000200H	next PC
	Interrupt	INTGCC15	GCC15IC	CC coincidence Channel 5	Timer G1	25	0210H	00000210H	next PC
	Interrupt	INTTMC0	TMC0IC	Time base Overflow	Timer C0	26	0220H	00000220H	next PC

Notes: 1. n = 0 to FH

2. INTFC3RX, INTFC3TX, INTFC3ER, INTFC4RX, INTFC4TX and INTFC4ER are available only in the derivatives µPD703129 (A) and µPD703129 (A1).

Table 8-1: Interrupt/Exception Source List (2/3)

Type	Classifi-cation	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
		Name	Controlling Register	Generating Source	Generating Unit				
Maskable	Interrupt	INTCCC00	CCC0IC	CC coincidence Channel 0	Timer C0	27	0230H	00000230H	next PC
	Interrupt	INTCCC01	CCC1IC	CC coincidence Channel 1	Timer C0	28	0240H	00000240H	next PC
	Interrupt	INTAD	ADIC	A/D conversion end	A/D	29	0250H	00000250H	next PC
	Interrupt	INTMAC	MACIC	MAC Interrupt CGINTP 1-2	FCAN, MAC	30	0260H	00000260H	nextPC
	Interrupt	INTFC1RX	FC1RXIC	CAN1 Receive Interrupt	FCAN, machine 1	31	0270H	00000270H	nextPC
	Interrupt	INTFC1TX	FC1TXIC	CAN1 Transmit Interrupt	FCAN, machine 1	32	0280H	00000280H	nextPC
	Interrupt	INTFC1ER	FC1ERIC	CAN1 Error Interrupt	FCAN, machine 1	33	0290H	00000290H	nextPC
	Interrupt	INTFC2RX	FC2RXIC	CAN2 Receive Interrupt	FCAN, machine 2	34	02A0H	000002A0H	nextPC
	Interrupt	INTFC2TX	FC2TXIC	CAN2 Transmit Interrupt	FCAN, machine 2	35	02B0H	000002B0H	nextPC
	Interrupt	INTFC2ER	FC2ERIC	CAN2 Error Interrupt	FCAN, machine 2	36	02C0H	000002C0H	nextPC
	Interrupt **Note 2**	INTFC3RX	FC3RXIC	CAN3 Receive Interrupt	FCAN, machine 3	37	02D0H	000002D0H	nextPC
	Interrupt **Note 2**	INTFC3TX	FC3TXIC	CAN3 Transmit Interrupt	FCAN, machine 3	38	02E0H	000002E0H	nextPC
	Interrupt **Note 2**	INTFC3ER	FC3ERIC	CAN3 Error Interrupt	FCAN, machine 3	39	02F0H	000002F0H	nextPC
	Interrupt **Note 2**	INTFC4RX	FC4RXIC	CAN4 Receive Interrupt	FCAN, machine 4	40	0300H	00000300H	nextPC
	Interrupt **Note 2**	INTFC4TX	FC4TXIC	CAN4 Transmit Interrupt	FCAN, machine 4	41	0310H	00000310H	nextPC
	Interrupt **Note 2**	INTFC4ER	FC4ERIC	CAN4 Error Interrupt	FCAN, machine 4	42	0320H	00000320H	nextPC
	Interrupt	INTCSI0	CSI0IC	Transmission/ Reception Completion	CSI0	43	0330H	00000330H	nextPC
	Interrupt	INTCSI1	CSI1IC	Transmission/ Reception Completion	CSI1	44	0340H	00000340H	nextPC
	Interrupt	INTCSI2	CSI2IC	Transmission/ Reception Completion	CSI2	45	0350H	00000350H	nextPC
	Interrupt	INTSER0	SER0IC	Reception Error	UART0	46	0360H	00000360H	nextPC
	Interrupt	INSR0	SR0IC	Reception Completion	UART0	47	0370H	00000370H	nextPC
	Interrupt	INTST0	ST0IC	Transmission Completion	UART0	48	0380H	00000380H	nextPC
	Interrupt	INTSER1	SER1IC	Reception Error	UART1	49	0390H	00000390H	nextPC
	Interrupt	INSR1	SR1IC	Reception Completion	UART1	50	03A0H	000003A0H	nextPC
	Interrupt	INTST1	ST1IC	Transmission Completion	UART1	51	03B0H	000003B0H	nextPC
	Interrupt	INTDMA0	DMA0IC	DMA Channel 0 transfer completed	DMA0	52	03C0H	000003C0H	nextPC

Notes: 1. n = 0 to FH

2. INTFC3RX, INTFC3TX, INTFC3ER, INTFC4RX, INTFC4TX and INTFC4ER are available only in the derivatives µPD703129 (A) and µPD703129 (A1).

Table 8-1: Interrupt/Exception Source List (3/3)

Type	Classifi-cation	Interrupt/Exception Source				Default Priority	Exception Code	Handler Address	Restored PC
		Name	Controlling Register	Generating Source	Generating Unit				
Maskable	Interrupt	INTDMA1	DMA1IC	DMA Channel 1 transfer completed	DMA1	53	03D0H	000003D0H	nextPC
	Interrupt	INTDMA2	DMA2IC	DMA Channel 2 transfer completed	DMA2	54	03E0H	000003E0H	nextPC
	Interrupt	INTDMA3	DMA3IC	DMA Channel 3 transfer completed	DMA3	55	03F0H	000003F0H	nextPC
	Interrupt	INTDOVF	DOVFIC	DMA Overflow	DMA Trigger	56	0400H	00000400H	nextPC
	Interrupt	INTP00	P00IC	P30	Port Module	57	0410H	00000410H	nextPC
	Interrupt	INTP05	P05IC	P35	Port Module	58	0420H	00000420H	nextPC
	Interrupt	INTP10	P10IC	P40	Port Module	59	0430H	00000430H	nextPC
	Interrupt	INTP15	P15IC	P45	Port Module	60	0440H	00000440H	nextPC
	Interrupt	INTP20	P20IC	P54	Port Module	61	0450H	00000450H	nextPC
	Interrupt	INTP21	P21IC	P55	Port Module	62	0460H	00000460H	nextPC
	Interrupt	reserved	PIC63	reserved	reserved	63	0470H	00000470H	nextPC

Notes: 1. n = 0 to FH

2. INTFC3RX, INTFC3TX, INTFC3ER, INTFC4RX, INTFC4TX and INTFC4ER are available only in the derivatives µPD703129 (A) and µPD703129 (A1).

Remarks: 1. Default priority: The priority order when two or more maskable interrupt requests are generated at the same time. The highest priority is 0.

2. Restored PC: The value of the PC saved to EIPC or FEPC when interrupt/exception processing is started. However, the value of the PC saved when an interrupt is acknowledged during division (DIV, DIVH, DIVU, DIVHU) instruction execution is the value of the PC of the current instruction (DIV, DIVH, DIVU, DIVHU).

3. nextPC: The PC value that starts the processing following interrupt/exception processing.

4. The execution address of the illegal instruction when an illegal opcode exception occurs is calculated by (Restored PC – 4).

8.2 Non-Maskable Interrupts

A non-maskable interrupt request is acknowledged unconditionally, even when interrupts are in the interrupt disabled (DI) status.
Non-maskable interrupts of V850E/CA2 are available for the following two requests:

- NMI pin input
- Non-maskable watchdog timer interrupt request

When the valid edge specified by the ESN0 bit of the Interrupt mode register 3 (INTM3) is detected on the NMI pin, the interrupt occurs.
The watchdog timer interrupt request is only effective as non-maskable interrupt if the WDTM3 bit of the watchdog timer mode register (WDTM) is set 0.
If multiple non-maskable interrupts are generated at the same time, the highest priority servicing is executed according to the following priority order (the lower priority interrupt is ignored):

NMIWDT > NMI0

Note that if a NMI from port pin or NMIWDT request is generated while NMI from port pin is being serviced, the service is executed as follows.

(1) If a NMI0 is generated while NMI0 is being serviced

The new NMI0 request is held pending regardless of the value of the PSW.NP bit. The pending NMIVC request is acknowledged after servicing of the current NMI0 request has finished (after execution of the RETI instruction).

(2) If a NMIWDT request is generated while NMI0 is being serviced

If the PSW.NP bit remains set (1) while NMI0 is being serviced, the new NMIWDT request is held pending. The pending NMIWDT request is acknowledge after servicing of the current NMI0 request has finished (after execution of the RETI instruction).
If the PSW.NP bit is cleared (0) while NMI0 is being serviced, the newly generated NMIWDT request is executed (NMI0 servicing is halted).

Remark: PSW.NP: The NP bit of the PSW register.

Cautions: 1. Although the values of the PC and PSW are saved to an NMI status save register (FEPC, FEPSW) when a non-maskable interrupt request is generated, only the NMI0 can be restored by the RETI instruction at this time. Because NMIWDT cannot be restored by the RETI instruction, the system must be reset after servicing this interrupt.

2. If PSW.NP is cleared to 0 by the LDSR instruction during non-maskable interrupt servicing, a NMI0 interrupt afterwards cannot be acknowledged correctly.

Figure 8-1: Example of Non-Maskable Interrupt Request Acknowledgement Operation (1/2)

(a) Multiple NMI requests generated at the same time

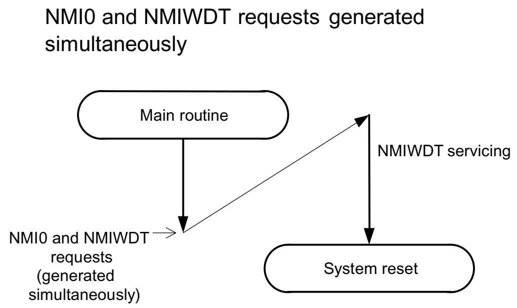
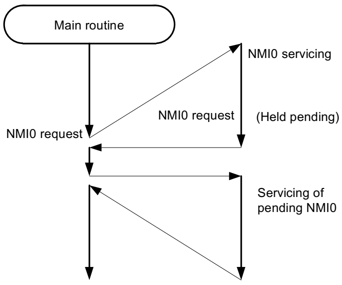
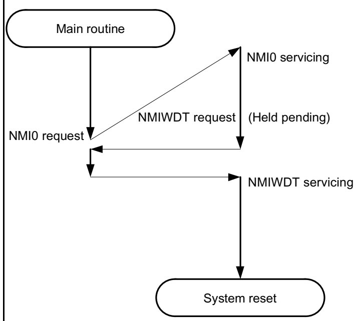
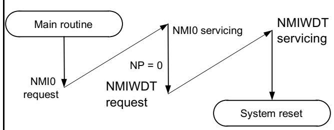
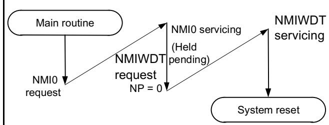
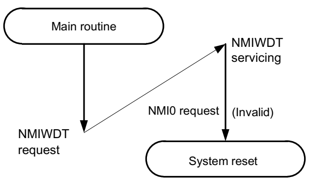
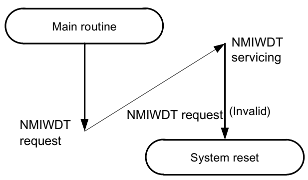


Figure 8-1: Example of Non-Maskable Interrupt Request Acknowledgement Operation (2/2)

(b) NMI request generated during NMI servicing

<table>
<tr><th rowspan="2">NMI being serviced</th><th colspan="2">NMI request generated during NMI servicing</th></tr>
<tr><th>NMI0</th><th>NMIWDT</th></tr>
<tr><td>NMI0</td><td>NMI0 request generated during NMI0 servicing

</td><td>NMIWDT request generated during NMI0 servicing (NP = 1 retained before NMI1 request)


NMIWDT request generated during NMI0 servicing (NP=0 set before NMIWDT request)


NMIWDT request generated during NMI0 servicing (NP=0 set after NMIWDT request)

</td></tr>
<tr><td>NMIWDT</td><td>NMI0 request generated during NMIWDT servicing

</td><td>NMI1 request generated during NMIWDT servicing

</td></tr>
</table>

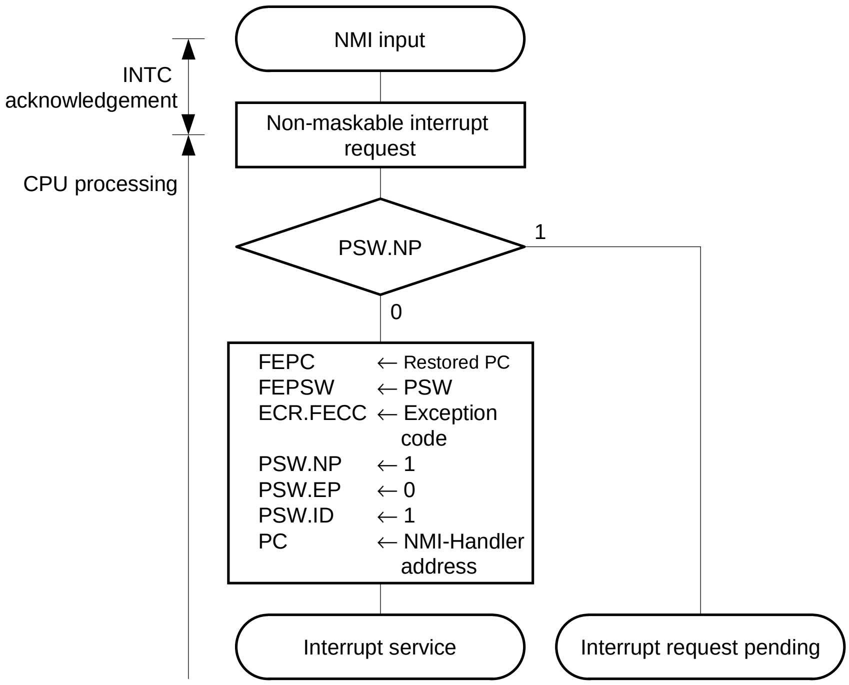
8.2.1 Operation

If a non-maskable interrupt is generated, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the restored PC to FEPC.
(2) Saves the current PSW to FEPSW.
(3) Writes exception code 0010H to the higher halfword (FECC) of ECR.
(4) Sets the NP and ID bits of the PSW and clears the EP bit.
(5) Sets the handler address corresponding to the non-maskable interrupt to the PC, and transfers control.

The processing configuration of a non-maskable interrupt is shown in Figure 8-2.

Figure 8-2: Processing Configuration of Non-Maskable Interrupt



8.2.2 Restore

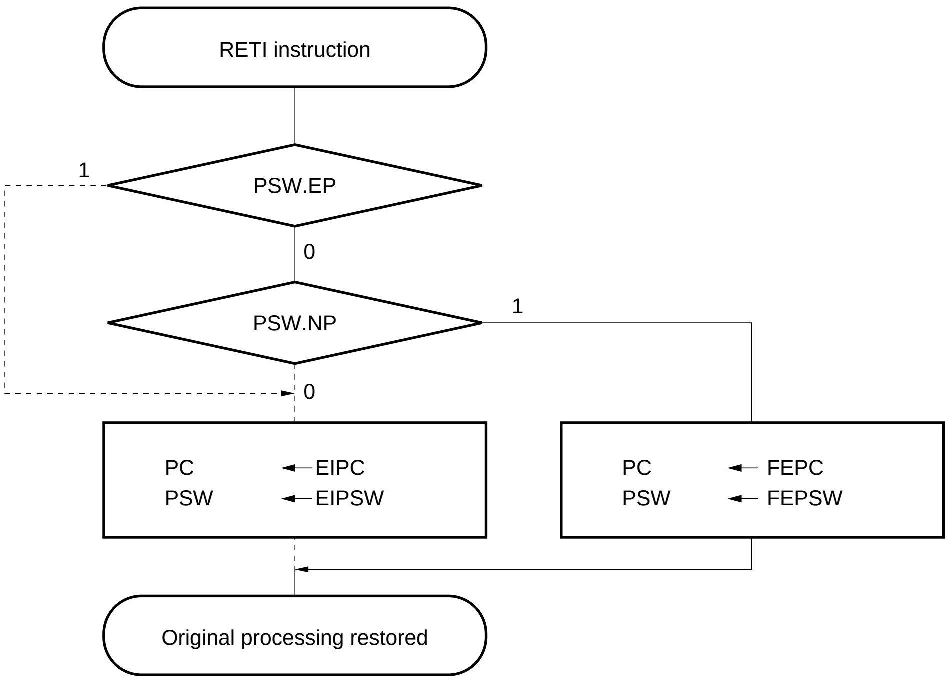
(1) NMI0

Execution is restored from the non-maskable interrupt (NMI0) processing by the RETI instruction. When the RETI instruction is executed, the CPU performs the following processing, and transfers control to the address of the restored PC.

<1> Restores the values of the PC and the PSW from FEPC and FEPSW, respectively, because the EP bit of the PSW is 0 and the NP bit of the PSW is 1.
<2> Transfers control back to the address of the restored PC and PSW.

Figure 8-3 illustrates how the RETI instruction is processed.

Figure 8-3: RETI Instruction Processing



Caution: When the PSW.EP bit and PSW.NP bit are changed by the LDSR instruction during non-maskable interrupt processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 1 using the LDSR instruction immediately before the RETI instruction.

Remark: The solid line indicates the CPU processing flow.

(2) NMIWDT

Restoring by RETI instruction is not possible. Perform a system reset after interrupt servicing.

8.2.3 Non-maskable interrupt status flag (NP)

The NP flag is a status flag that indicates that non-maskable interrupt (NMI) processing is under execution.
This flag is set when an NMI interrupt has been acknowledged, and masks all interrupt requests and exceptions to prohibit multiple interrupts from being acknowledged.

Figure 8-4: Non-maskable Interrupt Status Flag (NP)

	31																							8	7	6	5	4	3	2	1	0	Initial value
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

Bit Position	Bit Name	Function
7	NP	Indicates whether NMI interrupt processing is in progress. 0: No NMI interrupt processing 1: NMI interrupt currently being processed

8.2.4 Edge Detection Function

The behaviour of the non-maskable interrupt (NMI0) can be specified by the interrupt mode register 3. The valid edge of the external NMI pin input can be specified by the ESN0 bit.
The register can be read/written in 8-bit or 1-bit units.

Figure 8-5: Interrupt Mode Register 3 (INTM3)

	7	6	5	4	3	2	1	0	Address	Initial value
INTM3	0	0	0	0	0	0	0	ESN0	FFFFF886H	00H

Bit Position	Bit Name	Function
0	ESN0	Specifies the NMI pin's valid edge 0: Falling edge 1: Rising edge

Notes: 1. This register can be written only once. ESN0 is cleared to “0” by Reset.

2. This register should always be programmed even if the user needs to use the reset value. This will prevent unintended write to this register afterwards.

3. NMI functionality is masked by PMC60. Selection of valid edge for NMI must be performed while PMC60 is “0”.

8.3 Maskable Interrupts

Maskable interrupt requests can be masked by interrupt control registers. The V850E/CA2 has 63 maskable interrupt sources.
If two or more maskable interrupt requests are generated at the same time, they are acknowledged according to the default priority. In addition to the default priority, eight levels of priorities can be specified by using the interrupt control registers (programmable priority control).
When an interrupt request has been acknowledged, the acknowledgement of other maskable interrupt requests is disabled and the interrupt disabled (DI) status is set.
When the EI instruction is executed in an interrupt processing routine, the interrupt enabled (EI) status is set, which enables servicing of interrupts having a higher priority than the interrupt request in progress (specified by the interrupt control register). Note that only interrupts with a higher priority will have this capability; interrupts with the same priority level cannot be nested.
However, if multiple interrupts are executed, the following processing is necessary.

(1) Save EIPC and EIPSW in memory or a general-purpose register before executing the EI instruction.
(2) Execute the DI instruction before executing the RETI instruction, then reset EIPC and EIPSW with the values saved in (1).

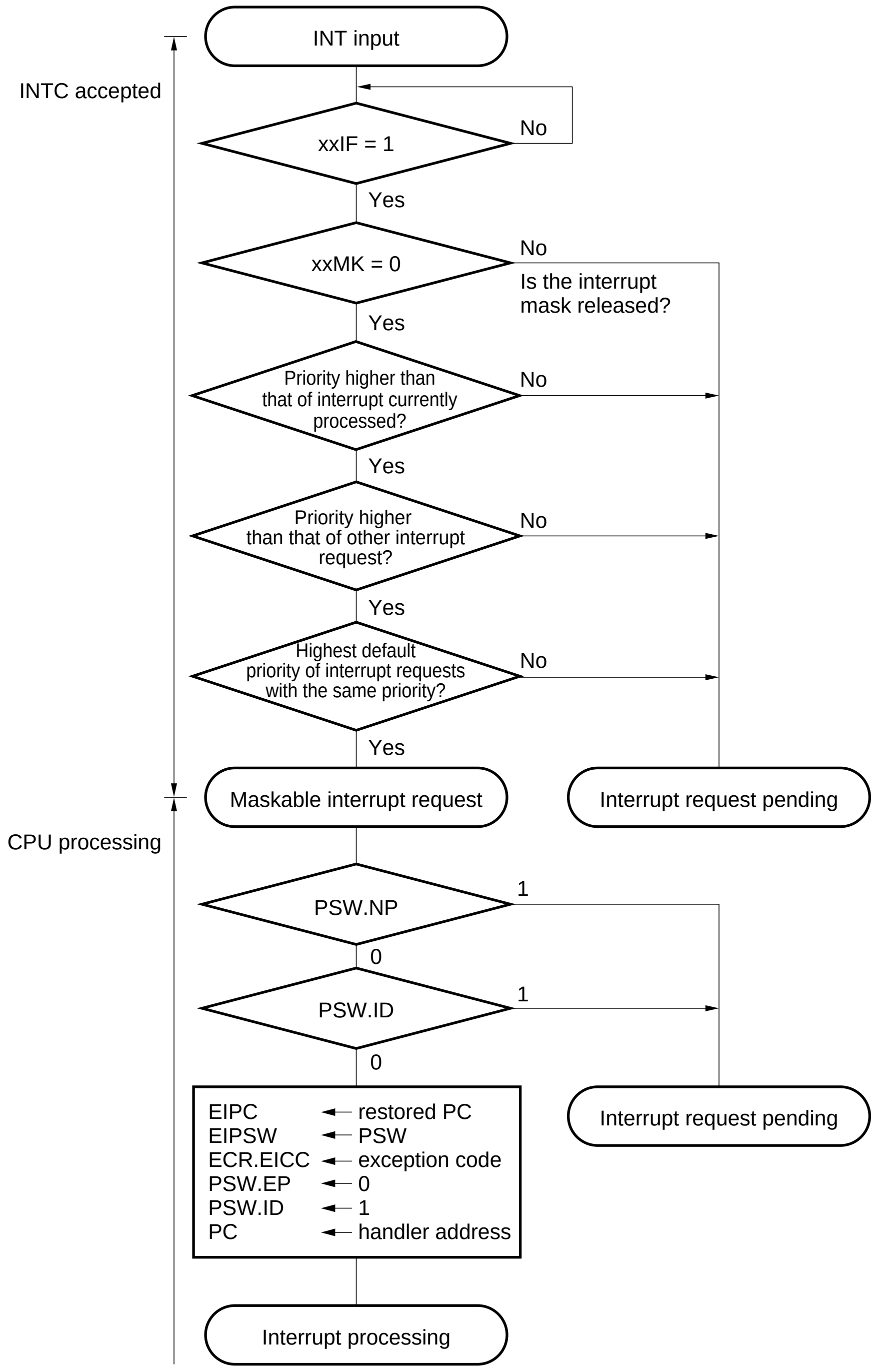
8.3.1 Operation

If a maskable interrupt occurs by INT input, the CPU performs the following processing, and transfers control to a handler routine:

(1) Saves the restored PC to EIPC.
(2) Saves the current PSW to EIPSW.
(3) Writes an exception code to the lower halfword of ECR (EICC).
(4) Sets the ID bit of the PSW and clears the EP bit.
(5) Sets the handler address corresponding to each interrupt to the PC, and transfers control.

The processing configuration of a maskable interrupt is shown in Figure 8-6, “Maskable Interrupt Processing,” on page 210.

Figure 8-6: Maskable Interrupt Processing



Note: For the ISPR register, see 8.3.6 "In-service priority register (ISPR)" on page 220.

An INT input masked by the interrupt controllers and an INT input that occurs while another interrupt is being processed (when PSW.NP = 1 or PSW.ID = 1) are held pending internally by the interrupt controller. In such case, if the interrupts are unmasked, or when PSW.NP = 0 and PSW.ID = 0 as set by the RETI and LDSR instructions, input of the pending INT starts the new maskable interrupt processing.

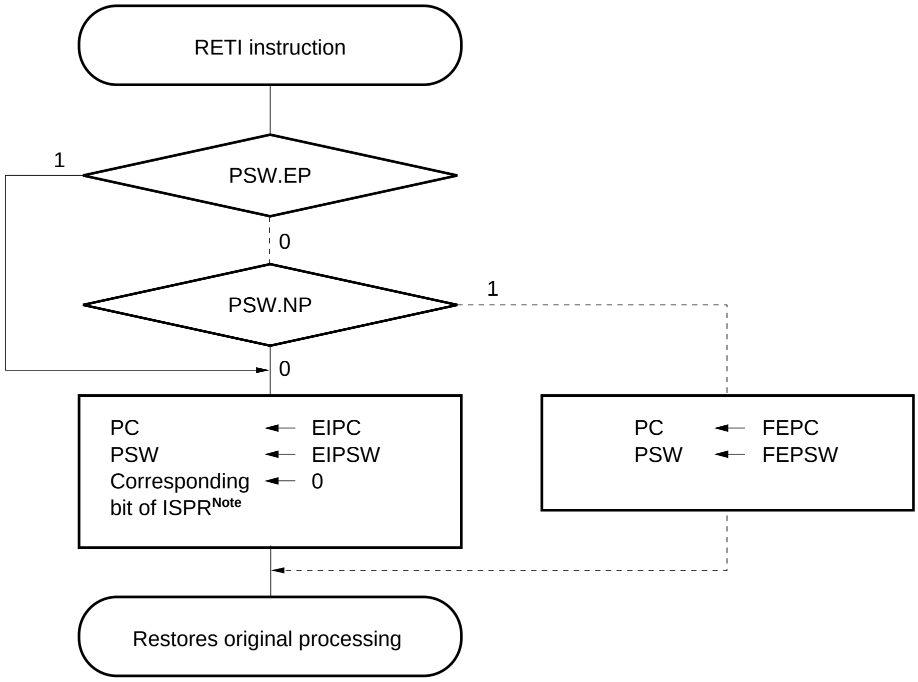
8.3.2 Restore

Recovery from maskable interrupt processing is carried out by the RETI instruction.
When the RETI instruction is executed, the CPU performs the following steps, and transfers control to the address of the restored PC.

(1) Restores the values of the PC and the PSW from EIPC and EIPSW because the EP bit of the PSW is 0 and the NP bit of the PSW is 0.
(2) Transfers control to the address of the restored PC and PSW.

Figure 8-7 illustrates the processing of the RETI instruction.

Figure 8-7: RETI Instruction Processing



Note: For the ISPR register, see 8.3.6 “In-service priority register (ISPR)” on page 220.

Caution: When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during maskable interrupt processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 0 using the LDSR instruction immediately before the RETI instruction.

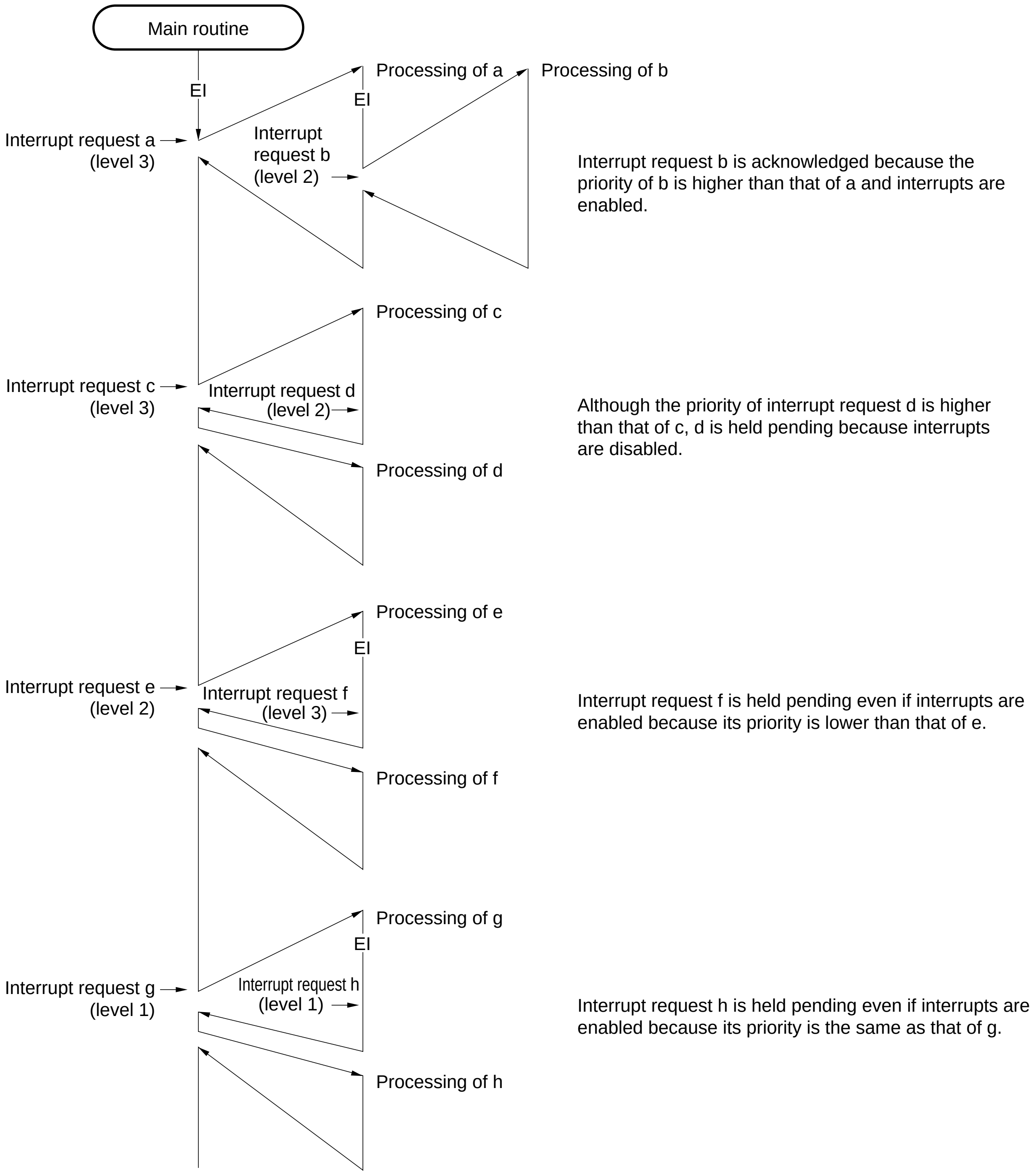
Remark: The solid lines show the CPU processing flow.

8.3.3 Priorities of maskable interrupts

The V850E/CA2 provides multiple interrupt servicing in which an interrupt is acknowledged while another interrupt is being serviced. Multiple interrupts can be controlled by priority levels.
There are two types of priority level control: control based on the default priority levels, and control based on the programmable priority levels that are specified by the interrupt priority level specification bit (xxPRn) of the interrupt control register (xxICn). When two or more interrupts having the same priority level specified by the xxPRn bit are generated at the same time, interrupts are serviced in order depending on the priority level allocated to each interrupt request type (default priority level) beforehand. For more information, refer to Table 8-1: “Interrupt/Exception Source List” on page 200. The programmable priority control customizes interrupt requests into eight levels by setting the priority level specification flag.

Note that when an interrupt request is acknowledged, the ID flag of PSW is automatically set to 1. Therefore, when multiple interrupts are to be used, clear the ID flag to 0 beforehand (for example, by placing the EI instruction in the interrupt service program) to set the interrupt enable mode.

Figure 8-8: Example of Processing in Which Another Interrupt Request Is Issued While an Interrupt Is Being Processed (1/2)

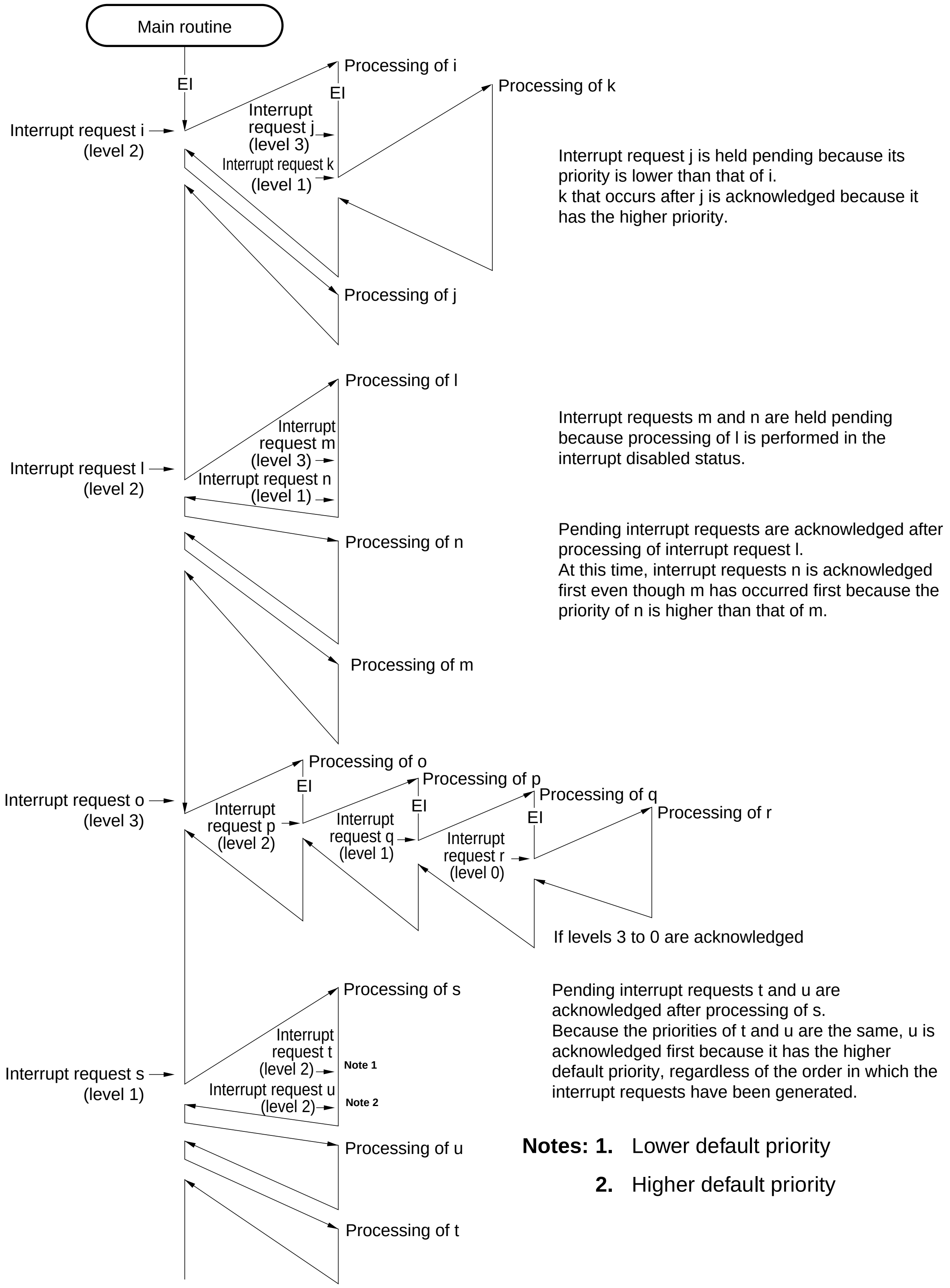


Caution: The values of the EIPC and EIPSW registers must be saved before executing multiple interrupts. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.

Remarks: 1. <a> to <u> in the figure are the temporary names of interrupt requests shown for the sake of explanation.

2. The default priority in the figure indicates the relative priority between two interrupt requests.

Figure 8-8: Example of Processing in Which Another Interrupt Request Is Issued While an Interrupt Is Being Processed (2/2)

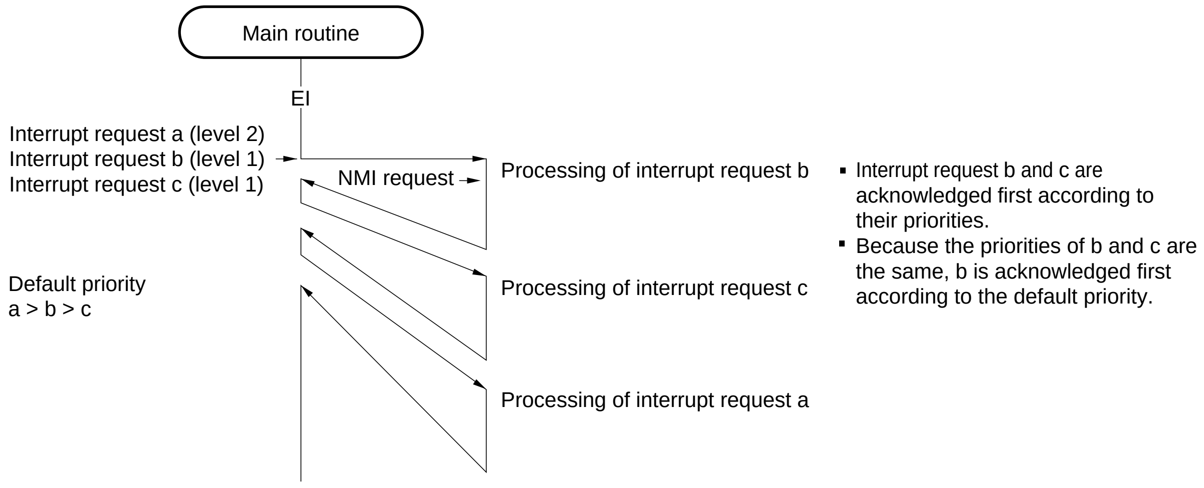


Notes: 1. Lower default priority

2. Higher default priority

Caution: The values of the EIPC and EIPSW registers must be saved before executing multiple interrupts. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.

Figure 8-9: Example of Processing Interrupt Requests Simultaneously Generated



Caution: **The values of the EIPC and EIPSW registers must be saved before executing multiple interrupts. When returning from multiple interrupt servicing, restore the values of EIPC and EIPSW after executing the DI instruction.**

Remark: <a> to <c> in the figure are the temporary names of interrupt requests shown for the sake of explanation.

8.3.4 Interrupt control register (xxIC)

An interrupt control register is assigned to each interrupt request (maskable interrupt) and sets the control conditions for each maskable interrupt request.
This register can be read/written in 8-bit or 1-bit units.

Figure 8-10: Interrupt Control Register (xxIC)

	7	6	5	4	3	2	1	0	Address	Initial value
xxIC	xxIF	xxMK	0	0	0	xxPR2	xxPR1	xxPR0	FFFFF110H to FFFF18EH	47H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7</td><td>xxIF</td><td>This is an interrupt request flag.
0: Interrupt request not issued
1: Interrupt request issued
The flag xxIFn is reset automatically by the hardware if an interrupt request is acknowledged.</td></tr>
<tr><td>6</td><td>xxMK</td><td>This is an interrupt mask flag.
0: Enables interrupt processing
1: Disables interrupt processing (pending)</td></tr>
<tr><td>2 to 0</td><td>xxPR2 to xxPR0</td><td>8 levels of priority order are specified for each interrupt.
<table>
<tr><th>xxPR2</th><th>xxPR1</th><th>xxPR0</th><th>Interrupt Priority Specification Bit</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Specifies level 0 (highest)</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Specifies level 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Specifies level 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Specifies level 3</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Specifies level 4</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Specifies level 5</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Specifies level 6</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Specifies level 7 (lowest)</td></tr>
</table></td></tr>
</table>

Remark: xx: Identification name of each peripheral unit (WT, TMD, P, TMG, GCC, AD, MAC, FC, CSI, UART, DMA)

The address and bit of each interrupt control register are shown in the following Table 8-2, “Addresses and Bits of Interrupt Control Registers,” on page 217.

Table 8-2: Addresses and Bits of Interrupt Control Registers (1/2)

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF110H	WTIC	WTIF	WTMK	0	0	0	WTPR2	WTPR1	WTPR0
FFFFF112H	TMD0IC	TMD0IF	TMD0MK	0	0	0	TMD0PR2	TMD0PR1	TMD0PR0
FFFFF114H	TMD1IC	TMD1IF	TMD1MK	0	0	0	TMD1PR2	TMD1PR1	TMD1PR0
FFFFF116H	WTIIC	WTIIF	WTIMK	0	0	0	WTIPR2	WTIPR1	WTIPR0
FFFFF118H	P0IC	P0IF	P0MK	0	0	0	P0PR2	P0PR1	P0PR0
FFFFF11AH	P1IC	P1IF	P1MK	0	0	0	P1PR2	P1PR1	P1PR0
FFFFF11CH	P2IC	P2IF	P2MK	0	0	0	P2PR2	P2PR1	P2PR0
FFFFF11EH	P3IC	P3IF	P3MK	0	0	0	P3PR2	P3PR1	P3PR0
FFFFF120H	P4IC	P4IF	P4MK	0	0	0	P4PR2	P4PR1	P4PR0
FFFFF122H	P5IC	P5IF	P5MK	0	0	0	P5PR2	P5PR1	P5PR0
FFFFF124H	TMG00IC	TMG00IF	TMG00MK	0	0	0	TMG00PR2	TMG00PR1	TMG00PR0
FFFFF126H	TMG01IC	TMG01IF	TMG01MK	0	0	0	TMG01PR2	TMG01PR1	TMG01PR0
FFFFF128H	GCC00IC	GCC00IF	GCC00MK	0	0	0	GCC00PR2	GCC00PR1	GCC00PR0
FFFFF12AH	GCC01IC	GCC01IF	GCC01MK	0	0	0	GCC01PR2	GCC01PR1	GCC01PR0
FFFFF12CH	GCC02IC	GCC02IF	GCC02MK	0	0	0	GCC02PR2	GCC02PR1	GCC02PR0
FFFFF12EH	GCC03IC	GCC03IF	GCC03MK	0	0	0	GCC03PR2	GCC03PR1	GCC03PR0
FFFFF130H	GCC04IC	GCC04IF	GCC04MK	0	0	0	GCC04PR2	GCC04PR1	GCC04PR0
FFFFF132H	GCC05IC	GCC05IF	GCC05MK	0	0	0	GCC05PR2	GCC05PR1	GCC05PR0
FFFFF134H	TMG10IC	TMG10IF	TMG10MK	0	0	0	TMG10PR2	TMG10PR1	TMG10PR0
FFFFF136H	TMG11IC	TMG11IF	TMG11MK	0	0	0	TMG11PR2	TMG11PR1	TMG11PR0
FFFFF138H	GCC10IC	GCC10IF	GCC10MK	0	0	0	GCC10PR2	GCC10PR1	GCC10PR0
FFFFF13AH	GCC11IC	GCC11IF	GCC11MK	0	0	0	GCC11PR2	GCC11PR1	GCC11PR0
FFFFF13CH	GCC12IC	GCC12IF	GCC12MK	0	0	0	GCC12PR2	GCC12PR1	GCC12PR0
FFFFF13EH	GCC13IC	GCC13IF	GCC13MK	0	0	0	GCC13PR2	GCC13PR1	GCC13PR0
FFFFF140H	GCC14IC	GCC14IF	GCC14MK	0	0	0	GCC14PR2	GCC14PR1	GCC14PR0
FFFFF142H	GCC15IC	GCC15IF	GCC15MK	0	0	0	GCC15PR2	GCC15PR1	GCC15PR0
FFFFF144H	TMC0IC	TMC0IF	TMC0MK	0	0	0	TMC0PR2	TMC0PR1	TMC0PR0
FFFFF146H	CCC00IC	CCC00IF	CCC00MK	0	0	0	CCC00PR2	CCC00PR1	CCC00PR0
FFFFF148H	CCC01IC	CCC01IF	CCC01MK	0	0	0	CCC01PR2	CCC01PR1	CCC01PR0
FFFFF14AH	ADIC	ADIF	ADMK	0	0	0	ADPR2	ADPR1	ADPR0
FFFFF14CH	MACIC	MACIF	MACMK	0	0	0	MACPR2	MACPR1	MACPR0
FFFFF14EH	FC1RXIC	FC1RXIF	FC1RXMK	0	0	0	FC1RXPR2	FC1RXPR1	FC1RXPR0
FFFFF150H	FC1TXIC	FC1TXIF	FC1TXMK	0	0	0	FC1TXPR2	FC1TXPR1	FC1TXPR0
FFFFF152H	FC1ERIC	FC1ERIF	FC1ERMK	0	0	0	FC1ERPR2	FC1ERPR1	FC1ERPR0
FFFFF154H	FC2RXIC	FC2RXIF	FC2RXMK	0	0	0	FC2RXPR2	FC2RXPR1	FC2RXPR0
FFFFF156H	FC2TXIC	FC2TXIF	FC2TXMK	0	0	0	FC2TXPR2	FC2TXPR1	FC2TXPR0
FFFFF158H	FC2ERIC	FC2ERIF	FC2ERMK	0	0	0	FC2ERPR2	FC2ERPR1	FC2ERPR0
FFFFF15AH	FC3RXIC[NOTE]	FC3RXIF	FC3RXMK	0	0	0	FC3RXPR2	FC3RXPR1	FC3RXPR0
FFFFF15CH	FC3TXIC[NOTE]	FC3TXIF	FC3TXMK	0	0	0	FC3TXPR2	FC3TXPR1	FC3TXPR0
FFFFF15EH	FC3ERIC[NOTE]	FC3ERIF	FC3ERMK	0	0	0	FC3ERPR2	FC3ERPR1	FC3ERPR0
FFFFF160H	FC4RXIC[NOTE]	FC4RXIF	FC4RXMK	0	0	0	FC4RXPR2	FC4RXPR1	FC4RXPR0
FFFFF162H	FC4TXIC[NOTE]	FC4TXIF	FC4TXMK	0	0	0	FC4TXPR2	FC4TXPR1	FC4TXPR0
FFFFF164H	FC4ERIC[NOTE]	FC4ERIF	FC4ERMK	0	0	0	FC4ERPR2	FC4ERPR1	FC4ERPR0

Table 8-2: Addresses and Bits of Interrupt Control Registers (2/2)

Address	Register	Bit							
		7	6	5	4	3	2	1	0
FFFFF166H	CSI0IC	CSI0IF	CSI0MK	0	0	0	CSI0PR2	CSI0PR1	CSI0PR0
FFFFF168H	CSI1IC	CSI1IF	CSI1MK	0	0	0	CSI1PR2	CSI1PR1	CSI1PR0
FFFFF16AH	CSI2IC	CSI2IF	CSI2MK	0	0	0	CSI2PR2	CSI2PR1	CSI2PR0
FFFFF16CH	SER0IC	SER0IF	SER0MK	0	0	0	SER0PR2	SER0PR1	SER0PR0
FFFFF16EH	SR0IC	SR0IF	SR0MK	0	0	0	SR0PR2	SR0PR1	SR0PR0
FFFFF170H	ST0IC	ST0IF	ST0MK	0	0	0	ST0PR2	ST0PR1	ST0PR0
FFFFF172H	SER1IC	SER1IF	SER1MK	0	0	0	SER1PR2	SER1PR1	SER1PR0
FFFFF174H	SR1IC	SR1IF	SR1MK	0	0	0	SR1PR2	SR1PR1	SR1PR0
FFFFF176H	ST1IC	ST1IF	ST1MK	0	0	0	ST1PR2	ST1PR1	ST1PR0
FFFFF178H	DMA0IC	DMA0IF	DMA0MK	0	0	0	DMA0PR2	DMA0PR1	DMA0PR0
FFFFF17AH	DMA1IC	DMA1IF	DMA1MK	0	0	0	DMA1PR2	DMA1PR1	DMA1PR0
FFFFF17CH	DMA2IC	DMA2IF	DMA2MK	0	0	0	DMA2PR2	DMA2PR1	DMA2PR0
FFFFF17EH	DMA3IC	DMA3IF	DMA3MK	0	0	0	DMA3PR2	DMA3PR1	DMA3PR0
FFFFF180H	DOVFIC	DOVFIF	DOVFMK	0	0	0	DOVFPR2	DOVFPR1	DOVFPR0
FFFFF182H	P00IC	P00IF	P00MK	0	0	0	P00PR2	P00PR1	P00PR0
FFFFF184H	P05IC	P05IF	P05MK	0	0	0	P05PR2	P05PR1	P05PR0
FFFFF186H	P10IC	P10IF	P10MK	0	0	0	P10PR2	P10PR1	P10PR0
FFFFF188H	P15IC	P15IF	P15MK	0	0	0	P15PR2	P15PR1	P15PR0
FFFFF18AH	P20IC	P20IF	P20MK	0	0	0	P20PR2	P20PR1	P20PR0
FFFFF18CH	P21IC	P21IF	P21MK	0	0	0	P21PR2	P21PR1	P21PR0
FFFFF18EH	Reserved	1	1	0	0	0	1	1	1

Remark: For the interrupt source to the respective controlling registers xxICn refer to Table 8-1, “Interrupt/Exception Source List,” on page 200.

Note: FC3RXIC, FC3TXIC, FC3ERIC, FC4RXIC, FC4TXIC and FC4ERIC are available only in the derivatives µPD703129 (A) and µPD703129 (A1).

8.3.5 Interrupt mask registers 0 to 3 (IMR0 to IMR3)

These registers set the interrupt mask state for the maskable interrupts.
The xxMK bit of the IMR0 to IMR3 registers is equivalent to the xxMK bit of the xxIC register.
IMRm registers can be read/written in 16-bit units (m = 0 to 3).
When the IMRm register is divided into two registers: higher 8 bits (IMRmH register) and lower 8 bits (IMRmL register), these registers can be read/written in 8-bit or 1-bit units (m = 0 to 3).
The address of the lower 8-bit register IMRmL is equal to that of the 16-bit IMRm register, and the higher 8-bit register IMRmH can be accessed on the following address (address (IMRm) + 1).

Figure 8-11: Interrupt Mask Registers 0 to 3 (IMR0 to IMR3)

	15	14	13	12	11	10	9	8	Address	Initial value
IMR0	GCC03MK	GCC02MK	GCC01MK	GCC00MK	TMG01MK	TMG00MK	P5MK	P4MK	FFFFF100H	FFFFH
	7	6	5	4	3	2	1	0		
	P3MK	P2MK	P1MK	P0MK	WTIMK	TMD1MK	TMD0MK	WTMK		
	15	14	13	12	11	10	9	8	Address	Initial value
IMR1	FC1RXMK	MACMK	ADMK	CCC01MK	CCC00MK	TMC0MK	GCC15MK	GCC14MK	FFFFF102H	FFFFH
	7	6	5	4	3	2	1	0		
	GCC13MK	GCC12MK	GCC11MK	GCC10MK	TMG11MK	TMG10MK	GCC05MK	GCC04MK		
	15	14	13	12	11	10	9	8	Address	Initial value
IMR2	SR0MK	SER0MK	CSI2MK	CSI1MK	CSI0MK	FC4ERMK	FC4TXMK	FC4RXMK	FFFFF104H	FFFFH
	7	6	5	4	3	2	1	0		
	FC3ERMK	FC3TXMK	FC3RXMK	FC2ERMK	FC2TXMK	FC2RXMK	FC1ERMK	FC1TXMK		
	15	14	13	12	11	10	9	8	Address	Initial value
IMR3	1	P21MK	P20MK	P15MK	P10MK	P05MK	P00MK	DOVFMK	FFFFF106H	FFFFH
	7	6	5	4	3	2	1	0		
	DMA3MK	DMA2MK	DMA1MK	DMA0MK	ST1MK	SR1MK	SER1MK	ST0MK		

Bit Position	Bit Name	Function
15 to 0	xxMK	Interrupt mask flag. 0: Interrupt servicing enabled 1: Interrupt servicing disabled (pending)

Remark: xx: Identification name of each peripheral unit (WT, TMD, P, TMG, GCC, AD, MAC, FC, CSI, UART, DMA)

8.3.6 In-service priority register (ISPR)

This register holds the priority level of the maskable interrupt currently acknowledged. When an interrupt request is acknowledged, the bit of this register corresponding to the priority level of that interrupt request is set to 1 and remains set while the interrupt is serviced.
When the RETI instruction is executed, the bit corresponding to the interrupt request having the highest priority is automatically reset to 0 by hardware. However, it is not reset to 0 when execution is returned from non-maskable interrupt servicing or exception processing.
This register is read-only in 8-bit or 1-bit units.

Figure 8-12: In-Service Priority Register (ISPR)

	7	6	5	4	3	2	1	0	Address	Initial value
ISPR	ISPR7	ISPR6	ISPR5	ISPR4	ISPR3	ISPR2	ISPR1	ISPR0	FFFFF19AH	00H

Bit Position	Bit Name	Function
7 to 0	ISPR7 to ISPR0	Indicates priority of interrupt currently acknowledged 0: Interrupt request with priority n not acknowledged 1: Interrupt request with priority n acknowledged

Remark: n = 0 to 7 (priority level)

8.3.7 Maskable interrupt status flag (ID)

The ID flag is bit 5 of the PSW and this controls the maskable interrupt's operating state, and stores control information regarding enabling or disabling of interrupt requests.

Figure 8-13: Maskable Interrupt Status Flag (ID)

	31																							8	7	6	5	4	3	2	1	0	Initial value
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

Bit Position	Bit Name	Function
5	ID	Indicates whether maskable interrupt processing is enabled or disabled. 0: Maskable interrupt request acknowledgement enabled 1: Maskable interrupt request acknowledgement disabled (pending) This bit is set to 1 by the DI instruction and reset to 0 by the EI instruction. Its value is also modified by the RETI instruction or LDSR instruction when referencing to PSW. Non-maskable interrupt requests and exceptions are acknowledged regardless of this flag. when a maskable interrupt is acknowledged, the ID flag is automatically set to 1 by hardware. The interrupt request generated during the acknowledgement disabled period (ID = 1) is acknowledged when the PIFn bit of PICn register is set to 1, and the ID flag is reset to 0.

8.4 Noise Elimination Circuit

V850E/CA2 is provided with input filter for noise suppression for ports P3 to P5 and on all interrupt and timer G, timer C control inputs. For peripheral interrupts, programmable edge detection is available. Inputs for Timer G are equipped with edge detection and need only noise suppression.

Figure 8-14: Port Interrupt Input Circuit (P52, P53, P61, P62, P63, P64)

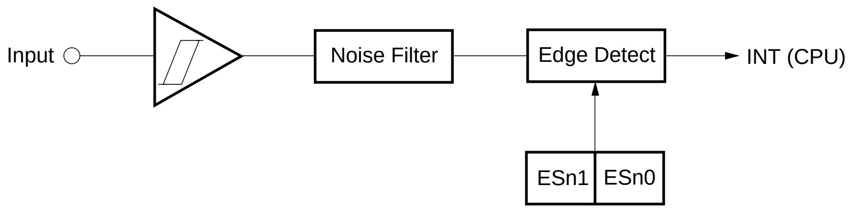


Figure 8-15: Timer G Input Circuit (P30, P35, P40, P45, P54, P55)

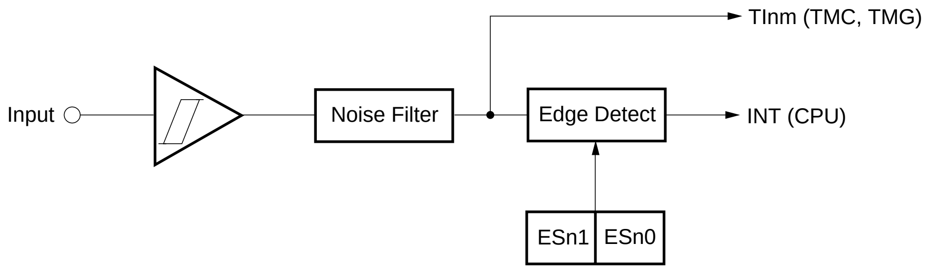
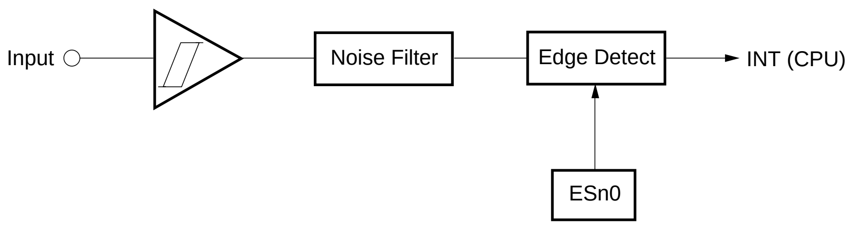


Figure 8-16: NMI Input Circuit



Note: Edge select circuit is different for NMI and INTPn. NMI can only configured as rising or falling edge sensitive whereas INTPn can be triggered by rising, falling or both edges.

8.4.1 Analog Filter

The analog filter consists of a comparator stage, which compares the input pin level against a delayed input pin level. The filter output follows the filter input, if this compare operation matches.

8.4.2 Interrupt Trigger Mode Selection

The valid edge of the INTP pins can be selected by the program. The edge that can be selected as the valid edge is one of the following.

- Rising edge
- Falling edge
- Both, the rising and the falling edges

8.4.3 Interrupt Edge Detection Control Registers

Valid interrupt edges can be selected by INTM0 to INTM3 registers. Masking of interrupts is done inside the concerning interrupt control registers xxIC.

(1) Interrupt mode register 0 (INTM0)

Figure 8-17: Interrupt Mode Register 0 (IMTM0)

	7	6	5	4	3	2	1	0	Address	Initial value
INTM0	ES031	ES030	ES021	ES020	ES011	ES010	ES001	ES000	FFFFF880H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7, 6</td><td>ES031, ES030</td><td>Edge selection for INTP3 to interrupt controller.
Selects active edge for interrupt generation.
<table>
<tr><th>ES031</th><th>ES030</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>5, 4</td><td>ES021, ES020</td><td>Edge selection for INTP2 to interrupt controller.
<table>
<tr><th>ES021</th><th>ES020</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>3, 2</td><td>ES011, ES010</td><td>Edge selection for INTP1 to interrupt controller.
<table>
<tr><th>ES011</th><th>ES010</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>1, 0</td><td>ES001, ES000</td><td>Edge selection for INTP0 to interrupt controller.
<table>
<tr><th>ES001</th><th>ES000</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
</table>

Note: Programming edge detection or port mode register can trigger unintended interrupt requests. Therefore be sure to mask the respective interrupt requests.

(2) Interrupt mode register 1 (INTM1)

Figure 8-18: Interrupt Mode Register 1 (IMTM1)

	7	6	5	4	3	2	1	0	Address	Initial value
INTM1	ES071	ES070	ES061	ES060	ES051	ES050	ES041	ES040	FFFFF882H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7, 6</td><td>ES071,
ES070</td><td>Edge selection for INTP05 to interrupt controller.
Selects active edge for interrupt generation.
<table>
<tr><th>ES071</th><th>ES070</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>5, 4</td><td>ES061,
ES060</td><td>Edge selection for INTP00 to interrupt controller.
<table>
<tr><th>ES061</th><th>ES060</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>3, 2</td><td>ES051,
ES050</td><td>Edge selection for INTP5 to interrupt controller.
<table>
<tr><th>ES051</th><th>ES050</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>1, 0</td><td>ES041,
ES040</td><td>Edge selection for INTP4 to interrupt controller.
<table>
<tr><th>ES041</th><th>ES040</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
</table>

Note: Programming edge detection or port mode register can trigger unintended interrupt requests. Therefore be sure to mask the respective interrupt requests.

(3) Interrupt mode register 2 (INTM2)

Figure 8-19: Interrupt Mode Register 2 (IMTM2)

	7	6	5	4	3	2	1	0	Address	Initial value
INTM2	ES111	ES110	ES101	ES100	ES091	ES090	ES081	ES080	FFFFF884H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7, 6</td><td>ES111, ES110</td><td>Edge selection for INTP21 to interrupt controller.
Selects active edge for interrupt generation.
<table>
<tr><th>ES111</th><th>ES110</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>5, 4</td><td>ES101, ES100</td><td>Edge selection for INTP20 to interrupt controller.
<table>
<tr><th>ES101</th><th>ES100</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>3, 2</td><td>ES091, ES090</td><td>Edge selection for INTP15 to interrupt controller.
<table>
<tr><th>ES091</th><th>ES090</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
<tr><td>1, 0</td><td>ES081, ES080</td><td>Edge selection for INTP10 to interrupt controller.
<table>
<tr><th>ES081</th><th>ES080</th><th>Edge selection</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Reserved</td></tr>
<tr><td>1</td><td>1</td><td>Both edges</td></tr>
</table></td></tr>
</table>

Note: Programming edge detection or port mode register can trigger unintended interrupt requests. Therefore be sure to mask the respective interrupt requests.

(4) Interrupt mode register 3 (INTM3)

Figure 8-20: Interrupt Mode Register 3 (IMTM3)

	7	6	5	4	3	2	1	0	Address	Initial value
INTM3								ESN0	FFFFF886H	00H

Bit Position	Bit Name	Function
0	ESN0	Edge selection for NMI. Selects active edge for interrupt generation. 0: Falling edge. 1: Rising edge.

Notes: 1. NMI functionality is masked by PMC60. Selection of valid edge for NMI must be performed while PMC60 is “0”.

2. Install appropriate interrupt handler for NMI before reprogramming edge detection or port function.

8.5 Software Exception

A software exception is generated when the CPU executes the TRAP instruction, and can be always acknowledged.

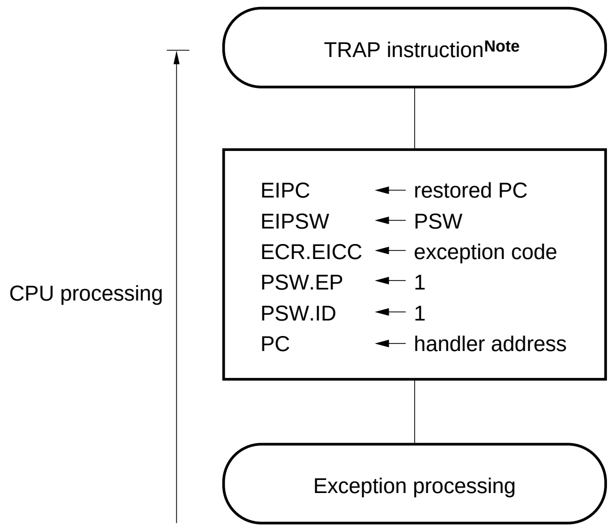
8.5.1 Operation

If a software exception occurs, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the restored PC to EIPC.
(2) Saves the current PSW to EIPSW.
(3) Writes an exception code to the lower 16 bits (EICC) of ECR (interrupt source).
(4) Sets the EP and ID bits of the PSW.
(5) Sets the handler address (00000040H or 00000050H) corresponding to the software exception to the PC, and transfers control.

Figure 8-21 illustrates the processing of a software exception.

Figure 8-21: Software Exception Processing



Note: TRAP Instruction Format: TRAP vector (the vector is a value from 0 to 1FH.)

The handler address is determined by the TRAP instruction's operand (vector). If the vector is 0 to 0FH, it becomes 00000040H, and if the vector is 10H to 1FH, it becomes 00000050H.

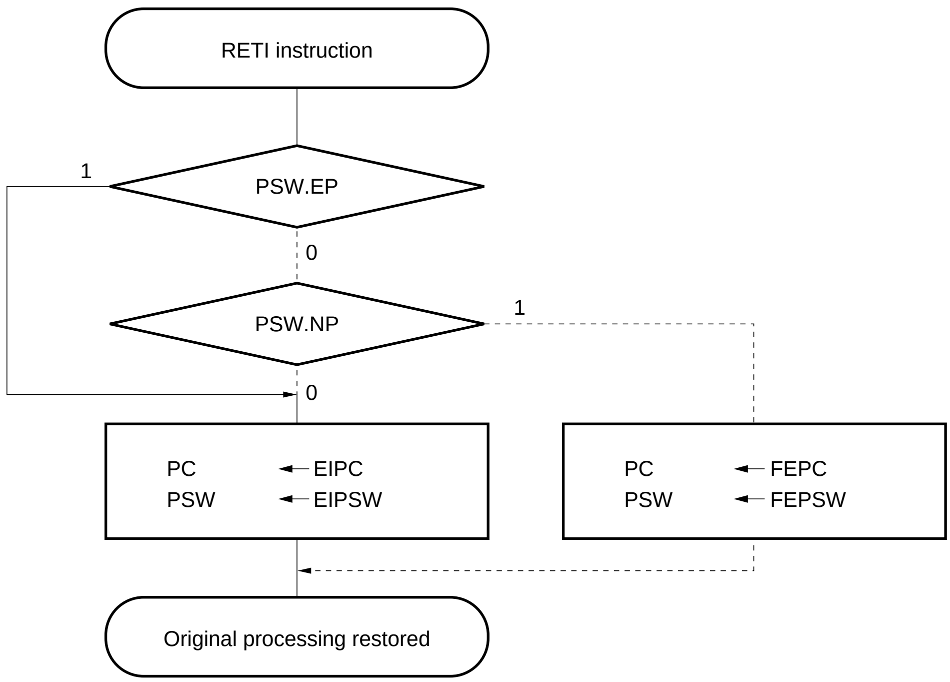
8.5.2 Restore

Recovery from software exception processing is carried out by the RETI instruction.
By executing the RETI instruction, the CPU carries out the following processing and shifts control to the restored PC's address.

(1) Loads the restored PC and PSW from EIPC and EIPSW because the EP bit of the PSW is 1.
(2) Transfers control to the address of the restored PC and PSW.

Figure 8-22 illustrates the processing of the RETI instruction.

Figure 8-22: RETI Instruction Processing



Caution: When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during the software exception processing, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 1 using the LDSR instruction immediately before the RETI instruction.

Remark: The solid lines show the CPU processing flow.

8.5.3 Exception status flag (EP)

The EP flag is bit 6 of PSW, and is a status flag used to indicate that exception processing is in progress. It is set when an exception occurs.

Figure 8-23: Exception Status Flag (EP)

	31																							8	7	6	5	4	3	2	1	0	Initial value
PSW	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	NP	EP	ID	SAT	CY	OV	S	Z	00000020H

Bit Position	Bit Name	Function
6	EP	Shows that exception processing is in progress. 0: Exception processing not in progress. 1: Exception processing in progress.

8.6 Exception Trap

An exception trap is an interrupt that is requested when an illegal execution of an instruction takes place. In the V850E/CA2, an illegal opcode exception (ILGOP: Illegal Opcode Trap) is considered as an exception trap.

8.6.1 Illegal opcode definition

The illegal instruction has an opcode (bits 10 to 5) of 111111B, a sub-opcode (bits 23 to 26) of 0111B to 1111B, and a sub-opcode (bit 16) of 0B. An exception trap is generated when an instruction applicable to this illegal instruction is executed.

15 … 11	10 … 5	4 … 0	31 … 27	26 … 23	22 …	16
× × × × ×	1 1 1 1 1 1	× × × × ×	× × × × ×	0 1 1 1 to 1 1 1 1	× × × × × ×	0

Remark: ×: Arbitrary

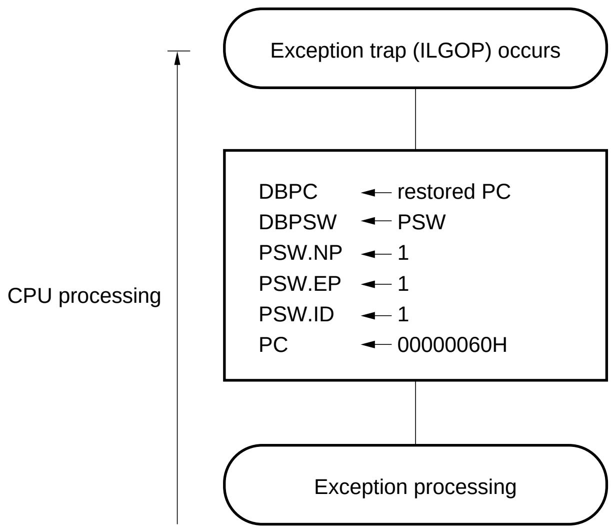
(1) Operation

If an exception trap occurs, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the restored PC to DBPC.
(2) Saves the current PSW to DBPSW.
(3) Sets the NP, EP, and ID bits of the PSW.
(4) Sets the handler address (00000060H) corresponding to the exception trap to the PC, and transfers control.

Figure 8-24 illustrates the processing of the exception trap.

Figure 8-24: Exception Trap Processing



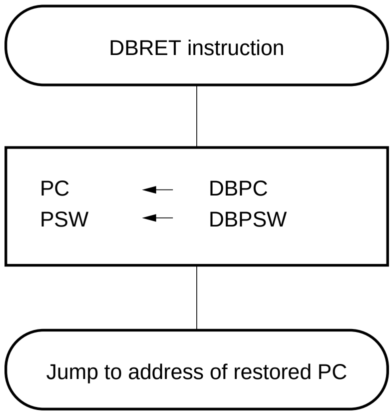
(2) Restore

Recovery from an exception trap is carried out by the DBRET instruction. By executing the DBRET instruction, the CPU carries out the following processing and controls the address of the restored PC.

(1) Loads the restored PC and PSW from DBPC and DBPSW.
(2) Transfers control to the address indicated by the restored PC and PSW.

Figure 8-25 illustrates the restore processing from an exception trap.

Figure 8-25: Restore Processing from Exception Trap



8.6.2 Debug trap

The debug trap is an exception that can be acknowledged every time and is generated by execution of the DBTRAP instruction.
When the debug trap is generated, the CPU performs the following processing.

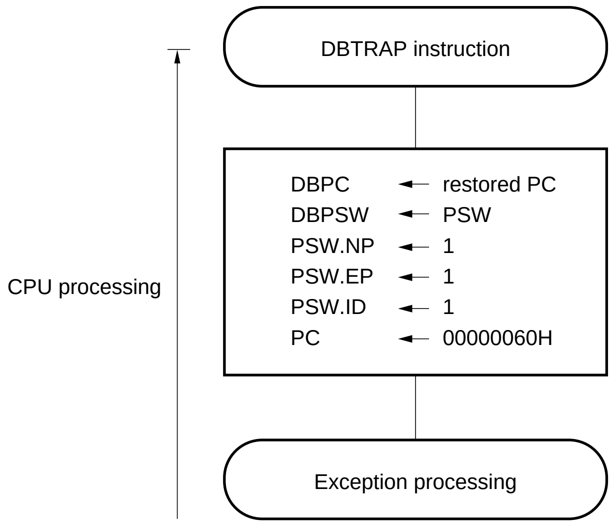
(1) Operation

When the debug trap is generated, the CPU performs the following processing, transfers control to the debug monitor routine, and shifts to debug mode.

(1) Saves the restored PC to DBPC.
(2) Saves the current PSW to DBPSW.
(3) Sets the NP, EP and ID bits of the PSW.
(4) Sets the handler address (00000060H) corresponding to the debug trap to the PC and transfers control.

Figure 8-26 illustrates the processing of the debug trap.

Figure 8-26: Debug Trap Processing



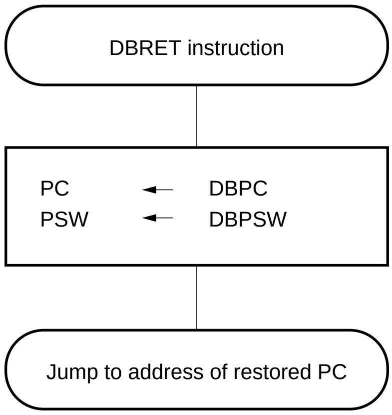
(2) Restore

Recovery from a debug trap is carried out by the DBRET instruction. By executing the DBRET instruction, the CPU carries out the following processing and controls the address of the restored PC.

(1) Loads the restored PC and PSW from DBPC and DBPSW.
(2) Transfers control to the address indicated by the restored PC and PSW.

Figure 8-27 illustrates the restore processing from a debug trap.

Figure 8-27: Restore Processing from Debug Trap



8.7 Multiple Interrupt Processing Control

Multiple interrupt processing control is a process by which an interrupt request that is currently being processed can be interrupted during processing if there is an interrupt request with a higher priority level, and the higher priority interrupt request is received and processed first.
If there is an interrupt request with a lower priority level than the interrupt request currently being processed, that interrupt request is held pending.
Maskable interrupt multiple processing control is executed when an interrupt has an enable status (ID = 0). Thus, if multiple interrupts are executed, it is necessary to have an interrupt enable status (ID = 0) even for an interrupt processing routine.
If a maskable interrupt enable or a software exception is generated in a maskable interrupt or software exception service program, it is necessary to save EIPC and EIPSW.
This is accomplished by the following procedure.

(1) Acknowledgment of maskable interrupts in service program

Service program of maskable interrupt or exception

```
...
...
• EIPC saved to memory or register
• EIPSW saved to memory or register
• EI instruction (interrupt acknowledgment enabled)
...
...                                                   ¨ Maskable interrupt acknowledgment
...
...
• DI instruction (interrupt acknowledgment disabled)
• Saved value restored to EIPSW
• Saved value restored to EIPC
• RETI instruction
```

(2) Generation of exception in service program

Service program of maskable interrupt or exception

...	
...	
• EIPC saved to memory or register	
• EIPSW saved to memory or register	
...	
• TRAP instruction	¨ Exception such as TRAP instruction acknowledged.
...	
• Saved value restored to EIPSW	
• Saved value restored to EIPC	
• RETI instruction	

The priority order for multiple interrupt processing control has 8 levels, from 0 to 7 for each maskable interrupt request (0 is the highest priority), but it can be set as desired via software. Setting of the priority order level is done using the PPRn0 to PPRn2 bits of the interrupt control request register (PICn), which is provided for each maskable interrupt request. After system reset, an interrupt request is masked by the PMKn bit and the priority order is set to level 7 by the PPRn0 to PPRn2 bits.

The priority order of maskable interrupts is as follows.

(High) Level 0 > Level 1 > Level 2 > Level 3 > Level 4 > Level 5 > Level 6 > Level 7 (Low)

Interrupt processing that has been suspended as a result of multiple processing control is resumed after the processing of the higher priority interrupt has been completed and the RETI instruction has been executed.
A pending interrupt request is acknowledged after the current interrupt processing has been completed and the RETI instruction has been executed.

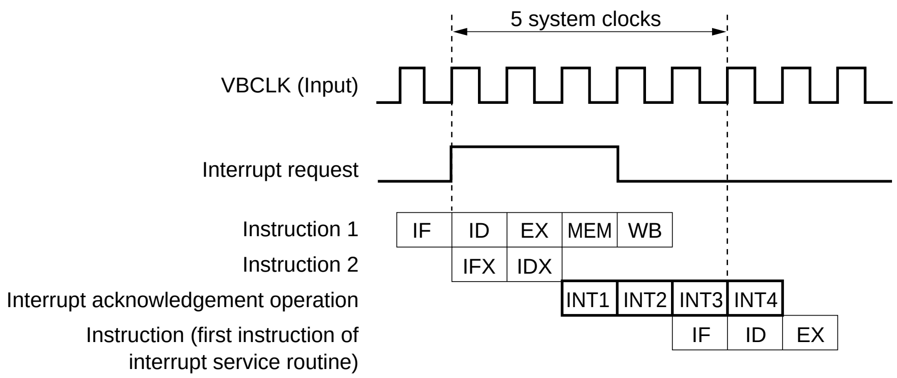
Caution: In a non-maskable interrupt processing routine (time until the RETI instruction is executed), maskable interrupts are suspended and not acknowledged.

8.8 Interrupt Response Time

The following table describes the V850E/CA2 interrupt response time (from interrupt generation to start of interrupt processing).
Except in the following cases, the interrupt response time is a minimum of 5 clocks. To input interrupt requests continuously, leave a space of at least 5 clocks between interrupt request inputs.

- During software or hardware STOP mode
- When an external bus is accessed
- When there are two or more successive interrupt request non-sampling instructions (see 8.9 “Periods in Which Interrupts Are Not Acknowledged” on page 237).
- When the interrupt control register is accessed

Figure 8-28: Pipeline Operation at Interrupt Request Acknowledgment (Outline)



Remark: INT1 to INT4: Interrupt acknowledgement processing
IFx: Invalid instruction fetch
IDx: Invalid instruction decode

Table 8-3: Interrupt Response Time

Interrupt Response Time (Internal System Clocks)			Condition
	Internal Interrupt	External interrupt	
Minimum	5	5 + analog delay time	The following cases are exceptions: • In IDLE/software STOP mode • External bit access • Two or more interrupt request non-sample instructions are executed • Access to interrupt control register
Maximum	11	11 + analog delay time	

8.9 Periods in Which Interrupts Are Not Acknowledged

An interrupt is acknowledged while an instruction is being executed. However, no interrupt will be acknowledged between an interrupt non-sample instruction and the next instruction.
The interrupt request non-sampling instructions are as follows.

- EI instruction
- DI instruction
- LDSR reg2, 0x5 instruction (for PSW)
- The store instruction for the interrupt control register (PICn), in-service priority register (ISPR), and command register (PRCMD).

[MEMO]

Chapter 9 Clock Generator

9.1 Features

- Multiplication function by PLL synthesizer
  - Spread Spectrum PLL for CPU/BCU clock supply

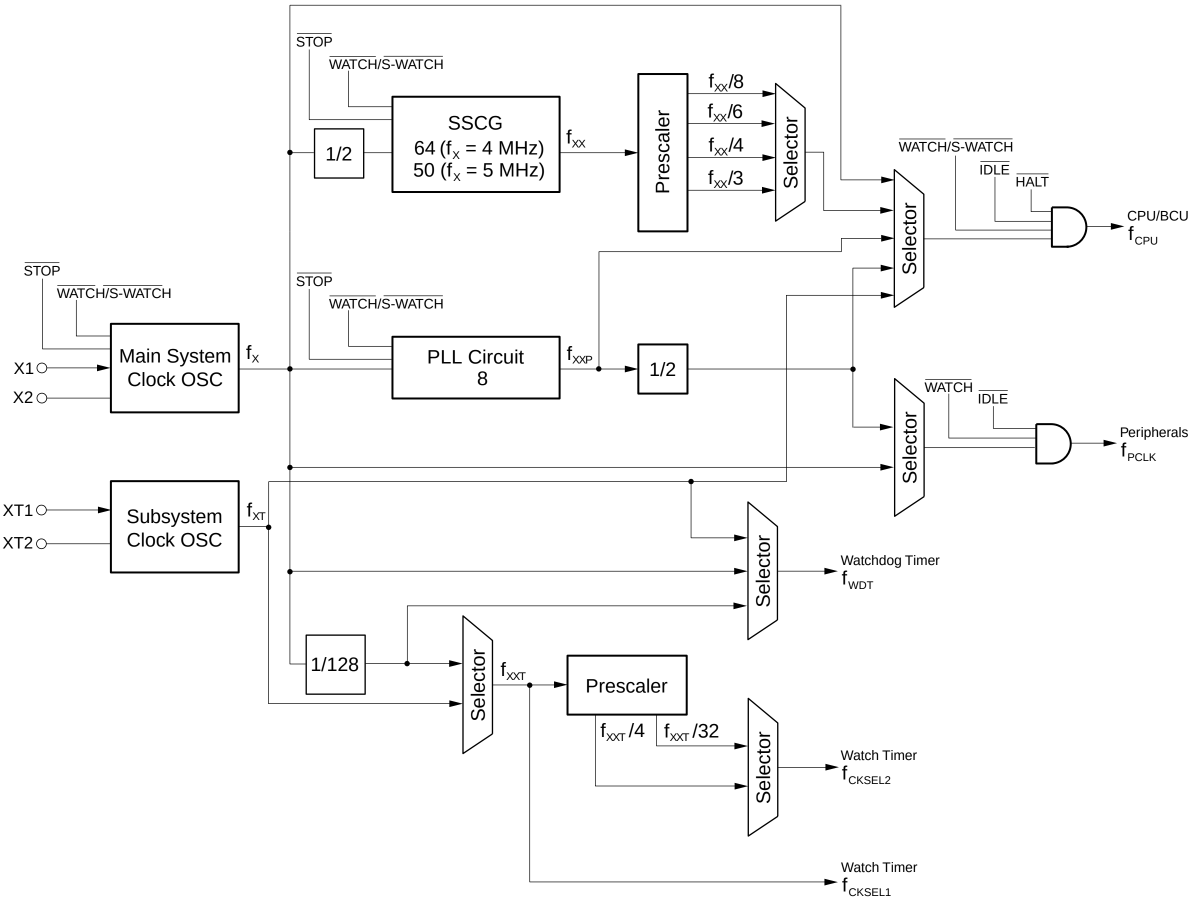
- Clock sources
  - Oscillation through oscillator connection
  - Oscillation through sub-oscillator connection during sub-watch-mode

- Power save modes
  - WATCH mode
  - Sub-WATCH mode
  - HALT mode
  - IDLE mode
  - STOP mode

- low power sub-clock for watch timer and watchdog timer to reduce power consumption in watch mode.

9.2 Configuration

Figure 9-1: Block Diagram of the Clock Generator



This block diagram does not necessarily show the exact wiring in hardware but the functional structure.

9.3 Control Registers

9.3.1 Clock Control Register (CKC)

This is an 8-bit register that controls the clock management. Data can be written to it only in a sequence of specific instructions so that its contents are not easily rewritten in case of program hang-up. See also PHCMD register.

This register can be read or written in 8-bit units.

Figure 9-2: Clock Control Register (CKC) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
CKC	PLLEN	SCEN	DEN	0	PERIC	0	WTSEL1	WTSEL0	FFFFF822H	00H

Bit name	Function
PLLEN	PLLEN enable bit This bit enables or disables PLL operation. 0: PLL disabled 1: PLL enabled **Caution: The PLL is enabled when this bit is set (1). Before applying the PLL clock as the clock supply for the CPU or the peripherals, it must have been secured by software that the PLL stabilization time (1ms) has been elapsed. During this stabilization time, the software must remain in a loop and the CPU and the peripherals are supplied by the main-oscillator clock. Switching to an unstable clock source is not protected by hardware.**
SCEN	SSCG enable bit This bit enables or disables Spread-Spectrum-Clock-Generation 0: SSCG disabled 1: SSCG enabled **Caution: The SSCG is enabled when this bit is set (1). Before applying the SSCG clock as the clock supply for the CPU, it must have been secured that the SSCG stabilization time has been elapsed. The SCSTAT bit in the CGSTAT register shows the status of the SSCG. The value of the read-only bit SCSTAT must be read as set (1) before enabling the SSCG clock to the CPU. Switching to an unstable clock source is not protected by hardware.**
DEN	SSCG frequency dithering enable bit 0: SSCG uses fixed multiplication factor of SCFC1 1: SSCG uses multiplication factor of SCFC0 and SCFC1 alternately **Caution: The DEN bit can be toggled only in case that the SCEN bit is 0 (SSCG disabled).**
PERIC	Peripheral clock source select bit 0: Main oscillator (x1) is clock source for peripherals 1: PLL (x4) is clock source for peripherals

Figure 9-2: Clock Control Register (CKC) (2/2)

Bit name	Function
WTSEL1	Sub-clock source select bit 0: Main oscillator/128 is clock source for sub-clock 1: Sub-oscillator is clock source for sub-clock
WTSEL0	Sub-clock divider select for f_{CKSEL2} 0: f_{CKSEL2} = sub-clock/4 1: f_{CKSEL2} = sub-clock/32

Cautions: 1. Data is set to the CKC register by the following sequence:

- **Write the set data to the command register (PHCMD)**
  (see Chapter 3.6.2 "Peripheral Command Register (PHCMD)" on page 105).
- **Write the set data to the destination register (CKC)**

2. If PLL or SSCG operation is required, the PLLEN bit and the SCEN bit are allowed to be set (1) when the system remains in the main-oscillation mode (CPU and peripherals are using the main-oscillator as the clock supply).

To write data to the CKC register, use the store instruction (ST/SST) and bit manipulation instruction (SET1/CLR1/NOT1).
The contents of this register can be read in the normal sequence.

9.3.2 Clock Generator Status Register (CGSTAT)

This is an 8-bit register that monitors the status of the SSCG and main oscillator hard macro operation.

This register can be read in 8- or 1-bit units.

Figure 9-3: Clock Generator Status Register (CGSTAT)

	7	6	5	4	3	2	1	0	Address	Initial value
CGSTAT	0	0	0	0	0	0	OSCSTAT	SCSTAT	FFFFF824H	00H

Bit name	Function
OSCSTAT	Main-clock stabilization indication bit (determined by counter) 0: Main-oscillator is not stabilized 1: Main-oscillator is stabilized
SCSTAT	SSCG lock status (determined by SSCG Lock signal) 0: SSCG is not stabilized 1: SSCG is stabilized

9.3.3 Watchdog Timer Clock Control Register (WCC)

This is an 8-bit register that controls the watchdog timer clock. Data can be written to it only in a sequence of specific instructions so that its contents are not easily rewritten in case of program hang-up. See also PHCMD register.

This register can be read or written in 8-bit units.

Figure 9-4: Watchdog Timer Clock Control Register (WCC)

	7	6	5	4	3	2	1	0	Address	Initial value
WCC	0	0	0	0	0	0	WDTSEL1	WDTSEL0	FFFFF826H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>1, 0</td><td>WDTSEL1,
WDTSEL0</td><td>Specifies the clock source for the Watchdog
<table>
<tr><th>WDTSEL1</th><th>WDTSEL0</th><th>Watchdog clock source</th></tr>
<tr><td>0</td><td>0</td><td>Main oscillator (f_X)</td></tr>
<tr><td>0</td><td>1</td><td>Sub oscillator (f_{XT})</td></tr>
<tr><td>1</td><td>0</td><td>Main oscillator /128 (f_X/128)</td></tr>
<tr><td>1</td><td>1</td><td>reserved</td></tr>
</table></td></tr>
</table>

Caution: Data is set to the registers by the following sequence:

- **Write the set data to the command register (PHCMD) (see Chapter 3.6.2 “Peripheral Command Register (PHCMD)” on page 105).**
- **Write the set data to the destination register (WCC)**

Remarks: 1. If it is required to switch to another WDT clock source, it is recommended to monitor the status of the concerned clock source to be selected before. Switching to an unstable clock source is not protected by hardware.

2. The WCC register should be programmed immediately after occurrence of a system Reset, even in the case that the default settings are intended to be used.

3. It is possible to change the contents of the WCC register only for one time after the occurrence of a Reset.

9.3.4 Processor Clock Control Register (PCC)

This is an 8-bit register that controls the CPU clock. Data can be written to it only in a sequence of specific instructions so that its contents are not easily rewritten in case of program hang-up. See also PHCMD register.
The setup of this register can be changed only one time. After a power save mode has been released, the CPU clock is supplied by the main oscillator and again a new clock source can be selected.

This register can be read or written in 8-bit units.

Figure 9-5: Processor Clock Control Register (PCC) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
PCC	FRC	0	MFRC	CLS	0	0	CKS1	CKS0	FFFFF828H	00H

Bit name	Function
FRC	Sub-system clock oscillation circuit control of internal return resistance. 0: Resistance connected 1: Resistance disconnected **Remark:** The FRC bit must always remain cleared (0) while sub-system clock operation is enabled.
MFRC	Main-system clock oscillation circuit control of internal return resistance 0: Resistance connected 1: Resistance disconnected **Remark:** The initial setting of the MFRC bit must not be changed at anytime. To secure proper operation of the main-system clock, the internal feed-back resistance must remain connected always.

Figure 9-5: Processor Clock Control Register (PCC) (2/2)

Bit name	Function			
CLS, CKS1, CKS0	Specifies the CPU clock source			
	CLS	CKS1	CKS0	CPU Clock
	0	0	0	Main oscillator
	0	0	1	SSCG
	0	1	0	PLL (Main oscillator frequency × 4)
	0	1	1	PLL (Main oscillator frequency × 8)
	1	X	X	Sub-Oscillator

Note: X: don’t care

Caution: Data is set to the registers by the following sequence:

- **Write the set data to the command register (PHCMD) (see Chapter 3.6.2 “Peripheral Command Register (PHCMD)” on page 105).**
- **Write the set data to the destination register (PCC)**

Remarks:

1. If it is required to switch to another CPU clock source, it is recommended to monitor the status of the clock source to be selected before. Switching to an unstable clock source is not protected by hardware.

2. It is only possible to change the contents of the PCC register for one time after the occurrence of a Reset or if a power-save mode has been released.

3. After release from Watch mode, Idle mode or Stop mode the register PCC is set to Main oscillator mode.
After release from Sub-Watch mode the register PCC is set to Main oscillator mode in case that the bit OSCDIS is cleared (0) or the register PCC is set to Sub-Oscillator mode if the bit OSCDIS is set (1). The bit OSCDIS can be found in the register Power Save Mode PSM.

9.3.5 Reset Source Monitor Register (RSM)

This is a 8-bit register that indicates the source of the last system reset.

This register can only be read in 8- or 1-bit units.

Figure 9-6: Reset Source Monitor Register (RSM)

	7	6	5	4	3	2	1	0	Address	Initial value
RSM	0	0	0	0	0	0	0	RESM	FFFFF830H	00/01H

Bit name	Function
RESM	Reset Source Monitor flag 0: Last Reset was caused by external $\overline{\text{RESET}}$ input 1: Last Reset was caused by internal Watchdog timer overflow

9.3.6 SSCG Frequency Modulation Control Register (SCFMC)

This is a 5-bit register that controls the frequency modulation of SSCG in dithering mode and the post scale factor of the SSCG.

This register can be read or written in 8- or 1-bit units.

Figure 9-7: SSCG Frequency Modulation Control Register (SCFMC)

	7	6	5	4	3	2	1	0	Address	Initial value
SCFMC	0	SCPS1	SCPS0	SCFMC4	SCFMC3	SCFMC2	SCFMC1	SCFMC0	FFFFF82AH	0AH

<table>
<tr><th>Bit name</th><th>Function</th></tr>
<tr><td>SCPS1,
SCPS0</td><td>Frequency modulation control bits
<table>
<tr><th>SCPS1</th><th>SCPS0</th><th>Post Scale Factor of the SSCG</th></tr>
<tr><td>0</td><td>0</td><td>$f_{XX}/3$
Note: If SSCG operation is required for the CPU/BCU clock supply, the setting of the bits SCPS0, SCPS1 = 0x00 is not supported at any time. Therefore, the setting of these bits must be modified before the SSCG is enabled.</td></tr>
<tr><td>0</td><td>1</td><td>$f_{XX}/4$</td></tr>
<tr><td>1</td><td>0</td><td>$f_{XX}/6$</td></tr>
<tr><td>1</td><td>1</td><td>$f_{XX}/8$</td></tr>
</table></td></tr>
<tr><td>SCFMC4
to
SCFMC0</td><td>Specifies the dithering frequency

The initial setting (0x0A) of these bits must not be changed at any time.</td></tr>
</table>

Cautions: 1. This register can only be written if the SSCG enable bit SCEN is cleared.

2. After the first initialization of the SCFMC register, no further write access is allowed until the occurrence of a Reset or the release of a power-save mode happened. Afterwards a power-save mode has been released, one bit is allowed to be changed.

9.3.7 SSCG Frequency Control Register 0 (SCFC0)

This is an 8-bit register that controls the first frequency divider of the SSCG. It determines the lower SSCG output frequency in dithering mode.

This register can be read or written in 8- or 1-bit units.

Figure 9-8: SSCG Frequency Control Register 0 (SCFC0)

	7	6	5	4	3	2	1	0	Address	Initial value
SCFC0	SCFC07	SCFC06	SCFC05	SCFC04	SCFC03	SCFC02	SCFC01	SCFC00	FFFFF82CH	3FH

Bit Position	Bit Name	Function
7 to 0	SCFC07 to SCFC00	Specifies the first frequency divider of the SSCG **f_X = 4 MHz:** SCFC07 to SCFC00: 003EH — Lower SSCG frequency f_{XX}: 126 MHz **f_X = 5 MHz:** SCFC07 to SCFC00: 0031H — Lower SSCG frequency f_{XX}: 125 MHz The initialization of the SCFC0 register depends to the output frequency supplied by the main oscillation circuit. The values mentioned above must not be changed after initialization.

f_X = 4 MHz:

SCFC07 to SCFC00	Lower SSCG frequency f_{XX}
003EH	126 MHz

f_X = 5 MHz:

SCFC07 to SCFC00	Lower SSCG frequency f_{XX}
0031H	125 MHz

Caution: This register can only be written if the SSCG enable bit SCEN is cleared.

9.3.8 SSCG Frequency Control Register 1 (SCFC1)

This is an 8-bit register that controls the second frequency divider of the SSCG. It determines the SSCG output frequency in fixed frequency mode and the upper SSCG output frequency in dithering mode.

This register can be read or written in 8-bit or 1-bit units.

Figure 9-9: SSCG Frequency Control Register 1 (SCFC1)

	7	6	5	4	3	2	1	0	Address	Initial value
SCFC1	SCFC17	SCFC16	SCFC15	SCFC14	SCFC13	SCFC12	SCFC11	SCFC10	FFFFF82EH	40H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7 to 0</td><td>SCFC17 to SCFC10</td><td>Specifies the second frequency divider of the SSCG

f_X = 4 MHz:
<table><tr><th>SCFC17 to SCFC10</th><th>Fixed or Upper SSCG frequency f_{XX}</th></tr><tr><td>003FH</td><td>128 MHz</td></tr></table>
f_X = 5 MHz:
<table><tr><th>SCFC17 to SCFC10</th><th>Fixed or Upper SSCG frequency f_{XX}</th></tr><tr><td>0032H</td><td>127.5 MHz</td></tr></table>
The initialization of the SCFC1 register depends to the output frequency supplied by the main oscillation circuit. The values mentioned above must not be changed after initialization.</td></tr>
</table>

Caution: This register can only be written if the SSCG enable bit SCEN is cleared.

9.4 Power Saving Functions

9.4.1 General

The device provides the following power saving functions. These modes can be combined and switched to suit the target application, which enables effective implementation of low-power systems.

The device provides the following power saving functions. These modes can be combined and switched to suit the target application, which enables effective implementation of low-power systems.

Table 9-1: Power Saving Modes Overview

Clock Source		Mode	Operation of			Clock Supply to		
			Oscillator		SSCG/ PLL	Peripherals	CPU	Watch
			Main Osc.	Sub Osc.				
OSC mode	Initial Mode	Normal	×	×	–	×	×	×
	SSCG/PLL enabled	Normal	×/– **Note 1**	×	×	×	×	×
		HALT	×/– **Note 1**	×	×	×	–	×
		IDLE	¥	×	×	–	–	×
		WATCH	¥	×	–	–	–	×
		SUB WATCH	–	×	–	–	–	×
		STOP	–	–/× **Note 2**	–	–	–	–

Remarks: 1. ×: Operates

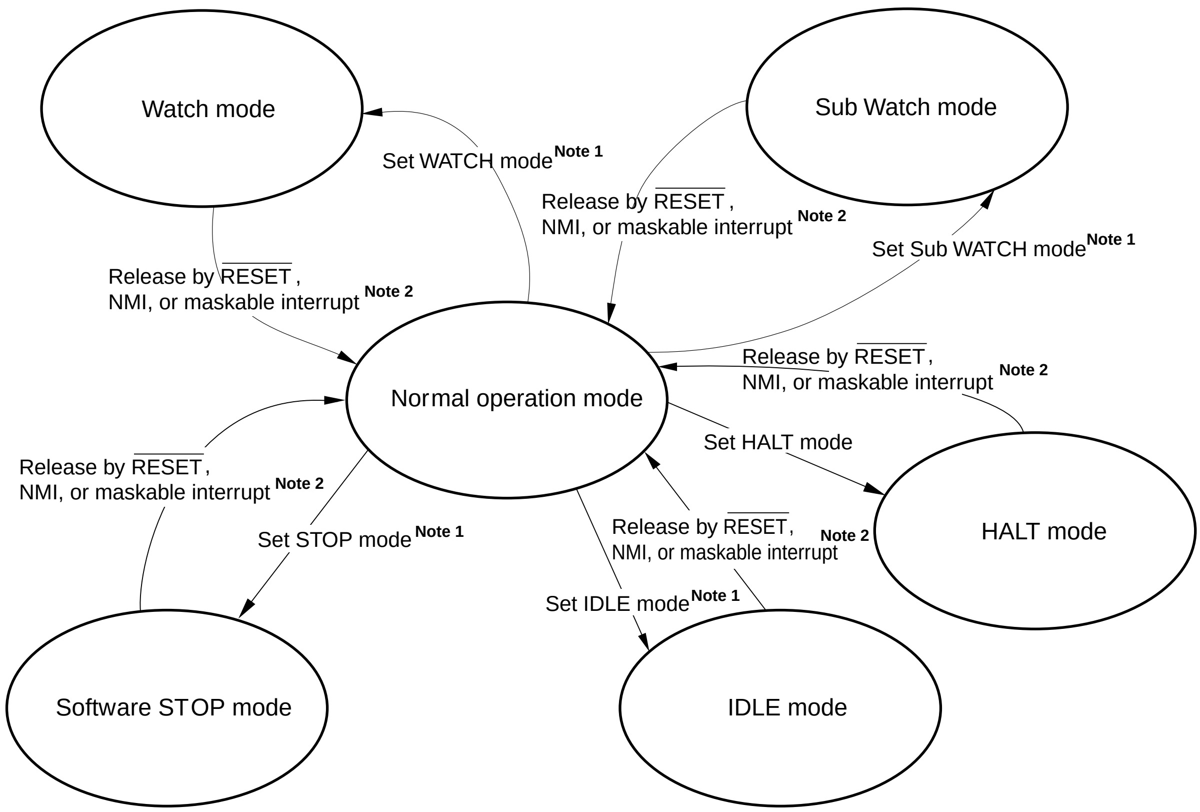
2. –: Stopped

Notes: 1. If the OSCDIS bit = 1, than the Main Oscillator is stopped.

2. If the SOSTP bit = 0, than the Sub Oscillator operates

Figure 9-10 shows the operation of the clock generator in normal operation mode, HALT mode, IDLE mode, WATCH mode, SUB WATCH mode and software STOP mode.
An effective low power consumption system can be realized by combining these modes and switching modes according to the required use.

Figure 9-10: Power Save Mode State Transition Diagram



Notes: 1. The SSCG and PLL is deactivated per hardware.

2. Enable SSCG and PLL manual.

9.4.2 Power Save Modes Outline

V850E/CA2 Jupiter is provided with the following standby modes: HALT, IDLE, WATCH, and software STOP. Application systems, which are designed so that these modes are switched appropriately according to operation purposes, reduce power consumption efficiently.

(1) HALT mode:

In this mode supply of the operating clock to the CPU is stopped whereby other on-chip peripheral functions continue to operate. Combining this mode with the normal operating mode to provide intermittent operations enables the overall system power consumption to be reduced.
This mode is entered by executing the dedicated instruction (HALT).

(2) IDLE mode:

In this mode, the clock generator continues to operate but stopping the supply of internal system clock stops the overall system. As it is not necessary to secure the oscillation stabilization time, it is possible to switch to the normal operating mode quickly in response to a release signal.
This mode provides low power consumption, where the power is only consumed from the OSC (Main-oscillator, Sub-Oscillator) and Watch timer / Watchdog timer. This mode is entered by setting registers with software.

(3) WATCH mode:

In this mode, the clock generator (PLL and SSCG) stops operation. Therefore, the entire system excluding Watch timer / Watchdog timer unit stops. This mode provides low power consumption, where the power consumed is only from OSC (Main-oscillator, Sub-Oscillator) and Watch timer / Watchdog timer circuit. This mode is entered by setting registers with software.

(4) SUB-WATCH mode:

In this mode, the clock generator (PLL and SSCG) stops operation. Therefore, the entire system excluding Watch timer / Watchdog timer unit stops. This mode provides ultra-low power consumption, where the power consumed is only from the Sub-Oscillator and Watch timer / Watchdog timer circuit. This mode is entered by setting registers with software.

(5) Software STOP mode:

In this mode, the main clock generator is stopped and the entire system stops. This mode provides ultra-low power consumption, where the power consumed is only leakage current and Sub-Oscillator operation if a crystal is connected. This mode is entered by setting registers with software.

9.4.3 Power Saving Mode Functions

The Clock Controller supports 3 type of standby modes: IDLE, WATCH, STOP. The behaviour of all output clocks is described in the following tables.

Table 9-2: Power Saving Mode Functions

Macro	Condition (settings of important bits)	reset	normal	HALT	IDLE	release IDLE	STOP	release STOP	WATCH	release WATCH	SUB WATCH	release SUB WATCH
Main Oscillator	OSCDIS=0	on	on	on	on	on	off	on	on	on	off	on
	OSCDIS=1	N.A.	off	off	N.A.	N.A.	off	on	N.A.	N.A.	off	off
Sub Oscillator	SOSTP=1	on	on	on	on	on	off	on	on	on	on	on
	SOSTP=0	on	on	on	on	on	on	on	on	on	on	on
SSCG		off	SCEN **Note**	SCEN **Note**	SCEN **Note**	SCEN **Note**	off	off	off	off	off	off
PLL		off	PLLEN **Note**	PLLEN **Note**	PLLEN **Note**	PLLEN **Note**	off	off	off	off	off	off
CPU clock	CLS/CKS=000	f_X	f_X	f_X	off	f_X	off	f_X	off	f_X	off	f_X
	CLS/CKS=001	N.A.	f_{XX}	f_{XX}	off	N.A.	off	N.A.	off	N.A.	off	N.A.
	CLS/CKS=01x	N.A.	f_{XXP}	f_{XXP}	off	N.A.	off	N.A.	off	N.A.	off	N.A.
	CLS/CKS=1xx	N.A.	f_{XT}	f_{XT}	off	N.A.	off	N.A.	off	N.A.	off	f_{XT}
Peripheral clock	PERIC=0	f_X	f_X	f_X	off	f_X	off	f_X	off	f_X	off	f_X
	PERIC=1	N.A.	f_{XXP}	f_{XXP}	off	N.A.	off	N.A.	off	N.A.	off	N.A.
TMC clock	CMODE=0	f_{PCLK}	f_{PCLK}	f_{PCLK}	off	f_{PCLK}	off	f_{PCLK}	off	f_{PCLK}	off	f_{PCLK}
	CMODE=1	N.A.	f_X	f_X	f_X	f_X	off	f_X	f_X	f_X	off	f_X
Port	CLS/CKS=000	f_X	f_X	f_X	f_X	f_X	off	f_X	f_X	f_X	off	f_X
	CLS/CKS=001	N.A.	f_{XX}	f_{XX}	f_X	N.A.	N.A.	N.A.	f_X	N.A.	N.A.	N.A.
	CLS/CKS=01x	N.A.	f_{XXP}	f_{XXP}	f_X	N.A.	N.A.	N.A.	f_X	N.A.	N.A.	N.A.
	CLS/CKS=1xx	N.A.	f_{XT}	f_{XT}	f_X	N.A.	N.A.	N.A.	f_X	N.A.	f_{XT}	f_{XT}
Watchdog	WDTSEL=x0	off	f_X	f_X	f_X	f_X	off	f_X	f_X	f_X	off	f_X
	WDTSEL=01	N.A.	f_{XT}	f_{XT}	f_{XT}	f_{XT}	off	f_{XT}	f_{XT}	f_{XT}	f_{XT}	f_{XT}
Watch timer	WTSEL1=0	off	f_X	f_X	f_X	f_X	off	f_X	f_X	f_X	off	f_X
	WTSEL1=1	N.A.	f_{XT}	f_{XT}	f_{XT}	f_{XT}	off	f_{XT}	f_{XT}	f_{XT}	f_{XT}	f_{XT}

Remarks: **1.** f_X: main oscillator frequency

2. f_{XX}: SSCG output frequency

3. f_{XXP}: PLL output frequency

4. f_{XT}: sub oscillator frequency

5. f_{PCLK}: peripheral clock frequency

6. N.A.: not available

Note: The functionality of the SSCG or the PLL depends on the setting of the according bit.

Table 9-3: Power Saving Mode Functions

Pin Function	RESET	STOP	WATCH	SUB WATCH	IDLE	HALT
D[15:0]	Hi-Z	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	operate
A[23-0]	Hi-Z	HOLD	HOLD	HOLD	HOLD	operate
$\overline{CS}$[4:3, 0]	Hi-Z	H	H	H	H	operate
$\overline{WR}$[1:0]	Hi-Z	H	H	H	H	operate
$\overline{RD}$	Hi-Z	H	H	H	H	operate
$\overline{WAIT}$	---	---	---	---	---	operate
$\overline{RESOUT}$	LOW	HIGH	HIGH	HIGH	HIGH	HIGH
TIG05 to TIG00 TIG15 to TIG10 TIC01 to TIC00	N.A.	---	---	---	---	operate
INTP05, INTP00 INTP15, INTP10 INTP21, INTP20 INTP5 to INTP0,NMI	N.A.	operate	operate	operate	operate	operate
TOG04 to TOG01 TOG14 to TOG11 TOC0	N.A.	HOLD	HOLD	HOLD	HOLD	operate
SO02, SO01,SO00	N.A.	HOLD	HOLD	HOLD	HOLD	operate
SI02, SI01,SI00	N.A.	---	---	---	---	operate
$\overline{SCK02}$, $\overline{SCK01}$,$\overline{SCK00}$	N.A.	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	operate
RXD51, RXD50	N.A.	---	---	---	---	operate
TXD51, TXD50	N.A.	HOLD	HOLD	HOLD	HOLD	operate
FCRXD4-1 **Note 3**	N.A.	---	---	---	---	operate
FCTXD4-1 **Note 3**	N.A.	HOLD Note 2	HOLD Note 2	HOLD Note 2	HOLD	operate
ANI11 to ANI0	---	---	---	---	---	operate
P1,P2,P3,P4,P5,P6,P9	Hi-Z	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	operate
PAH[7:0], PCS[4,3,0], PCT[1:0], PCT[4],PCM[0]	N.A.	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	Hi-Z/--- **Note 1**	operate

Remarks: 1. N.A.: This configuration is not available.

2. ---: Input data is not sampled.

Notes: 1. during output / during input

2. Output values must be set to recessive level by software before activating standby mode. Otherwise CAN bus might be continuously blocked by dominant level.

3. FCTXD4 to FCTXD3) / FCRXD4 to FCRXD3) only for µPD703129.

9.4.4 HALT mode

In this mode, the CPU clock is stopped, though the clock generators (oscillator, SSCG and PLL synthesizer) continue to operate for supplying clock signals to other peripheral function circuits.
Setting the HALT mode when the CPU is idle reduces the total system power consumption.
In the HALT mode, program execution is stopped but the contents of all registers and internal RAM prior are retained as is.
On-chip peripheral hardware irrelevant to the CPU instruction execution also continues to operate. The state of the various hardware units in the HALT mode is tabulated below.

Table 9-4: Operating states in HALT mode

Items	Operation
Clock generator	Operating
SSCG/PLL	Operating
Internal system clock	Operating
WT, WDT clock	Operating
CPU	Stopped (but CPU clock still operates)
I/O line	Unchanged
Peripheral function	Operating
TMC calibration input	Main Clock available
Internal data	Retains all internal data before entering HALT mode, such as CPU registers, status, data, and on-chip RAM.
CLKOUT pin	Clock output (when not inhibited by port setting)
D[15:0], A[23:0], $\overline{\text{RD}}$, $\overline{\text{WR1}}$/ $\overline{\text{WR0}}$, $\overline{\text{CS}}$[0], $\overline{\text{CS}}$[3:4], $\overline{\text{WAIT}}$	Operates

Remark: Even after the HALT instruction is executed, instruction fetch operations continue until the internal instruction pre-fetch queue is full. After the queue becomes full, the CPU stops with the items set as tabulated above.

HALT mode release:

The HALT mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, or $\overline{\text{RESET}}$ signal input.

(1) Release by interrupt request

The HALT mode is released unconditionally by an unmasked maskable interrupt request regardless of its priority level. However, if the HALT mode is entered during execution of an interrupt handler, the operation differs on interrupt priority levels as follows:

(a) If an interrupt request less prioritized than the currently serviced interrupt request is generated, the HALT mode is released but the interrupt is not acknowledged. The interrupt request itself is retained.
(b) If an interrupt request (including a non-maskable one) prioritized than the currently serviced interrupt request is generated, the interrupt request is acknowledged along with the HALT mode release.

Table 9-5: Operation after HALT mode release by interrupt request

Release cause	EI state	DI state
NMI request	Branches to handler address.	
Maskable interrupt request	Branches to handler address, or executes the next instruction.	Executes the next instruction.

Remark: If HALT mode is entered during execution of a particular interrupt handler and an unmasked interrupt request with a higher priority than the previous one is subsequently generated, the program branches to the vector address for the latter interrupt.

(2) Release by $\overline{\text{RESET}}$ pin input

This operation is the same as normal reset operation.

9.4.5 IDLE Mode

In this mode, the CPU clock is stopped resulting in stop of the entire system, though the clock generators (oscillator, SSCG and PLL synthesizer) continue to operate.
As it is not necessary to secure the oscillator oscillation stabilization time and the PLL lock-up time, it is possible to quickly switch to the normal operating mode in response to a release cause. The IDLE mode can be entered by configuration the PSM and PSC registers.
In the IDLE mode, program execution is stopped but the contents of all registers and internal RAM prior to entering this mode are retained. On-chip peripheral hardware operation is also stopped.

The state of the various hardware units in the IDLE mode is tabulated below.

Table 9-6: Operating States in IDLE Mode

Items	Operation
Clock generator	Operating
SSCG/PLL	Operating
Internal system clock	Stopped
WT, WDT clock	Operating
CPU	Stopped
I/O line	Unchanged
Peripheral function	Stops exclude Watch timer / Watchdog timer
TMC calibration input	Main Clock available
Internal data	Retains all internal data before entering IDLE mode, such as CPU registers, status, data, and on-chip RAM.
D[15:0], A[23:0]	Hi-Z
$\overline{RD}$, $\overline{WR1}$/$\overline{WR0}$, $\overline{CS}$[0], $\overline{CS}$[4:2]	H
CLKOUT	L
$\overline{WAIT}$	Input value is not sampled

IDLE mode release:

Release operation is same as release from HALT mode.
The IDLE mode is released by NMI, $\overline{RESET}$ signal input, or an unmasked maskable interrupt request.

(a) Release by Interrupt input:

When the IDLE mode is released, the NMI request is acknowledged.
If the IDLE mode is entered during the execution of NMI handler, the IDLE mode is released but the interrupt is not acknowledged. The interrupt itself is retained.

(b) Release by $\overline{RESET}$ input:

This operation is the same as normal reset operation.

9.4.6 WATCH mode

In this mode f_{CPU} clock is stopped while the oscillator continue to operate to achieve low power, though only oscillator & Watch timer / Watchdog timer continue to operate.
This mode compensates the HALT modes concerning the oscillator stabilization time and power consumption.
As it is not necessary to secure the oscillation stabilization time, it is possible immediately to switch to the normal operating mode in response to a release cause.
This mode is entered by configuration the PSM and PSC registers.
In the WATCH mode, program execution is stopped but the contents of all registers and internal RAM prior to entering this mode are retained. On-chip other peripheral hardware operation is also stopped.

The state of the various hardware units in the WATCH mode is tabulated below.

Table 9-7: Operating States in WATCH Mode

Items	Operation
Clock generator	Operating
SSCG/PLL	Stopped
Internal system clock	Stopped
WT, WDT clock	Operating
CPU	Stopped
I/O line	Unchanged
Peripheral function	Stops exclude Watch timer / Watchdog timer
TMC calibration input	Main Clock available
Internal data	Retains all internal data before entering WATCH mode, such as CPU registers, status, data, and on-chip RAM.
D[15:0], A[23:0]	Hi-Z
$\overline{RD}$, $\overline{WR1}$/$\overline{WR0}$, $\overline{CS}$[0], $\overline{CS}$[4:2]	H
CLKOUT	L
$\overline{WAIT}$	Input value is not sampled

Watch mode release:

The WATCH mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, or $\overline{RESET}$ signal input.

(1) Release by interrupt request:

The WATCH mode is released unconditionally by an unmasked maskable interrupt request regardless of its priority level. After oscillator stabilization time has passed, CPU starts operation. However, if the WATCH mode is entered during execution of an interrupt handler, the operation differs on interrupt priority levels as follows:

(a) If an interrupt request less priorities than the currently serviced interrupt request is generated, the WATCH mode is release but the interrupt is not acknowledged. The interrupt request itself is retained.

(b) If an interrupt request (including a non-maskable one) priorities than the currently serviced interrupt request is generated, the interrupt request is acknowledged along with the WATCH mode release.

Table 9-8: Operation after WATCH mode release by interrupt request

Release cause	EI state	DI state
NMI request	Branches to handler address.	
Maskable interrupt request	Branches to handler address, or executes the next instruction.	Executes the next instruction.

Remark: If WATCH mode is entered during execution of a particular interrupt handler and an unmasked interrupt request with a higher priority than the previous one is subsequently generated, the program branches to the vector address for the later interrupt.

(2) When released by $\overline{\text{RESET}}$ input

This operation is the same as normal reset operation except oscillation stabilization time is not required.

(3) When released by Watchdog Timer $\overline{\text{RESET}}$ input

After oscillator stabilization time has passed, CPU starts operation.

Remark: Before entering the WATCH mode the SSCG and PLL are switched off by hardware. After the WATCH mode has been released the SSCG and PLL can be switched on by software once.
However, the start-up of the SSCG and PLL cause always a certain delay of some Milliseconds. During this time, the clock operates, but the CPU operation is suspended due to clock security reasons.
If it is required to have a fast response when waking up from WATCH mode, the SSCG and PLL should not be re-enabled immediately after waking up, as this causes again the delay. In this case, time-relevant reactions of the CPU should be done first, before re-enabling the SSCG and PLL.

9.4.7 SUB WATCH mode

In this mode f_{CPU} clock and the main oscillator are stopped while the sub oscillator continue to operate to achieve low power, though only oscillator & Watch timer / Watchdog timer continue to operate.
This mode compensates the HALT modes concerning the oscillator stabilization time and power consumption.
At release, for the main oscillator an additional oscillator stabilization time is required.
This mode compensates the WATCH modes concerning the power consumption of the main oscillator.

This mode is entered by configuration the PSM and PSC registers.
In the SUB WATCH mode, program execution is stopped but the contents of all registers and internal RAM prior to entering this mode are retained. On-chip other peripheral hardware operation is also stopped.

The state of the various hardware units in the WATCH mode is tabulated below.

Table 9-9: Operating States in WATCH Mode

Items	Operation
Clock generator	Operating
SSCG/PLL	Stopped
Internal system clock	Stopped
WT, WDT clock	Operating
CPU	Stopped
I/O line	Unchanged
Peripheral function	Stops exclude Watch timer / Watchdog timer
TMC calibration input	Main Clock not available
Internal data	Retains all internal data before entering WATCH mode, such as CPU registers, status, data, and on-chip RAM.
D[15:0], A[23:0]	Hi-Z
$\overline{RD}$, $\overline{WR1}$/$\overline{WR0}$, $\overline{CS}$[0], $\overline{CS}$[4:2]	H
CLKOUT	L
$\overline{WAIT}$	Input value is not sampled

Sub WATCH mode release:

The SUB WATCH mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, or $\overline{RESET}$ signal input.

(1) Release by interrupt request:

The SUB WATCH mode is released unconditionally by an unmasked maskable interrupt request regardless of its priority level. After the main oscillator stabilization time has passed, CPU starts operation. However, if the SUB WATCH mode is entered during execution of an interrupt handler, the operation differs on interrupt priority levels as follows:

(a) If an interrupt request less priorities than the currently serviced interrupt request is generated, the SUB WATCH mode is release but the interrupt is not acknowledged. The interrupt request itself is retained.

(b) (If an interrupt request (including a non-maskable one) priorities than the currently serviced interrupt request is generated, the interrupt request is acknowledged along with the SUB WATCH mode release.

Table 9-10: Operation after SUB WATCH mode release by interrupt request

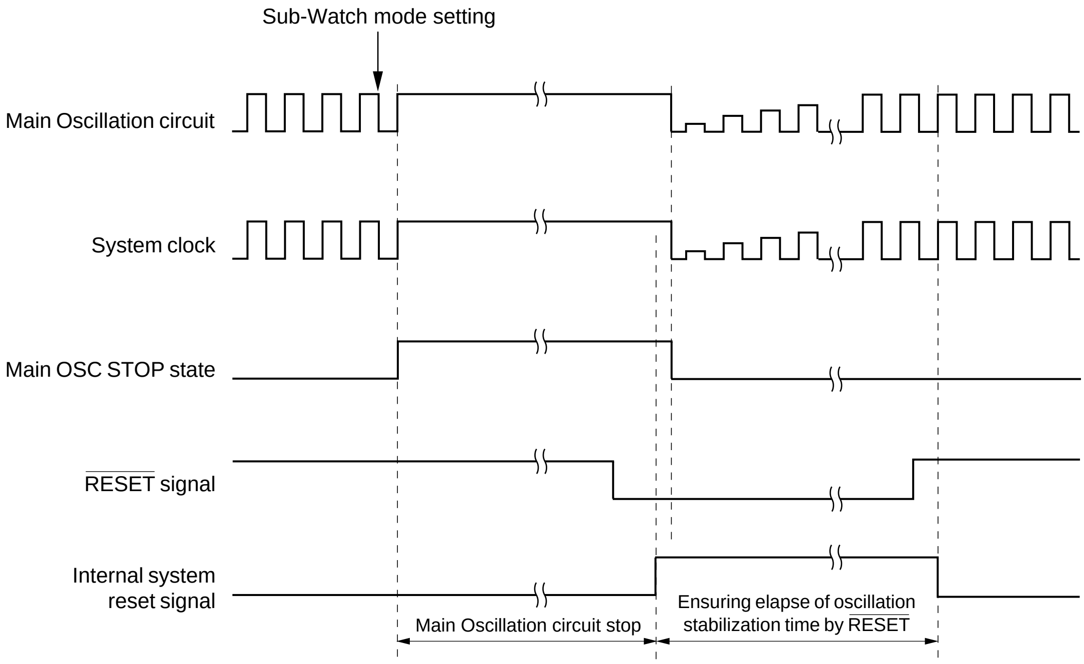
Release cause	EI state	DI state
NMI request	Branches to handler address.	
Maskable interrupt request	Branches to handler address, or executes the next instruction.	Executes the next instruction.

Remark: If SUB WATCH mode is entered during execution of a particular interrupt handler and an unmasked interrupt request with a higher priority than the previous one is subsequently generated, the program branches to the vector address for the later interrupt.

(2) When released by $\overline{\text{RESET}}$ input

This operation is the same as normal reset operation. The Oscillator stabilization time must be ensured by reset input.

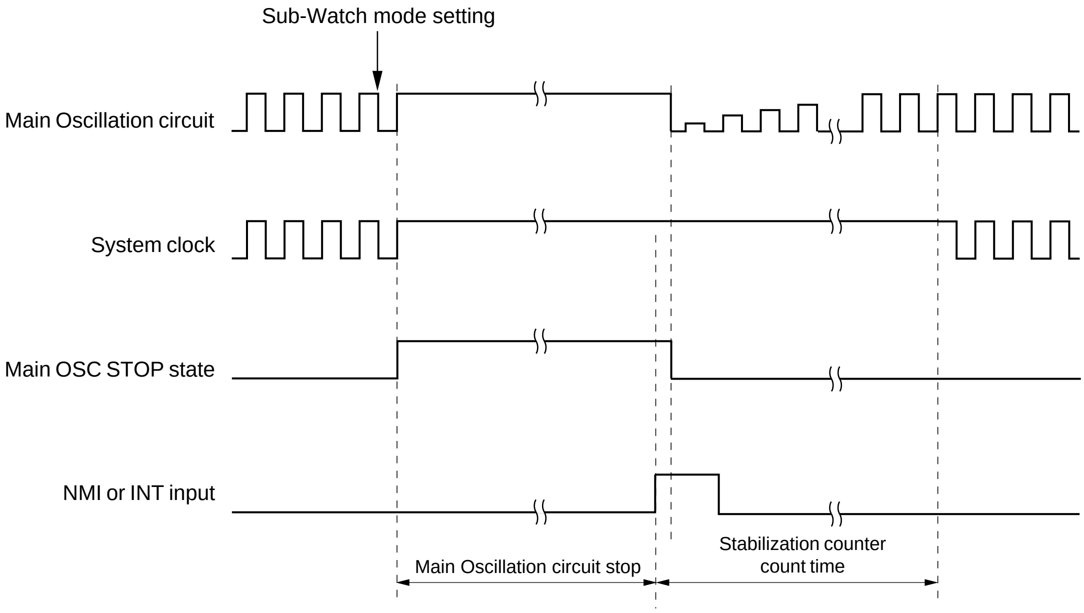
Figure 9-11: Sub Watch mode released by $\overline{\text{RESET}}$ input



(3) When released by Watchdog Timer $\overline{\text{RESET}}$ input

CPU operation starts after main oscillation stabilization time has been secured.

Figure 9-12: Sub Watch mode release by Watchdog reset, NMI, INT



After oscillator stabilization time has passed, CPU starts operation.

Remark: Before entering the SUB WATCH mode the SSCG and the PLL are switched off by hardware. After the SUB WATCH mode has been released the PLL can be switched on by software again once. However, the start-up of the PLL causes always a certain delay of some Milliseconds. During this time, the clock operates, but the CPU operation is suspended due to clock security reasons.
If it is required to have a fast response when waking up from SUB WATCH mode, the PLL should not be re-enabled after waking up, as this causes again the delay. In this case, time-relevant reactions of the CPU should be done first, before re-enabling the PLL.

9.4.8 Software STOP mode

In this mode, the CPU clock is stopped including the clock generators (oscillator, SSCG and PLL synthesizer), resulting in stop of the entire system for ultra-low power consumption (the only consumed is device leakage current).
However, if SOSTP bit = “1” the Sub oscillator and Watchdog timer keeps operating increasing STOP mode current consumption.
When this mode is released, the oscillation stabilization time for the oscillator should be secured until the system clock is stabilized. However, when the external clock operates this product, securing the oscillation stabilization time for the oscillator until the system clock is stabilized is unnecessary. In the direct mode as well, the lock-up time does not have to be secured.
This mode is entered by setting the PSM & PSC register.
In this mode, the program execution stops, but the contents of all registers and internal RAM prior to entering this mode are retained. V850E/CA2 peripherals operations are also stopped
(except Sub oscillator and Watchdog timer in case of SOSTP bit = “1”).

The state of the various hardware units in the software STOP mode is tabulated below.

Table 9-11: Operating States in STOP Mode

Items	Operation
Clock generator	Stopped (Sub OSC operates, if SOSTP bit = “1”)
SSCG/PLL	Stopped
Internal system clock	Stopped
WT clock	Stopped
WDT clock	Stopped, if SOSTP bit = “0”
CPU	Stopped
I/O line**Note**	Unchanged
Peripheral function	Stopped
Internal data**Note**	Retains all previous internal data, such as CPU registers, status, data, and on-chip RAM.
D[15:0], A[23:0]	Hi-Z
$\overline{RD}$, $\overline{WR1}$/$\overline{WR0}$, $\overline{CS0}$, $\overline{CS}$[4:2]	H
CLKOUT	L
$\overline{WAIT}$	Input value is not sampled

Note: When the V_{DD} value is within the operating range. However, even if V_{DD} falls below the lowest operating voltage, the internal RAM content is retained as long as the data retention voltage V_{DDDR} is maintained.

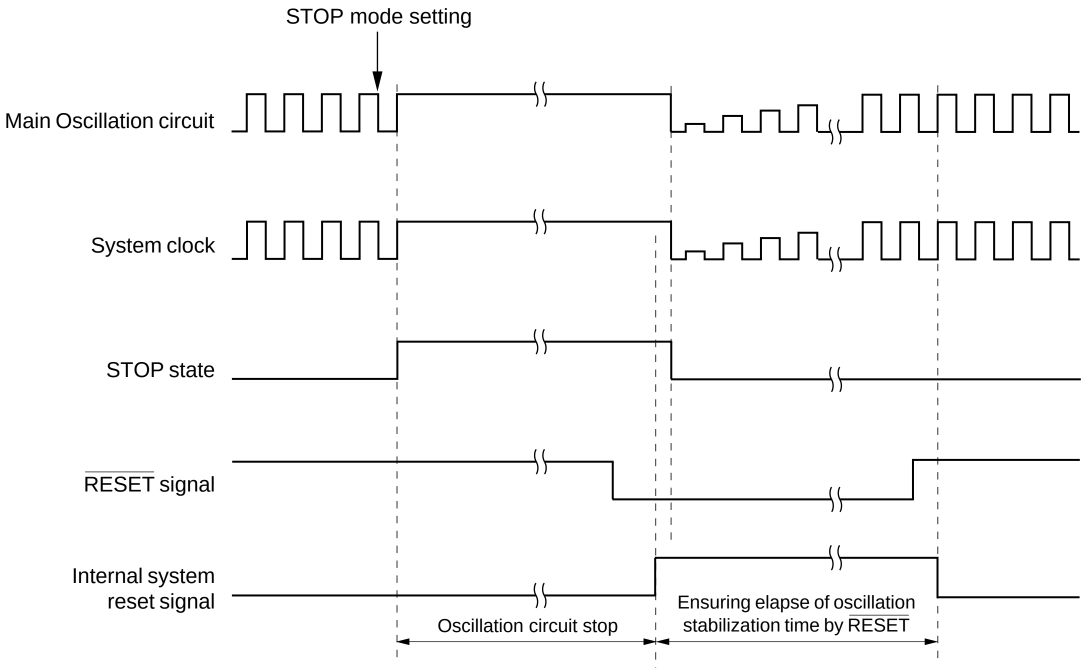
STOP mode release:

The STOP mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, or $\overline{RESET}$ signal input.

(1) When released by $\overline{\text{RESET}}$ input

This operation is the same as normal reset operation. Oscillator stabilization time must be ensured by reset input.

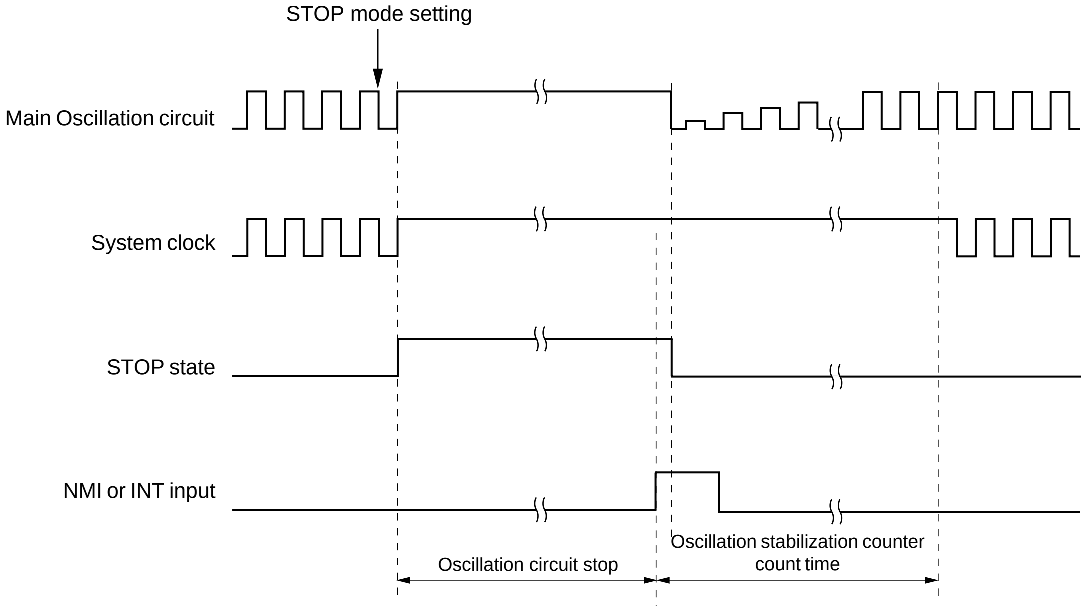
Figure 9-13: STOP mode released by $\overline{\text{RESET}}$ input



(2) When released by Watchdog Timer $\overline{\text{RESET}}$ input

CPU operation starts after main oscillation stabilization time has been secured

Figure 9-14: STOP mode release by Watchdog reset, NMI, INT



After oscillation stabilization time has passed, CPU starts operation.
Before entering the STOP mode the SSCG and PLL are switched off by hardware. After the STOP mode has been released the SSCG and PLL can be switched on any software again once.
However, the start-up of the SSCG and PLL cause always a certain delay of some Milliseconds. During this time, the clock operates, but the CPU operation is suspended due to clock security reasons.
If it is required to have a fast response when waking up from STOP mode, the SSCG and PLL should not be re-enabled after waking up, as this causes again the delay. In this case, time-relevant reactions of the CPU should be done first, before re-enabling the PLL.

9.5 Register Description

9.5.1 Power Save Control Register (PSC)

This is an 8-bit register that controls the power save mode. Data can be written only in a sequence of specific instructions so that its contents are not easily rewritten in case of program hang-up.

This register can be read or written in 8-bit or 1-bit units.

Figure 9-15: Power Save Control Register (PSC)

	7	6	5	4	3	2	1	0	Address	Initial value
PSC	0[Note]	NMI1M	NMI0M	INTM		0	STP		FFFFF1FEH	00H

Bit name	Function
NMI1M	Intsignal release mask 0: Permits NMI1 requests 1: Prohibits NMI0 requests
NMI0M	Intsignal release mask 0: Permits NMI0 requests 1: Prohibits NMI0 requests
INTM	Intsignal release mask 0: Release by maskable interrupt 1: Don’t release by maskable interrupt
STP	Power save mode specification 0: IDLE, WATCH, STOP mode are released 1: IDLE, WATCH, STOP mode are entered

Note: If this bit is set to 1, proper operation can not be guaranteed!

Data is set in the power save control register (PSC) according to the following sequence.

<1> Set the power save mode register (PSM) (with the following instructions).

- Store instruction (ST/SST instruction)
- Bit manipulation instruction (SET1/CLR1/NOT1 instruction)

<2> Prepare data in any one of the general-purpose registers to set to the specific register.
<3> Write arbitrary data to the command register (PRCMD).
<4> Set the power save control register (PSC) (with the following instructions).

- Store instruction (ST/SST instruction)
- Bit manipulation instruction (SET1/CLR1/NOT1 instruction)

<5> Assert the NOP instructions (5 instructions (<5> to <9>).

Sample coding

```
<1> ST.B   r11, PSM [r0]     ; Set PSM register
<2> MOV    0x04, r10
<3> ST.B   r10, PRCMD [r0]   ; Write PRCMD register
<4> ST.B   r10, PSC [r0]     ; Set PSC register
<5> NOP                      ; Dummy instruction
<6> NOP                      ; Dummy instruction
<7> NOP                      ; Dummy instruction
<8> NOP                      ; Dummy instruction
<9> NOP                      ; Dummy instruction
(next instruction)           ; Execution routine after software STOP mode and IDLE mode release
```

No special sequence is required to read the specific register.

Cautions: 1. A store instruction for the command register does not accept interrupts. This coding is made on assumption that <3> and <4> above are executed by the program with consecutive store instructions. If another instruction is set between <3> and <4>, the above sequence may become ineffective when the interrupt is accepted by that instruction, and a malfunction of the program may result.

2. Although the data written to the PHCMD register is dummy data, use the same register as the general register used in specific register setting <4> for writing to the PHCMD register (<3>). The same method should be applied when using a general register for addressing.

3. At least 5 NOP instructions must be inserted after executing a store instruction to the PSC register to set software STOP or IDLE mode.

4. Do not perform a write operation to the PRCMD and specific registers using DMA transfer.

Remarks: 1. To write data to the PSC register, use the store instruction (ST/SST) and bit manipulation instruction (SET1/CLR1/NOT1). The contents of this register can be read in the normal sequence.

2. It is recommended to monitor the status of the clock-sources after a power-save mode has been released. If a power-save mode release condition happened after setting the STP bit, but before the system has entered the related power-save mode, the clock source may not be changed already to the main-oscillator. In this case PLL/SSCG still remains operating.

9.5.2 Power Save Mode Register (PSM)

This is an 8-bit register that control the power save mode and sub-oscillator control.

This register can be read or written in 8-bit or 1-bit units.

Figure 9-16: Power Save Mode Register (PSM)

	7	6	5	4	3	2	1	0	Address	Initial value
PSM	0	CMODE	0	0	OSCDIS	0	PSM1	PSM0	FFFFF820H	00H

<table>
<tr><th>Bit name</th><th>Function</th></tr>
<tr><td>CMODE</td><td>Calibration mode control bit
0: Calibration timer clock is f_{PCLK}
1: Calibration timer clock is output from Main-oscillator clock input</td></tr>
<tr><td>OSCDIS</td><td>Main clock oscillator enable control bit
1: Main oscillator remains stopped after sub-Watch mode release. The CPU will start from sub-clock.
0: Main oscillator will be enabled after sub-Watch mode release and used for CPU clock generation after the oscillation stabilization counter expires.
If this bit is cleared after sub-Watch mode release, the main oscillator will start. After the oscillation stabilization time expires, the main oscillator can be used as system clock source by setting the PCC register accordingly.</td></tr>
<tr><td>PSM1, PSM0</td><td>Standby mode specification bits
<table>
<tr><th>PSM1</th><th>PSM0</th><th>Standby Mode</th></tr>
<tr><td>0</td><td>0</td><td>IDLE</td></tr>
<tr><td>0</td><td>1</td><td>STOP</td></tr>
<tr><td>1</td><td>0</td><td>WATCH</td></tr>
<tr><td>1</td><td>1</td><td>Sub-oscillator WATCH mode (Main oscillator shut-down). This mode can only be enabled if SUBEN is “1”. Otherwise normal WATCH mode is forced.</td></tr>
</table></td></tr>
</table>

Chapter 10 Timer

10.1 Timer C

10.1.1 Features (Timer C)

One channel of Timer C is implemented. Timer C (TMC0) is a 16-bit timer/counter that can perform the following operations.

- 2 capture/compare register
- Programmable pulse generator function
- Interval timer function
- PWM output
- External signal cycle measurement
- sub oscillator calibration function

Remark: In this Timer C chapter following indexes were consequently used

- n = 0, 1 (for each of the 2 Timer C Capture/Compare-Channels)

10.1.2 Function overview (Timer C)

- 16-bit timer/counter (TMC0): 1 channel
- Capture/compare registers: 2
- Count clock division selectable by prescaler (maximum frequency of count clock: 8 MHz)
- Prescaler divide ratio from $f_{PCLK}/2$ to $f_{PCLK}/256$
- Interrupt request sources
  - Capture/compare match interrupt requests: 2 sources

    In case of capture register:
    - INTCCC00 generated by TIC00
    - INTCCC01 generated by TIC01 input

    In case of compare register:
    - INTCCC00 generated by CCC00 match signal
    - INTCCC01 generated by CCC01 match signal
- Overflow interrupt request: 1 source

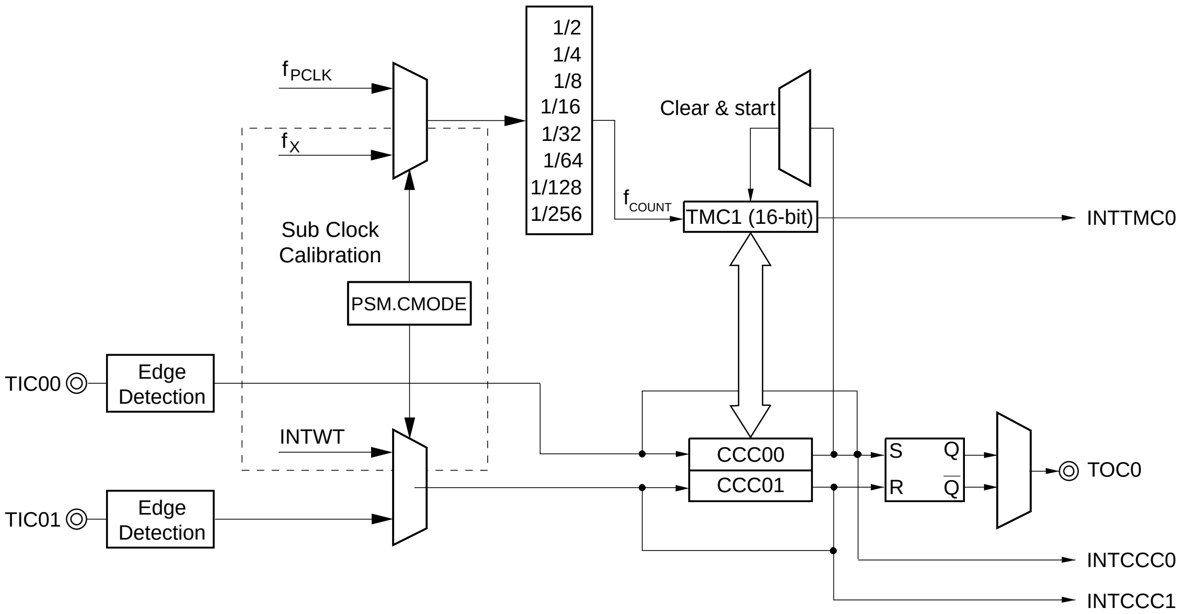
  INTTMC0 generated upon overflow of TMC0 register
- Timer/counter count clock sources: 1 type

  (internal peripheral clock cycle)
- One of two operation modes when the timer/counter overflows can be selected: free-running mode or overflow-stop mode
- The timer/counter can be cleared by match of timer/counter and compare register
- External pulse output (TOC0): 1
- With the sub oscillator calibration function the actual sub clock frequency can be measured taking the main oscillator frequency as reference. The calculated watch time can be corrected according to the actual sub clock deviation.

Remark: n = 0, 1

Figure 10-1, “Block Diagram of Timer C,” on page 273 shows the block diagram of Timer C.

Figure 10-1: Block Diagram of Timer C



Remark: f_{PCLK}: internal peripheral clock

10.1.3 Basic configuration

Table 10-1: Timer C Configuration List

Timer	Count Clock	Register	Read/ Write	Generated Interrupt Signal	Capture Trigger	Timer Output S/R
Timer C	$f_{PCLK}/2$, $f_{PCLK}/4$, $f_{PCLK}/8$,$f_{PCLK}/16$ $f_{PCLK}/32$, $f_{PCLK}/64$, $f_{PCLK}/128$, $f_{PCLK}/256$	TMC0	Read	INTTMC0	-	-
		CCC00	Read/ write	INTCCC00	INTCCC00	TOC0 (S)
		CCC01	Read/ write	INTCCC01	INTCCC01	TOC0 (R)

Remarks: 1. f_{PCLK}: Internal peripheral clock

2. S/R: Set/Reset

3. n = 0, 1

(1) 16-bit counter (TMC0)

TMC0 functions as a 16-bit free-running timer or as an event counter for an external signal. Besides being mainly used for cycle measurement, Timer C can be used as pulse output.

TMC0 is a 16-bit units read-only register.

Figure 10-2: Timer C counter (TMC0)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMC0																	FFF F600H	0000H

Remarks: 1. The TMC0 register can only be read. If writing is performed to the TMC0 register, the subsequent operation is undefined.

2. If the CAE-bit of the TMCC00 register is cleared to "0", a reset is performed asynchronously.

TMC0 performs the count-up operations of an internal count clock. Timer starting and stopping are controlled by the CE bit of Timer C control register 0 (TMCC00).

Selection of the internal count clock

TMC0 operates as a free-running timer.
TMC0 is counted up for each input clock cycle specified by the CS2 to CS0 bits of the TMCC00 register.
A division by the prescaler can be selected for the count clock from among
$f_{PCLK}/2$, $f_{PCLK}/4$, $f_{PCLK}/8$, $f_{PCLK}/16$, $f_{PCLK}/32$, $f_{PCLK}/64$, $f_{PCLK}/128$ and $f_{PCLK}/256$ by the TMCC00 register.

Remark: f_{PCLK}: internal peripheral clock.

An overflow interrupt can be generated if the timer overflows.

Caution: The count clock cannot be changed while the timer is operating.

The conditions when the TMC0 register becomes 0000H are:

(a) Asynchronous reset

- CAE bit of TMCC00 register = 0
- $\overline{\text{RESET}}$ input

(b) Synchronous reset

- CE bit of TMCC00 register = 0
- The CCC00 register is used as a compare register, and the TMC0 and CCC00 registers match when "clearing the TMC0 register" is enabled (CCLR bit of the TMCC01 register = 1).

(2) Capture/compare registers (CCC00 and CCC01)

These capture/compare registers are 16-bit registers.
They can be used as capture registers or compare registers according to the CMS1 bit and CMS0 bit specifications of Timer C control register 1 (TMCC01).

These registers can be read/written in 16-bit units (However, write operations can only be performed in compare mode).

Figure 10-3: Capture/Compare Register 0 (CCC00)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CCC00																	FFFF F602H	0000H

Figure 10-4: Capture/Compare Register 1 (CCC01)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CCC01																	FFFF F604H	0000H

(a) Setting CCC0n registers to capture registers (set CMS1, CMS0 bits of TMCC01 to 0)

When these registers are set to capture registers, the valid edges of the corresponding external interrupt signals TICn0 (n = 0, 1) are detected as capture triggers. The counter register TMC0 is synchronized with the capture trigger, and the value of TMC0 is latched in the CCC00 and CCC01 registers (capture operation).

The valid edge of the TIC00 pin is specified (rising, falling, or both edges) according to the IES10 and IES00 bits of the SESC0 register.
The valid edge of the TIC01 pin is specified according to the IES11 and IES10 bits of the SESC0 register.

The capture operation is performed asynchronously relative to the count clock. The latched value is held in the capture register until the next time the capture operation is performed.
When the CAE bit of Timer C control register 0 (TMCC00) is "0", 0000H is read.

If the CCC00 or CCC01 register is specified as capture register, an interrupt is generated (INTCCC00 or INTCCC01) by detecting the valid edge of signals.

Caution: If the capture operation and the TMC0 register count prohibit setting (CE bit of TMCC00 register = 0) timings conflict, the captured data becomes undefined, and no INTCCC00 interrupt is generated (n = 0, 1).

(b) Setting CCC0n registers to compare registers (CMS1 and CMS0 of TMCC01 = 1)

When these registers are set to compare registers, the TMC0 and register values are compared for each timer count clock, and an interrupt is generated by a match.
If the CCLR bit of Timer C control register 1 (TMCC01) is set (1), the TMC0 value is cleared (0) at the same time as a match with the CCC00 register (it is not cleared (0) by a match with the CCC01 register).

A compare register is equipped with a set/reset output function. The corresponding timer output (TOC0) is set or reset, synchronized with the generation of a match signal.

The interrupt selection source differs according to the function of the selected register.

Cautions:

1. **The minimum value for CCC0 to achieve a symmetrical output wave with the "Compare Clear Enable" function (CLR bit = 1) is 0003H.**
2. **To write to capture/compare registers 0 and 1 (CCC00, CCC01), always set the CAE bit to 1 first. When the CAE bit is 0, even if writing to registers CCC00 and CCC01, the data that is written will be invalid because the reset is asynchronous.**
3. **Perform a write operation to capture/compare registers 0 and 1 after setting them to compare registers according to the TMCC01 register setting. If they are set to capture registers (CMS1 and CMS0 bits of TMCC01 register = 0), no data is written even if a write operation is performed to CCC00 and CCC01.**
4. **When these registers are set to compare registers, the INTCCC00 or INTCCC01 interrupt can not be used for generating interrupts for external inputs edges.**

10.1.4 Control registers

(1) Timer C control register 0 (TMCC00)

The TMCC00 register controls the operation of TMC0.

This register can be read/written in 8-bit or 1-bit units.

Caution: The CAE bit and CE bit cannot be set at the same time. Be sure to set the CAE bit prior to setting the CE bit. To use an external pin related to the timer function when using Timer C, be sure to set the CAE bit to "1" after setting the external pin to the control mode.

Figure 10-5: Timer C control Register 0 (TMCC00) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
TMCC00	OVF	CS2	CS1	CS0	0	0	CE	CAE	FFFF F606H	00H

Bit Position	Bit Name	Function
7	OVF	Flag that indicates TMC0 overflow. 0: No overflow 1: Overflow The OVF bit becomes “1” when TMC0 changes from FFFFH to 0000H. An overflow interrupt request (INTTMC0) is generated at the same time. However, if CCC00 is set to the compare mode (CMS0 bit of the TMCC01 register = 1) and match clear during comparison of TMC0 and CCC00 is enabled (CCLR bit of TMCC01 register = 1), and TMC0 is cleared to 0000H following match at FFFFH, TMC0 is considered to have been cleared and the OVF bit does not become "1", nor is the INTTMC0 interrupt generated. The OVF bit holds a “1” until “0” is written to it or an asynchronous reset is applied while the CAE bit = 0. Interrupts by overflow and the OVF bit are independent, and even if the OVF bit is manipulated, this does not affect the interrupt request flag for INTTMC0 (TMCIF0 register). If an overflow occurs while the OVF bit is being read, the value of the flag changes and the value is returned at the next read.

Figure 10-5: Timer C control Register 0 (TMCC00) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>CS2 to CS0</td><td>Selects the internal count clock for TMC0.
<table>
<tr><th>CS2</th><th>CS1</th><th>CS0</th><th>Count Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>$f_{PCLK}/2$</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>$f_{PCLK}/4$</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>$f_{PCLK}/8$</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>$f_{PCLK}/16$</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>$f_{PCLK}/32$</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>$f_{PCLK}/64$</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>$f_{PCLK}/128$</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>$f_{PCLK}/256$</td></tr>
</table>
Caution: Do not change the CS2 to CS0 bits during timer operation. If they are to be changed, they must be changed after setting the CE bit to "0". If the CS2 to CS0 bits are overwritten during timer operation, the operation is not guaranteed.
Remark: f_{PCLK}: internal peripheral-clock</td></tr>
<tr><td>1</td><td>CE</td><td>Controls the operation of TMC0.
0: Disable count (timer stopped at 0000H and does not operate)
1: Perform count operation.
Caution: If CE = 0, the external pulse output (TOC0) becomes inactive level (The active level of TOC0 output is set with the ALV bit of the TMCC01 register).</td></tr>
<tr><td>0</td><td>CAE</td><td>Controls the internal count clock (f_{COUNT}).
0: Asynchronously reset entire TMC0 unit. Stop base clock supply to TMC0 unit.
1: Supply peripheral-clock (f_{PCLK}) to TMC0 unit.
Cautions: 1. When CAE = 0 is set, the TMC0 unit can be reset asynchronously.
2. When CAE = 0, the TMC0 unit is in a reset state. To operate TMC0, first set CAE = 1.
3. When the CAE bit is changed from "1" to "0", all the registers of the TMC0 unit are initialized. When again setting CAE = 1, be sure to then again set all the registers of the TMC0 unit.</td></tr>
</table>

(2) Timer C control register 1 (TMCC01)

The TMCC01 register controls the operation of TMC0.

This register can be read/written in 8-bit or 1-bit units.

Cautions: 1. Do not change the bits of the TMCC01 register during timer operation. If they are to be changed, they must be changed after setting the CE bit of the TMCC00 register to 0. If the TMCC01 register is overwritten during timer operation, the operation is not guaranteed.

2. If the ENTO bit and the ALV bit are changed simultaneously, a glitch (spike-shaped noise) may be generated in the TOC0 pin output. Either design the circuit that will not malfunction even if a glitch is generated, or make sure that the ENTO bit and the ALV bit do not change at the same time.

3. TOC0 output remains unchanged by external interrupt signals (INTCCC00, INTCCC01). When using the TOC0 signal, set the capture/compare register to the compare register (CMS1, CMS0 bits of TMCC01 register = 1).

Remark: A reset takes precedence for the flip-flop of the TOC0 output.

Figure 10-6: Timer C control Register 1 (TMCC01) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
TMCC01	OST	ENTO	ALV	0	CCLR	0	CMS1	CMS0	FFFF F608H	20H

Bit Position	Bit name	Function
7	OST	Setting of the timer operation after overflow: 0: After overflow the count operation is continued (free running mode) 1: After overflow the count operation is stopped (overflow stop mode). The count is restarted by writing "1" to the CE bit.
6	ETO	Enables/disables output of external pulse output (TOC0). 0: Disable external pulse output. Output of inactive level of ALV bit to TOC0 pin is fixed. TOC0 pin level remains unchanged even if match signal from correspond-ing compare register is generated. 1: Enable external pulse output. Compare register match causes TOC0 output to change. However, in capture mode, TOC0 output does not change. An ALV bit inactive level is output from the time when timer output is enabled until a match signal is generated. **Caution: If either CCC00 or CCC01 is specified as a capture register, the ENTO bit must be set to "0".**
5	ALV	Specifies active level of external pulse output (TOC0). 0: Active level is low level. 1: Active level is high level. **Caution: The initial value of the ALV bit is "1".**

Figure 10-6: Timer C control Register 1 (TMCC01) (2/2)

Bit Position	Bit name	Function
3	CLR	Enables/disables TMC0 clearing during compare operation. 0: Disable clearing. 1: Enable clearing (TMC0 is cleared when CCC00 and TMC0 match during compare operation).
1	CMS1	Selects operation mode of capture/compare register (CCC01). 0: Register operates as capture register. 1: Register operates as compare register.
0	CMS0	Selects operation mode of capture/compare register (CCC00). 0: Register operates as capture register. 1: Register operates as compare register.

(3) Valid edge selection register (SESC0)

This register specifies the valid edge of external interrupt requests from an external TICmn pin (n, m = 0 to 1).
The rising edge, the falling edge, or both rising and falling edges can be specified as the valid edge independently for each pin.

This register can be read/written in 8-bit or 1-bit units.

Caution: Do not change the bits of SESC0 register during timer operation. If they have to be changed, they must be changed after setting the CE bit of the TMCC00 register to "0". If the SESC0 register is overwritten during timer operation, the operation is not guaranteed.

Figure 10-7: Valid Edge Selection Register (SESC0)

	7	6	5	4	3	2	1	0	Address	Initial value
SESC0	0	0	0	0	IES11	IES10	IES01	IES00	FFFF F609H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>3, 2</td><td>IES11, IES10</td><td rowspan="2">Specifies the valid edge of TICn0 pins.
<table>
<tr><th>IESn1</th><th>IESn0</th><th>Operation</th></tr>
<tr><td>0</td><td>0</td><td>Falling edge</td></tr>
<tr><td>0</td><td>1</td><td>Rising edge</td></tr>
<tr><td>1</td><td>0</td><td>Setting prohibited</td></tr>
<tr><td>1</td><td>1</td><td>Both rising and falling edges</td></tr>
</table></td></tr>
<tr><td>1, 0</td><td>IES01, IES00</td></tr>
</table>

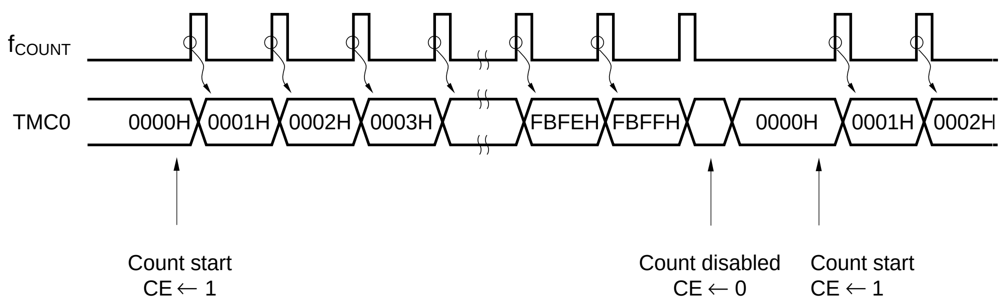
Remark: n = 0, 1

10.1.5 Operation

(1) Count operation

Timer C can function as a 16-bit free-running timer.
When it operates as a free-running timer and the CCC00 register or CCC01 register and the TMC0 count value match, an interrupt signal is generated and the timer output signal (TOC0) can be set or reset.
Also, a capture operation that holds the TMC0 count value in the CCC00 or CCC01 register is performed, synchronized with the valid edge that was detected from the external interrupt request input pin as an external trigger. The capture value is held until the next capture trigger is generated.

Figure 10-8: Timing of basic operation of Timer C



(2) Overflow

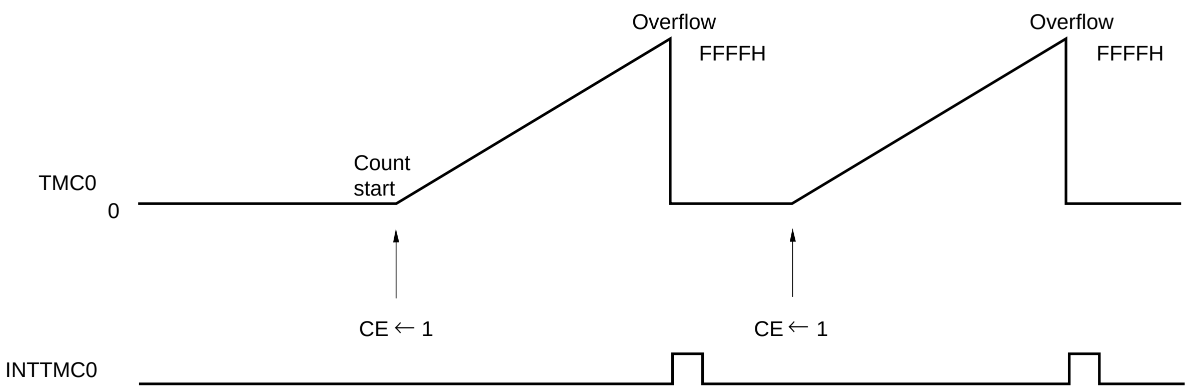
When the TMC0 register has counted the count clock from FFFFH to 0000H, the OVF bit of the TMCC00 register is set to "1", and an overflow interrupt (INTTMC0) is generated at the same time.

However, if the CCC00 register is set to compare mode (CMS0 = 1) and to the value FFFFH, when match clearing is enabled (CCLR = 1) the TMC0 counter register is considered to be cleared and the OVF bit is not set to "1" when the TMC0 counter register changes from FFFFH to 0000H. Also, the overflow interrupt (INTTMC0) is not generated.

When the TMC0 counter register is changed from FFFFH to 0000H because the CE bit changes from "1" to "0", the TMC0 register is considered to be cleared, but the OVF bit is not set to 1 and no INTTMC0 interrupt is generated.

Also, timer operation can be stopped after an overflow by setting the OST bit of the TMCC01 register to 1. When the timer is stopped due to an overflow, the count operation is not restarted until the CE bit of the TMCC00 register is set to "1".
Operation is not affected even if the CE bit is set to "1" during a count operation.

Figure 10-9: Timing of interrupt operation after overflow



(3) Capture operation

The TMC0 register has two capture/compare registers. These are the CCC00 register and the CCC01 register.
A capture operation or a compare operation is performed according to the settings of both the CMS1 and CMS0 bits of the TMCC01 register. If the CMS1 and CMS0 bits of the TMCC01 register are set to "0", the register operates as a capture register.

A capture operation that captures and holds the TMC0 count value asynchronously relative to the count clock (f_{COUNT}) is performed synchronized with an external trigger.

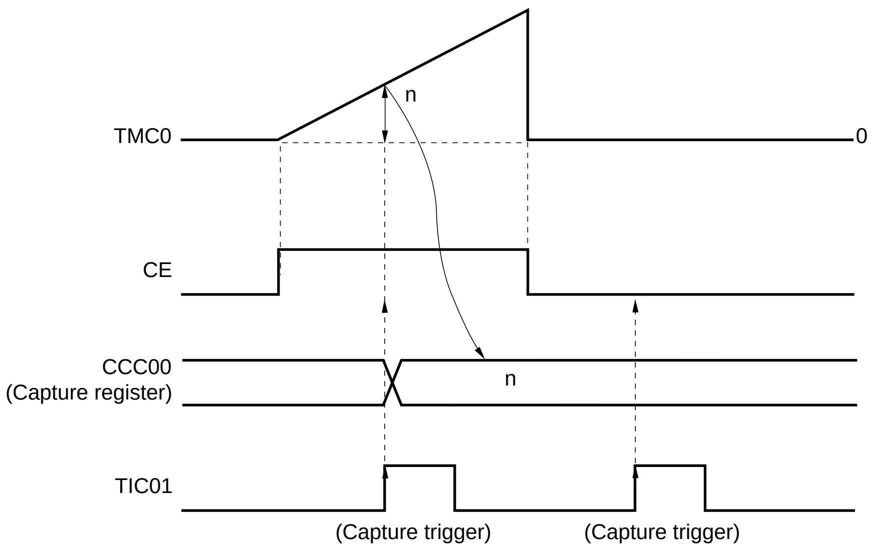
An interrupt request (INTCCC00 or INTCCC01) (n = 0, 1) is generated by TICn0 or TICn1 signal input and is used as an external trigger (capture trigger).
The valid edge of the capture trigger is set by valid edge selection register (SESC0).

The TMC0 count value during counting is captured and held in the capture register, synchronized with that capture trigger signal. The capture register value is held until the next capture trigger is generated.

(a) Example: capture for pulse cycle measurement

If one of the edges is set as the capture trigger, the input **pulse cycle** can be measured.

Figure 10-10: Timing of capture for pulse cycle measurement (rising edge)



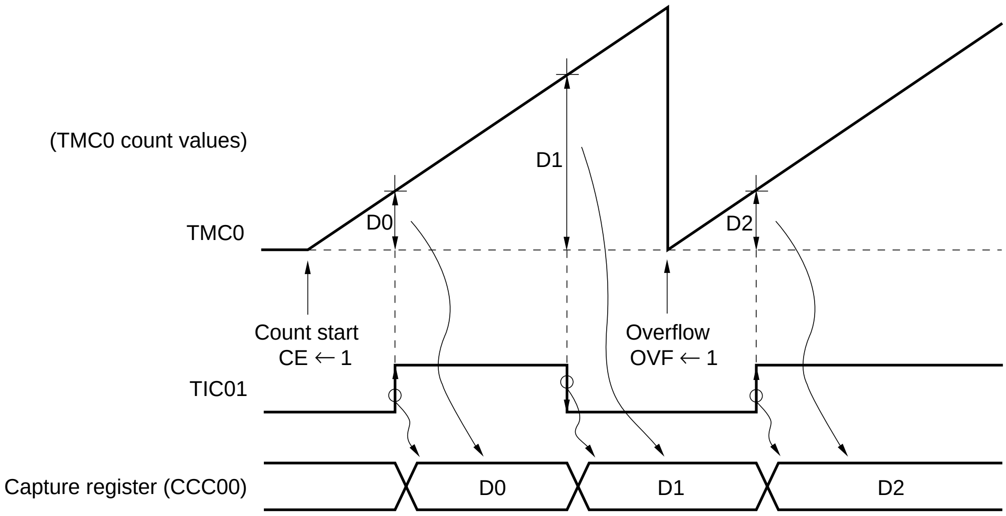
Remarks: 1. When the CE bit is 0, no capture operation is performed even if INTCCC01 is input.

2. Valid edge of TIC01: Rising edge.

(b) Example: capture for pulse cycle measurement

If both the rising and falling edges are set as capture triggers, the input **pulse width** from an external source can be measured.

Figure 10-11: Timing of capture for pulse width measurement (both edges)



Remark: D0 to D2: TMC0 count values

(c) Example: Cycle measurement

By setting the TMCC00 and TMCC01 registers as described below Timer C can measure the cycle of signals input to the TICn0 pin.
The valid edge of the TIC00 pin is selected according to the IES01 and IES00 bits of the SESC0 register.
(Similar the valid edge of the TIC01 pin is selected according to the IES11 and IES10 bits of the SESC0 register.)
Either the rising edge, the falling edge, or both edges can be selected as the valid edges of both pins.

Setting method:

(1) set corresponding port pins (P5) to Timer C input (PM5 to input, PMC5 to Timer C0)
(2) set CAE bit of TMCC00 register to 1 for activating the Timer C peripheral
(3) set the valid edge of the TICn0 pin with the IES01 and IES00 bits of the SESC0 register (here for rising edge: IES01 = 0, IES00 = 1)
(4) set CMS1 and CMS0 bits of TMCC01 register to 0
(5) set CE bit to enable the counter and start operation

Operation:

(1) the valid edge input of the TICn0 pin is set as the trigger for capturing the TMC0 register value in the CCC00 register.
(2) When this value is captured, an INTCCC00 interrupt is generated.

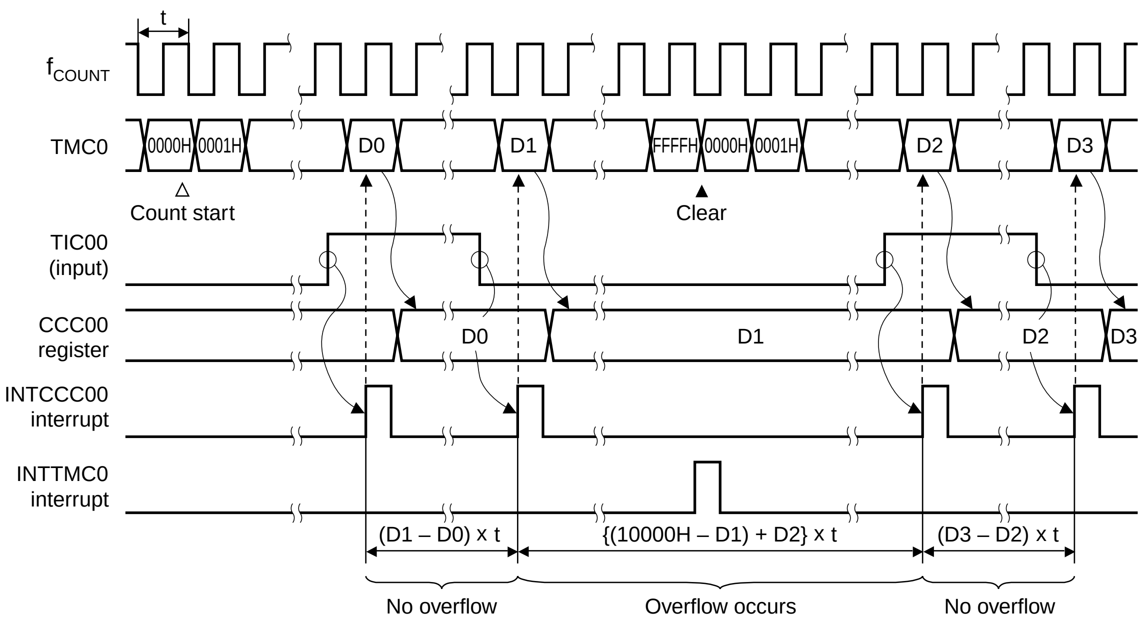
(Similarly, the valid edge input of the TICn0 pin is set as the trigger for capturing the TMC0 register value in the CCC01 register. When this value is captured, an INTCCC01 interrupt is generated.)

Calculation:

The cycle of signals input to the INTCCC00 pin is calculated by obtaining the difference between the TMC0 register's count value (Dx) that was captured in the CCC00 register according to the x-th valid edge input of the TIC00 pin and the TMC0 register's count value (D(x+1)) that was captured in the CCC00 register according to the (x+1)-th valid edge input of the TIC00 pin and multiplying the value of this difference by the cycle of the clock control signal.

(Similarly the cycle of signals input to the INTCCC01 pin is calculated by obtaining the difference between the TMC0 register's count value (Dx) that was captured in the CCC01 register according to the x-th valid edge input of the TIC01 pin and the TMC0 register's count value (D(x+1)) that was captured in the CCC01 register according to the (x+1)-th valid edge input of the TIC01 pin and multiplying the value of this difference by the cycle of the clock control signal.)

Figure 10-12: Timing of cycle measurement operation



Caution: An overflow must not be generated more than once between the 1st and 2nd INTCCC00 interrupts.

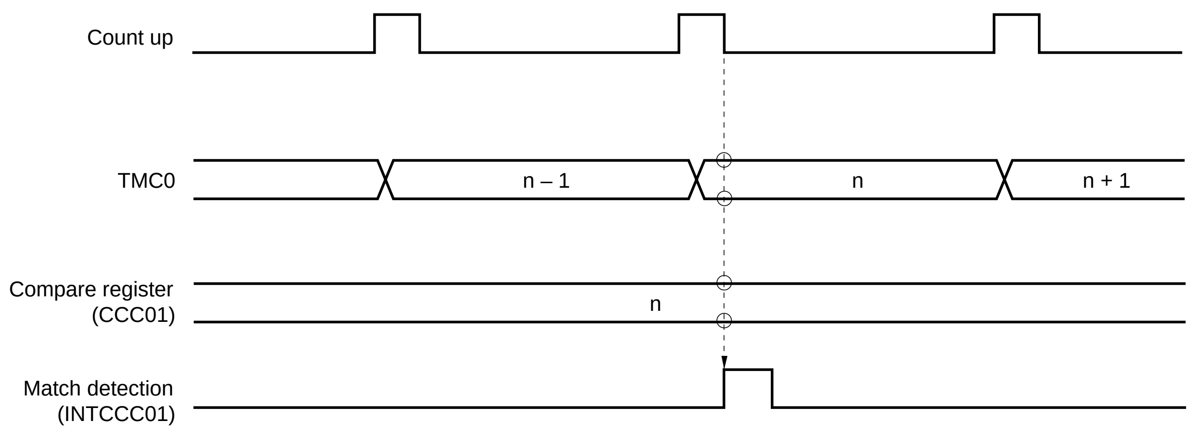
Remarks:
1. D0 to D3: TMC0 register's count values
2. t: Count clock cycle
3. In this example, the valid edge of TIC00 input has been set to both edges (rising and falling).

(4) Compare operation

The TMC0 register has two capture/compare registers. These are the CCC00 register and the CCC01 register.
A capture operation or a compare operation is performed according to the settings of both the CMS1 and CMS0 bits of the TMCC01 register. If "1" is set in the CMS1 and CMS0 bits of the TMCC01 register, the register operates as a compare register.
A compare operation that compares the value that was set in the compare register and the TMC0 count value is performed.

If the TMC0 count value matches the value of the compare register, which had been set in advance, a match signal is sent to the output control circuit. The match signal causes the timer output pin (TOC0) to change and an interrupt request signal (INTCCC00, INTCCC01) to be generated at the same time.

Figure 10-13: Timing of compare operation



Remark: The match is detected immediately after the count up, and the match detection signal is generated.

(a) When CCC00 register is set to 0000H

If the CCC00 register is set to 0000H, the 0000H after the TMC0 register counts up from FFFFH to 0000H is judged as a match. The 0000H when the TMC0 register begins counting is not judged as a match.

(b) When match clearing is enabled

If match clearing is enabled (CLR bit = 1) for the CCC00 register, the TMC0 register is cleared when a match with the TMC0 register occurs during a compare operation.

(c) Example: Interval timer

By setting the TMCC00 and TMCC01 registers as described below Timer C operates as an interval timer that repeatedly generates interrupt requests with the value that was set in advance in the CCC00 register as the interval.

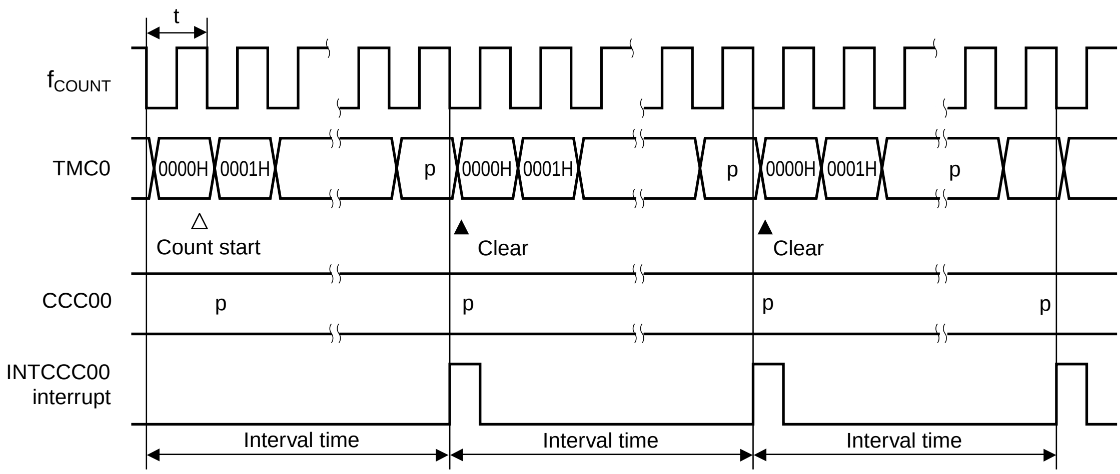
Setting method:

(1) set corresponding port pins (P5) to Timer C input (PM5 to input, PMC5 to Timer C0)
(2) set CAE bit to "1" for activate the Timer C peripheral
(3) set CLR- and CMS0 bit of TMCC01 register to "1"
(4) set CE bit to enable the counter and start operation

Operation:

(1) When the counter value of the TMC0 register matches the setting value of the CCC00 register, the TMC0 register is cleared (0000H)
(2) An interrupt request signal (INTCCC00) is generated at the same time that the count operation resumes.

Figure 10-14: Timing of interval timer operation



Remarks: 1. p:Setting value of CCC00 register (0000H to FFFFH)
2. t:Count clock cycle
3. Interval time = (p + 1) × t

(5) PWM output

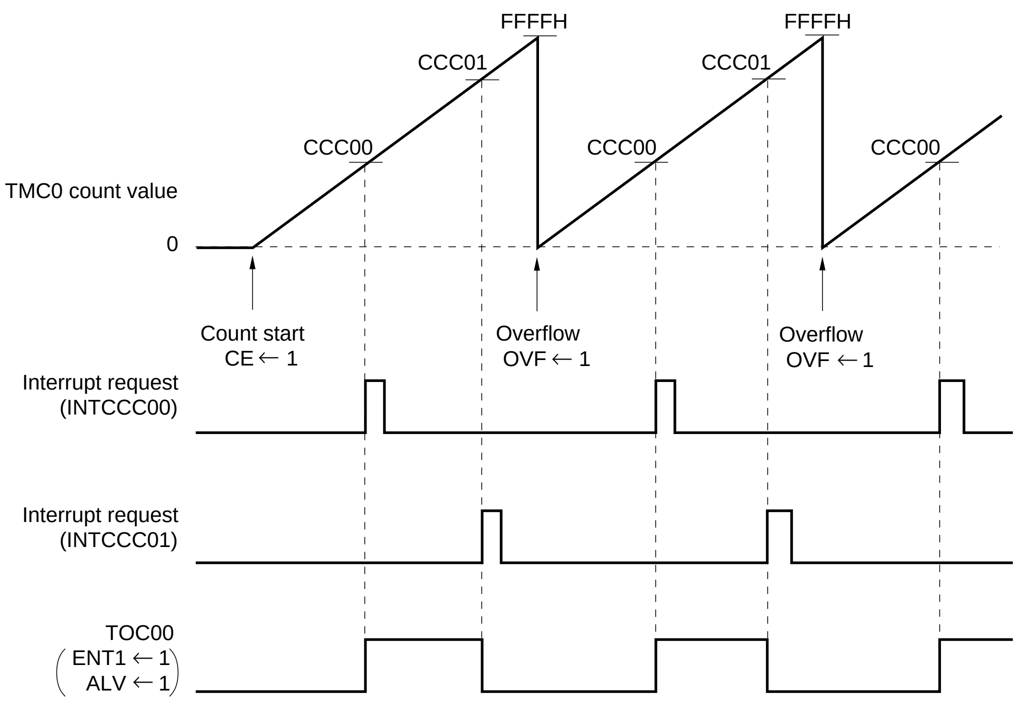
Timer C has one timer output pin (TOC0).
An external pulse output (TOC0) can be generated when a match of the two compare registers (CCC00 and CCC01) and the TMC0 register is detected.

If a match is detected when the TMC0 count value and the CCC00 value are compared, the output level of the TOC0 pin is set.
Also, if a match is detected when the TMC0 count value and the CCC01 value are compared, the output level of the TOC0 pin is reset.
The output level (set, reset) depends on the settings of the ALV and ENTO bits of the TMCC01 register.

Table 10-2: TOC0 Output Control

ENTO	ALV	TOC0 Output	
		External Pulse Output	Output Level
0	0	Disable	High level
0	1	Disable	Low level
1	0	Enable	When the CCC00 register is matched: Low level When the CCC01 register is matched: High level
1	1	Enable	When the CCC00 register is matched: High level When the CCC01 register is matched: Low level

Figure 10-15: Timing of PWM output operation (overview)



(a) Example PWM output

By setting the TMCC00 and TMCC01 registers as described below Timer C can output a PWM of an arbitrary frequency with the values that were set in advance in the CCC00 and CCC01 registers determining the intervals.

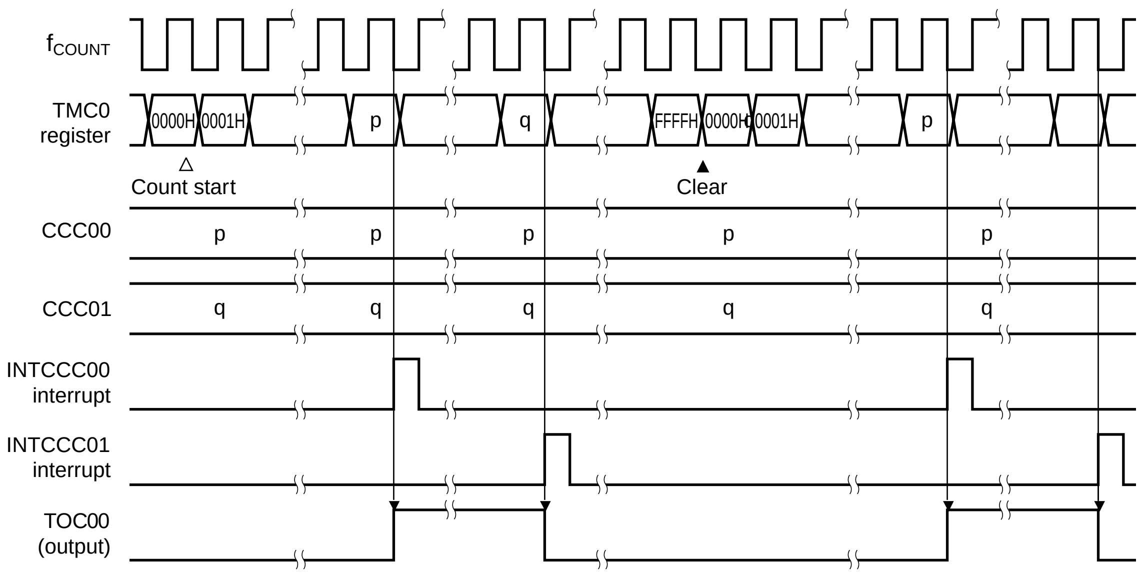
Setting method:

(1) set corresponding port pins (P5) to Timer C output (PM5 to input, PMC5 to Timer C0)
(2) set CAE bit of TMCC00 register to "1" for activating the Timer C peripheral
(3) set the active level of TOC0 output by the ALV bit of the TMCC01 register (here: ALV = 1)
(4) set ENT1, CMS1 and CMS0 bits of TMCC01 register to "1" (leave CLR bit to 0)
(5) set CE bit to "1" to enable the counter and start operation

Operation:

(1) When the counter value of the TMC0 register matches the setting value of the CCC00 register, the TOC0 output becomes active.
(2) When the counter value of the TMC0 register matches the setting value of the CCC01 register, the TOC0 output becomes inactive. This enables a PWM of an arbitrary frequency to be output.

Figure 10-16: Timing of PWM output operation (detail)



Remarks: **1.** p: Setting value of CCC00 register (0000H to FFFFH)

2. q: Setting value of CCC01 register (0000H to FFFFH)

3. p: 1/4 q

4. In this example, the active level of TOC0 output is set to high level.

(b) When CCC00 = CCC01

When the setting value of the CCC00 register and the setting value of the CCC01 register are the same, the TOC0 output remains inactive and does not change.

10.1.6 Sub Oscillator Calibration Function

For automotive dashboard application, customer need to achieve a watch timer accuracy of about 1 sec/week. This target is difficult to manage using a 32 KHz crystal for sub oscillator because of the temperature dependency of these crystal types. The crystal type used for main oscillator has a temperature deviation which compensates themselves over one year (summer/winter) while the temperature deviation of sub oscillator accumulates over one year.

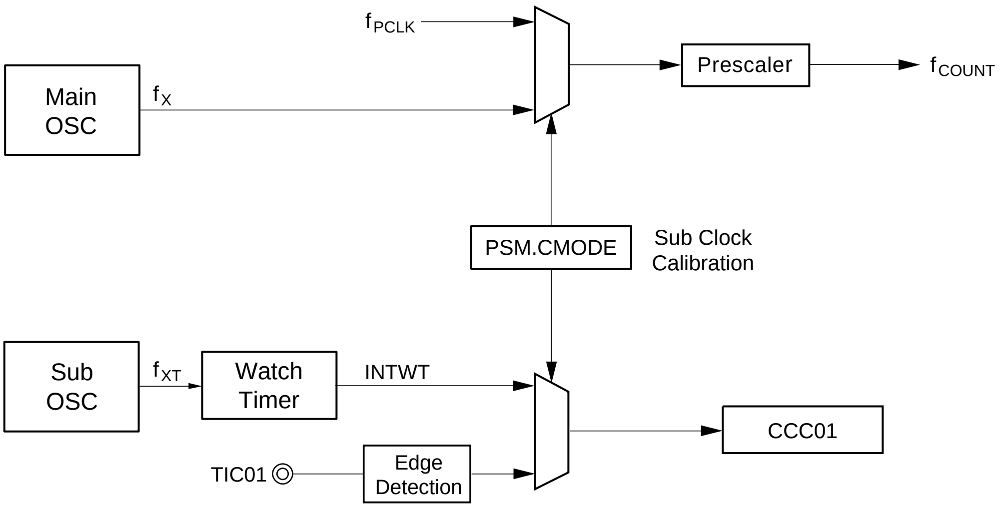
A sub-clock oscillator calibration function is available by measuring the sub-clock deviation with the main clock oscillator. To perform this measurement in a power efficient way, a special clock path for Timer C0 is supported. This clock supply switches between the peripheral macro clock prescaler output and direct clock from the main oscillator input. In the Jupiter device the Timer C0 will be used to measure the sub-clock frequency by capture operation of the Watch timer interrupts. The device can be switched to Watch mode during measurement operation.

To enable usage of the sub watch mode for real watch timer applications, a sub oscillator calibration mode is implemented into Jupiter. In this mode the actual sub clock frequency can be measured taking the main oscillator frequency as reference and the calculated watch time can be corrected according to the actual sub clock deviation. The sub oscillator calibration mode is implemented in the following way:

- Capture input for CCC01 input channel of Timer C can be switched to watch timer interrupt output
- f_{PCLK} input of Timer C can be clocked directly by the main oscillator with f_X.

For the sub oscillator calibration, the Timer C peripheral clock and the CCC01 capture input need to be multiplexed.

Figure 10-17: Multiplexed Inputs for Timer C Sub Oscillator Calibration Function



To use the sub oscillator calibration feature, the watch timer clock must be derived from the sub oscillator.

This is the flow for sub oscillator calibration:

(1) Disable Timer C0

(2) Enable sub oscillation calibration feature in Jupiter clock controller by setting the CMODE bit in the PSM register to "1".

(3) Enable Timer C0 and set CCC01 to capture mode.

(4) On the next watch timer wake up interrupt, the captured value of CCC01 gives the modulo counter for main oscillator clocks per watch timer interrupt. To achieve a higher accuracy measurement, capture value of the n-th watch timer interrupt should be taken as the result.

10.1.7 Precautions Timer C

Various precautions concerning Timer C are shown below.

(1) The following bits and registers must not be rewritten during operation (TMCC00 register CE = 1).

- CS2 to CS0 bits of TMCC00 register
- TMCC01 register
- SESC0 register

(2) The CAE bit of the TMCC00 register is a TMC0 counter reset signal. To use TMC0, first set the CAE bit to 1.

(3) The analog noise elimination time + two cycles of the input clock are required to detect a valid edge of the external input (TIC00 or TIC01). Therefore, edge detection will not be performed normally for changes that are less than the analog noise elimination time + two cycles of the input clock.

(4) The operation of an interrupt output (INTCCC00 or INTCCC01) is automatically determined according to the operating state of the capture/compare registers (CCC00, CCC01). When the capture/compare registers are used for a capture mode, the external trigger (TIC00,TIC01) is used for valid edge detection. When the capture/compare registers are used for a compare mode, the external interrupt output is used for a match interrupt indicating a match with the TMC0 register.

(5) If the ENTO and ALV bits of the TMCC01 register are changed at the same time, a glitch (spike shaped noise) may be generated in the TOC0 pin output. Either create a circuit configuration that will not malfunction even if a glitch is generated or make sure that the ENTO and ALV bits do not change at the same time.

10.2 Timer D

2 x 16-bit interval timer of Timer D are implemented:

- Timer D1
- Timer D2

10.2.1 Features Timer D

Timer Dn (TMD) functions as a 16-bit interval timer.

10.2.2 Function overview Timer Dn

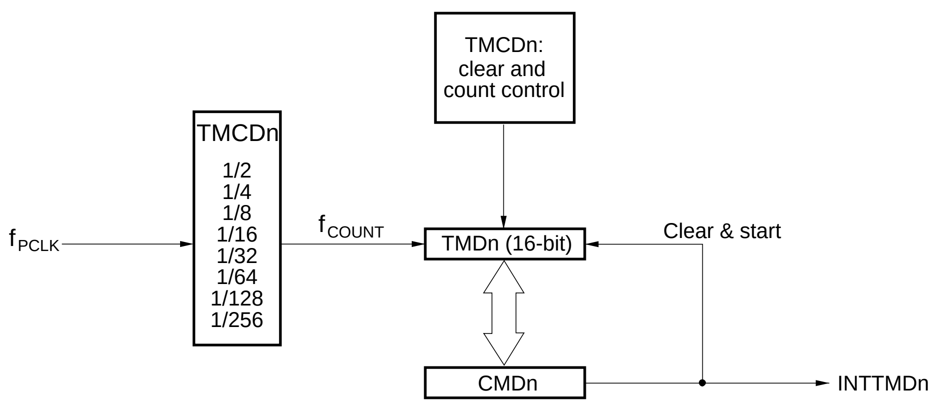
- Compare register: 1
- Count clock selected from divisions of internal peripheral clock
  (maximum frequency of count clock: $f_{PCLK}/2$ (10 MHz @ f_{PCLK} = 20 MHz))
- Prescaler division ratio
  8 division ratios can be selected related to the internal peripheral clock (f_{PCLK}). The range is from $f_{PCLK}/2$ to $f_{PCLK}/256$.
- Interrupt request sources: 1
  - Compare match interrupt
  INTTMDn generated with CMDn match signal
- Timer clear:
  TMDn register can be cleared by CMDn register match.

Remark: In this Timer D chapter following indexes is consequently used

- n = 0, 1 (for each of the 2 Timer D)
- f_{PCLK}: Internal peripheral clock

Figure 10-18 shows the block diagram of the channel of Timer Dn.

Figure 10-18: Block Diagram of Timer Dn (n = 0, 1)



Remark: n = 0, 1

10.2.3 Basic configuration

Table 10-3: Timer Dn Configuration List (n = 0, 1)

Timer	Count Clock	Register	R/W	Generated Interrupt Signal	Capture Trigger	Timer Output S/R	Other Functions
Timer Dn	$f_{PCLK}/2$, $f_{PCLK}/4$, $f_{PCLK}/8$, $f_{PCLK}/16$, $f_{PCLK}/32$, $f_{PCLK}/64$, $f_{PCLK}/128$, $f_{PCLK}/256$	TMDn	R	–	–	–	–
		CMDn	R/W	INTTMDn	–	–	–
		TMCDn	R/W	–	–	–	–

Remarks: 1. f_{PCLK}: Internal peripheral clock

2. S/R: Set/Reset

(1) Timer D counter Register (TMDn) (n = 0, 1)

Timer Dn is a 16-bit timer. It is mainly used as an interval timer for software (n = 0, 1).
Starting and stopping TMDn is controlled by the CE bit of the Timer Dn control register (TMCDn).
A division by the prescaler can be selected for the count clock from among $f_{PCLK}/2$ and $f_{PCLK}/256$ in 8 steps by the CS2 to CS0 bits of the TMCDn register.

TMDn is read-only in 16-bit units.

Figure 10-19: Timer Dn counter register (TMDn) (n = 0, 1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMD0																	FFFF F540H	0000H
TMD1																	FFFF F550H	0000H

The conditions for which the TMDn register becomes 0000H are shown below.

- Reset input
- CAE bit = 0
- CE bit = 0
- Match of TMDn register and CMDn register
- Overflow

Cautions: 1. If the CAE bit of the TMCDn register is cleared to "0", a reset is performed asynchronously.

2. If the CE bit of the TMCDn register is cleared to "0", a reset is performed, synchronized with the internal clock. Similarly, a synchronized reset is performed after a match with the CMDn register and after an overflow.

3. The count clock must not be changed during a timer operation. If it is to be overwritten, it should be overwritten after the CE bit is cleared to "0".

4. Up to $f_{PCLK}/2$ clocks are required after a value is set in the CE bit until the set value is transferred to internal units. When a count operation begins, the count cycle from 0000H to 0001H differs from subsequent count cycles.

5. After a compare match is generated, the timer is cleared at the next count clock. Therefore, if the division ratio is large, the timer value may not be zero even if the timer value is read immediately after a match interrupt is generated.

(2) Timer Dn compare register (CMDn) (n = 0, 1)

CMDn and the TMDn registers' count value are compared, and an interrupt request signal (INTTMDn) is generated when a match occurs. TMDn is cleared, synchronized with this match. If the CAE bit of the TMCDn register is set to "0", a reset is performed asynchronously, and the registers are initialized (n = 0, 1).

The CMDn register is configured with a master/slave configuration. When a write operation to a CMDn register is performed, data is first written to the master register and then the master register's data is transferred to the slave register. In a compare operation, the slave register's value is compared with the count value of the TMDn register. When a read operation to a CMDn register is performed, data in the master side is read out.

CMDn can be read/written in 16-bit units.

Figure 10-20: Timer Dn Compare Register (CMDn) (n = 0, 1)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
CMD0																	FFFF F542H	0000H
CMD1																	FFFF F552H	0000H

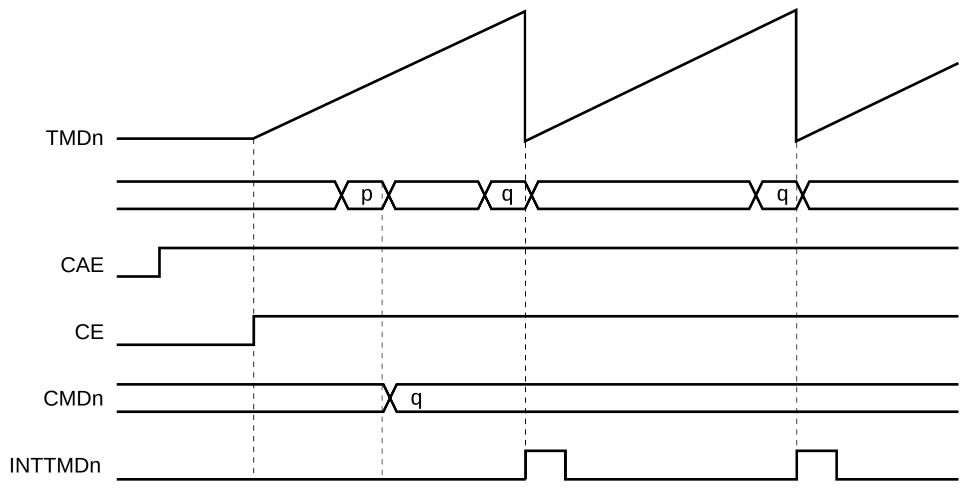
Cautions: 1. A write operation to the a CMDn register requires $f_{PCLK}/2$ clocks until the value that was set in the CMDn register is transferred to internal units. When writing continuously to the CMDn register, be sure to reserve a time interval of at least $f_{PCLK}/2$ clocks.

2. The CMDn register can be overwritten only once in a single TMDn register cycle (from 0000H until an INTTMDn interrupt is generated due to a match of the TMDn register and CMDn register). If this cannot be secured by the application, make sure that the CMDn register is not overwritten during timer operation.

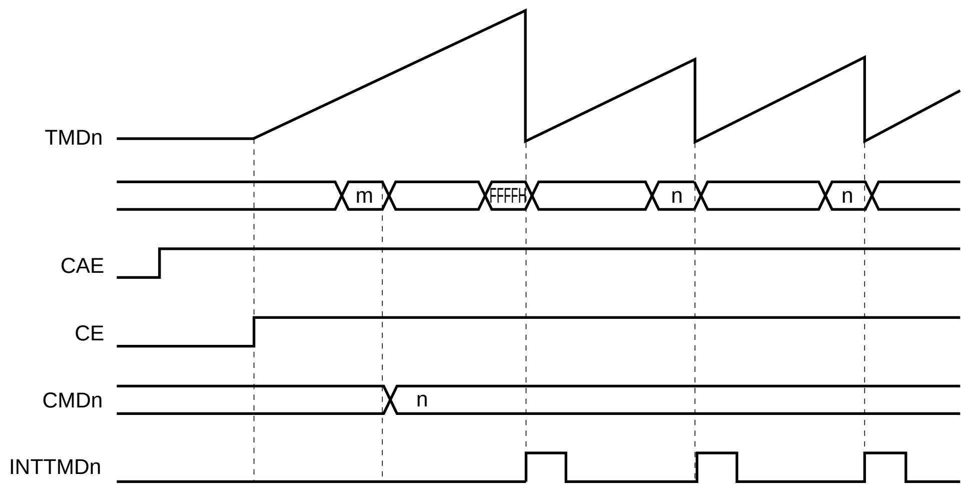
3. Note that an INTTMDn interrupt will be generated after an overflow if a value less than the counter value is written in the CMDn register during TMDn register operation (Figure 10-21, "Timing of Timer Dn Operation," on page 300).

Figure 10-21: Timing of Timer Dn Operation

(a) When TMDn < CMDn



(b) When TMDn > CMDn



Remarks: 1. p = TMDn value when overwritten

2. q = CMDn value when overwritten

3. n = 0, 1

10.2.4 Control register

(1) Timer Dn control register (TMCDn) (n = 0, 1)

The TMCDn register controls the operation of Timer Dn (n = 0, 1).

This register can be read/written in 8-bit or 1-bit units.

Figure 10-22: Timer Dn Control Register (TMCDn) (n = 0, 1)

	7	6	5	4	3	2	1	0	Address	Initial value
TMCD0	0	CS2	CS1	CS0	0	0	CE	CAE	FFFF F544H	00H
TMCD1	0	CS2	CS1	CS0	0	0	CE	CAE	FFFF F554H	00H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>6 to 4</td><td>CS2 to CS0</td><td>Selects the TMDn count clock (n = 0, 1)
<table>
<tr><th>CS2</th><th>CS1</th><th>CS0</th><th>Count Clock</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{PCLK} /2</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>f_{PCLK} /4</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>f_{PCLK} /8</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>f_{PCLK} /16</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>f_{PCLK} /32</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>f_{PCLK} /64</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>f_{PCLK} /128</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>f_{PCLK} /256</td></tr>
</table>
Caution: Do not change the CS2 to CS0 bits during timer operation. If they are to be changed, they must be changed after setting the CE bit to "0". If the CS2 to CS0 bits are overwritten during timer operation, the operation is not guaranteed.</td></tr>
<tr><td>1</td><td>CE</td><td>Count Enable: Controls the operation of TMDn (n = 0, 1).
0: Disable count (timer stopped at 0000H and does not operate)
1: Perform count operation
Caution: CE bit is not cleared even if a match is detected by the compare operation. To stop the count operation, clear the CE bit.</td></tr>
<tr><td>0</td><td>CAE</td><td>Count Action Enable: Controls the internal count clock.
0: Asynchronously reset entire TMDn unit. Stop clock supply to TMDn unit.
1: Supply clock to TMDn unit (n = 0, 1).
Cautions: 1. When CAE = 0 is set, the TMDn unit can be reset asynchronously.
2. When CAE = 0, the TMDn unit is in a reset state. To operate TMDn, first set CAE = 1.
3. When the CAE bit is changed from 1 to 0, all the registers of the TMDn unit are initialized. When again setting CAE = 1, be sure to then again set all the registers of the TMDn unit.</td></tr>
</table>

Caution: The CAE bit and CE bit cannot be set at the same time. Be sure to set the CAE bit prior to setting the CE bit.

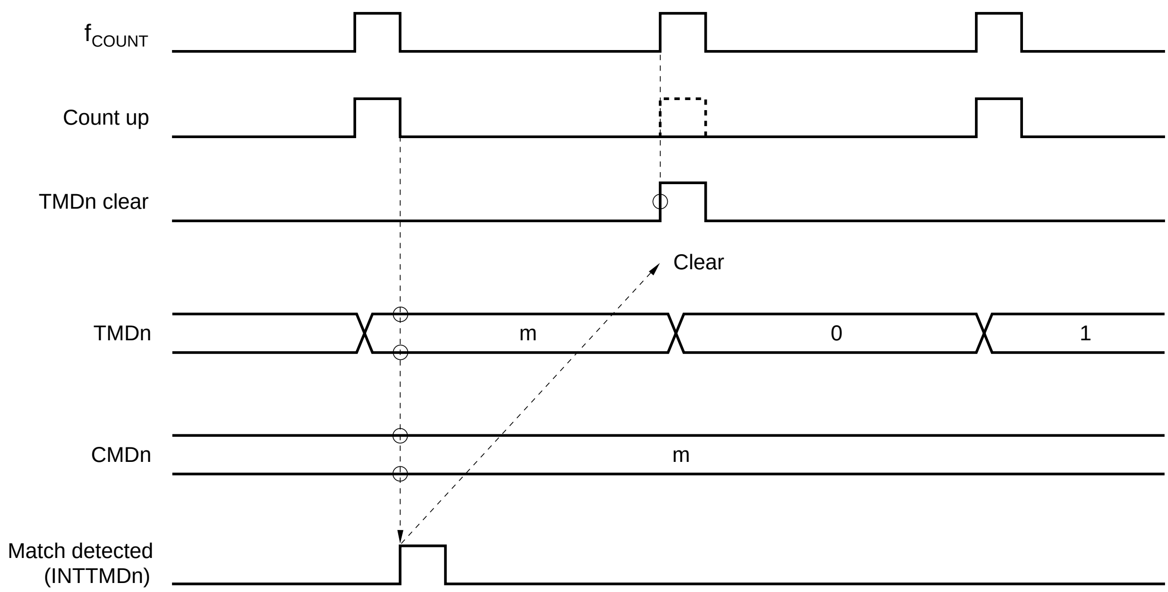
10.2.5 Operation

(1) Compare operation

TMDn can be used for a compare operation in which the value that was set in a compare register (CMDn) is compared with the TMDn count value (n = 0, 1).
If a match is detected by the compare operation, an interrupt (INTTMDn) is generated. The generation of the interrupt causes TMDn to be cleared to "0" at the next count timing. This function enables Timer Dn to be used as an interval timer.
CMDn can also be set to "0". In this case, when an overflow occurs and TMDn becomes "0", a match is detected and INTTMDn is generated. Although the TMDn value is cleared to "0" at the next count timing, INTTMDn is not generated according to this match.

Figure 10-23: Timing of Compare Operation (1/2)

(a) When CMDn is set to m (non-zero)



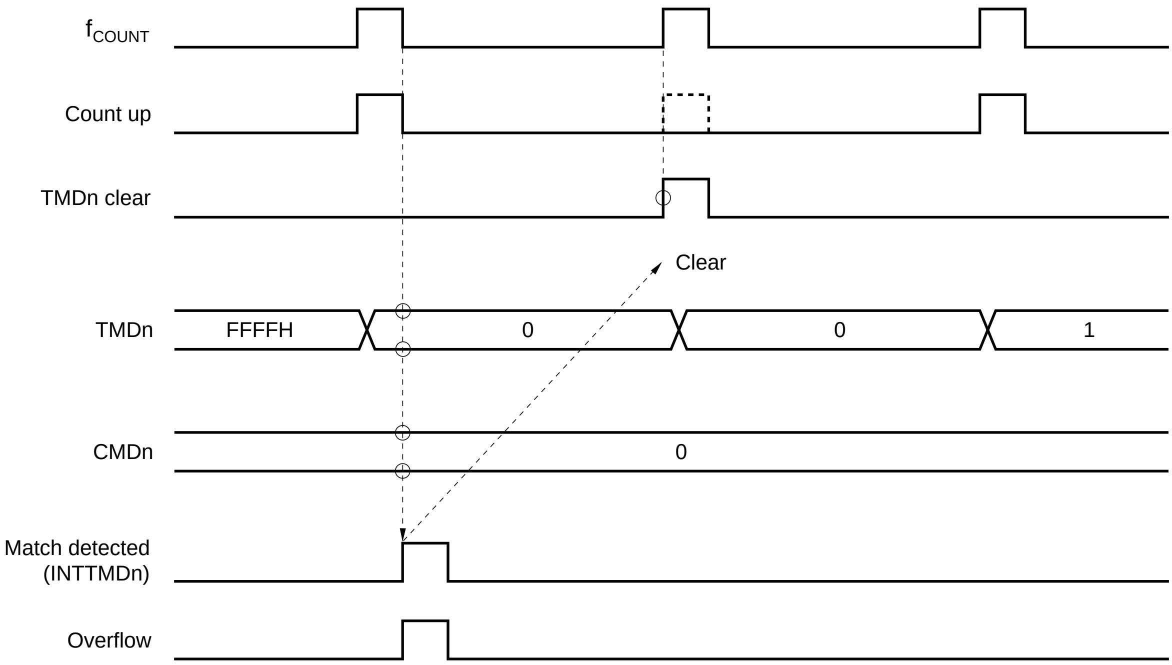
Remarks: 1. Interval time = (m + 1) × Count clock cycle

2. m = 1 to 65536 (FFFFH)

3. n = 0, 1

Figure 10-23: Timing of Compare Operation (2/2)

(b) When CMDn is set to 0



Remark: Interval time = (FFFFH + 2) × Count clock cycle

10.2.6 Application example

(1) Interval timer

This section explains an example in which Timer Dn is used as an interval timer with 16-bit precision.
Interrupt requests (INTTMDn) are output at equal intervals (refer to Figure 10-23, “Timing of Compare Operation (1/2),” on page 302). The setup procedure is shown below (n = 0, 1).

<1> Set the CAE bit to "1".

<2> Set each register:
Select the count clock using the CS2 to CS0 bits of the TMCDn register.
Set the compare value in the CMDn register.

<3> Start counting by setting the CE bit to "1".

<4> If the TMDn register and CMDn register’s values match, an INTTMDn interrupt is generated.

<5> INTTMDn interrupts are generated thereafter at equal intervals.

10.2.7 Precautions for Timer Dn

Various precautions concerning Timer Dn are shown below.

(1) To operate Timer Dn, first set to "1" the CAE bit of the TMCDn register.

(2) Up to $f_{PCLK}/2$ clocks are required after a value is set in the CE bit of the TMCDn register until the set value is transferred to internal units. When a count operation begins, the count cycle from 0000H to 0001H differs from subsequent count cycles.

(3) To initialize the TMDn register status and start counting again, clear the CE bit to "0" and then set the CE bit to "1" after an interval of $f_{PCLK}/2$ clocks has elapsed.

(4) Up to $f_{PCLK}/2$ clocks are required until the value that was set in the CMDn register is transferred to internal units. When writing continuously to the CMDn register, be sure to secure a time interval of at least $f_{PCLK}/2$ clocks.

(5) The CMDn register can be overwritten only once during a timer/counter operation (from 0000H until an INTTMDn interrupt is generated due to a match of the TMDn register and CMDn register). If this cannot be secured, make sure that the CMDn register is not overwritten during a timer/counter operation.

(6) The count clock must not be changed during a timer operation. If the clock selection by CS2 to CS0 bits is going to be changed, it should be overwritten after the CE bit is cleared to "0". If the count clock is changed during a timer operation, operation cannot be guaranteed.

(7) An INTTMDn interrupt will be generated after an overflow if a value less than the counter value is written in the CMDn register during TMDn register operation.

Remark: n = 0, 1

10.3 Timer G

2 x 16-bit multi purpose timer of Timer G are implemented:

- Timer G0
- Timer G1

10.3.1 Features of Timer G

The Timer Gn (n = 0, 1) operate as:

- Pulse interval and frequency measurement counter
- event counter
- Interval timer
- Programmable pulse output
- PWM output timer

Remark: In this Timer Gn chapter following indexes were consequently used

- m = 1 to 4 (for the free assignable Input/Output-channels)
- n = 0, 1 (for each of the 2 Timer G instance in Jupiter)
- x = 0, 1 (for bit-index, i.e. one of the 2 counters of each Timer Gn)
- y = 0 to 5 (for all of the 6 capture/compare-channels)

10.3.2 Function overview of each Timer Gn

- 16-bit timer/counter (TMGn0, TMGn1): 2 channels

- Bit length
  - Timer Gn registers (TMGn0, TMGn1): 16 bits

- Capture/compare register (GCCny): 6
  - 16-bit
  - 2 registers are assigned fix to the corresponding one of the 2 counters
  - 4 free assignable registers to one of the 2 counters

- Count clock division selectable by prescaler (frequency of peripheral clock: f_{PCLK} = 16 MHz)
  - In 8 steps from $f_{PCLK}/2$ to $f_{PCLK}/256$

- Interrupt request sources
  - Edge detection circuit with noise elimination.
  - Compare-match interrupt requests: 6 types
    Perform comparison of capture/compare register with one of the 2 counters (TMGn0, TMGn1) and generate the INTCCGny (y = 0 to 5) interrupt upon compare match.
  - Timer counter overflow interrupt requests: 2 types
    In free run mode the INTTMGn0 (INTTMGn1) interrupt is generated when the count value of TMGn0 (TMGn1) toggles from FFFFH to 0000H.
  - In match and clear mode the INTTMGn0 (INTTMGn1) interrupt is generated when the count value of TMGn0 (TMGn1) matches the GCC0 (GCC1) value.

- PWM output function
  - Control of the outputs of TOGn1- through TOGn4-pin in the compare mode. PWM output can be performed using the compare match timing of the GCCn1 to GCCn4 register and the corresponding timebase (TMGn0, TMGn1).

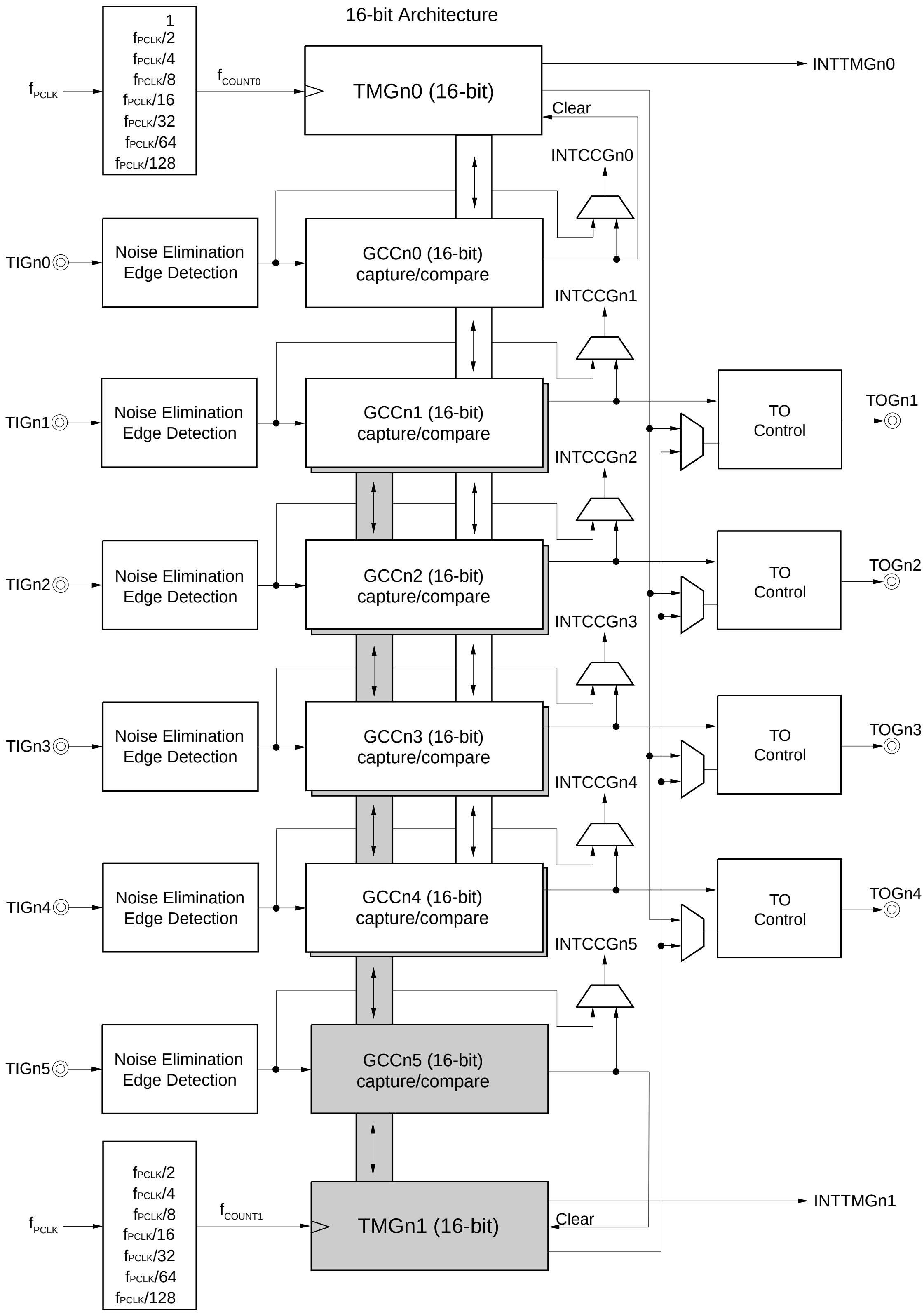
- Output delay operation
  - A clock-synchronized output delay can be added to the output signal of pins TOGn1 to TOGn4.
  - This is effective as an EMI counter measure.

- Edge detection and noise elimination filter
  - External signals shorter than 1 count clock (f_{COUNT}, not f_{PCLK}) are eliminated as noise.

Note: The TOGn1 to TOGn4 and TOGn1 to TOGn4 are each alternate function pins.

Figure 10-24, “Block Diagram of Timer Gn,” on page 308 shows the block diagram of Timer Gn.

Figure 10-24: Block Diagram of Timer Gn



Remark: f_{PCLK}: Internal peripheral clock (16 MHz)

Note: TMGn0/TMGn1 are cleared by GCCn0/GCCn5 register compare match.

10.3.3 Basic configuration

The basic configuration is shown below.

Table 10-4: Timer Gn Configuration List

Timer	Count Clock	Register	R/W	Generated Interrupt Signal	Capture Trigger	Timer Output PWM
Timer Gn	f_{PCLK} f_{PCLK} /2, f_{PCLK} /4, f_{PCLK} /8, f_{PCLK} /16, f_{PCLK} /32, f_{PCLK} /64, f_{PCLK} /128	TMGn0	R	INTTMGn0	-	-
		TMGn1	R	INTTMGn1	-	-
		GCCn0	R/W	INTCCGn0	TIGn0	-
		GCCn1	R/W	INTCCGn1	TIGn1	TOGn1
		GCCn2	R/W	INTCCGn2	TIGn2	TOGn2
		GCCn3	R/W	INTCCGn3	TIGn3	TOGn3
		GCCn4	R/W	INTCCGn4	TIGn4	TOGn4
		GCCn5	R/W	INTCCGn5	TIGn5	-

Remarks: 1. f_{PCLK}: Internal peripheral clock

2. n = 0, 1

(1) Timer Gn 16-bit counter registers (TMGn0, TMGn1)

The features of the 2 counters TMGn0 and TMGn1 are listed below:

- Free-running counter that enables counter clearing by compare match of registers GCCn0/GCCn5
- Counter clear can be set by software.
- Counter stop can be set by software.

These registers can be read in 16-bit units.

Figure 10-25: Timer Gn Counter 0 Value Registers TMGn0

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMG00																	FFFF F648H	0000H
TMG10																	FFFF F688H	0000H

Figure 10-26: Timer Gn Counter 1 Value Registers TMGn1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMG01																	FFFF F64AH	0000H
TMG11																	FFFF F68AH	0000H

(2) Timer Gn capture/compare registers of the 2 counters (GCCn0, GCCn5)

The GCCn0, GCCn5 registers are 16-bit capture/compare registers of Timer Gn. These registers are fixed assigned to the counter registers (TMGn0 and TMGn1).

In the **capture register mode**, GCCn0 (GCCn5) captures the TMGn0 (TMGn1) count value if an edge is detected at Pin TIGn0 (TIGn5).

In the **compare register mode**, GCCn0 (GCCn5) detects match with TMGn0 (TMGn1) and clears the assigned Timebase. So this “match and clear mode” is used to reduce the number of valid bits of the counter TMGn0 (TMGn1).

These registers can be read/written in 16-bit units.

Caution: If in Compare Mode write to this registers <u>before</u> POWER and ENFGx bit (x = 0, 1) are "1" at the same time.

Figure 10-27: Timer Gn counter TMGn0 assigned Capture/Compare Register (GCCn0)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
GCC00																	FFFF F64CH	0000H
GCC10																	FFFF F68CH	0000H

Remark: This register is assigned fix to timebase TMGn0.

Figure 10-28: Timer Gn counter TMGn1 assigned Capture/Compare Register (GCCn5)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
GCC05																	FFFF F656H	0000H
GCC15																	FFFF F696H	0000H

Remark: This register is assigned fix to timebase TMGn1.

(3) Timer G capture/compare registers with external PWW-output function (GCCn1 to GCCn4)

The GCCn1 to GCCn4 registers are 16-bit capture/compare registers of Timer Gn. They can be assigned to one of the 2 counters either TMGn0 or TMGn1.

In the **capture register mode**, these registers capture the value of TMGn0 when the TBGm bit (m = 1 to 4) of the TMGCMHn register = 0. When the TBGm bit = 1, these registers hold the value of TMGn1.

In **compare mode**, these registers represent the actual compare value and the TOGnm-Output (m = 1 to 4) can generate a PWW if they are activated.

These registers can be read/written in 16-bit units.

Figure 10-29: Timer Gn free assignable Capture/Compare Registers (GCCnm) (m = 1 to 4)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
GCC01																	FFFF F64EH	0000H
GCC11																	FFFF F68EH	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
GCC02																	FFFF F650H	0000H
GCC12																	FFFF F690H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
GCC03																	FFFF F652H	0000H
GCC13																	FFFF F692H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
GCC04																	FFFF F654H	0000H
GCC14																	FFFF F694H	0000H

Remarks: **1.** In capture mode only reading is possible

2. In compare mode read/write is possible

10.3.4 Control registers

(1) Timer Gn Mode Register (TMGMn) (n = 0, 1)

This register can be read/written in 16-bit, 8-bit or 1-bit units.

Figure 10-30: Timer Gn Mode Register (TMGMn) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMGM0	POWER	OLDE	CSE12	CSE11	CSE10	CSE02	CSE01	CSE00	CCSG5	CCSG0	0	0	CLRG1	TMG1E	CLRG0	TMG0E	FFFF F640H	0000H
TMGM1	POWER	OLDE	CSE12	CSE11	CSE10	CSE02	CSE01	CSE00	CCSG5	CCSG0	0	0	CLRG1	TMG1E	CLRG0	TMG0E	FFFF F680H	0000H

Bit Position	Bit Name	Function
15	POWER	Timer Gn Operation control. 0: operation Stop the capture registers and TMGSTn register are cleared the TOGnm pins (m = 1 to 4) are inactive all the time 1: operation enable **Remark:** At least 7 peripheral-clocks (f_{PCLK}) are need to start the timer function
14	OLDE	Set Output Delay Operation. 0: Don’t perform output delay operation 1: Set output delay to n count-clocks **Caution: When the POWER-Bit is set, the rewriting of this Bit is prohibited! Simultaneously writing with the POWER bit is allowed.** **Remark:** The delay operation is used for EMI counter measures.
13 to 8	CSEx2, CSEx1, CSEx0	Selects internal count clock of TMG (see table below) **Caution: When the POWER-Bit is set, the rewriting of this Bits are prohibited! Simultaneously writing with the POWER bit is allowed.** **Remarks: 1.** x = 0, 1 **2.** f_{PCLK}: peripheral clock

CSEx2	CSEx1	CSEx0	Count Clock
0	0	0	f_{PCLK}
0	0	1	$f_{PCLK}/2$
0	1	0	$f_{PCLK}/4$
0	1	1	$f_{PCLK}/8$
1	0	0	$f_{PCLK}/16$
1	0	1	$f_{PCLK}/32$
1	1	0	$f_{PCLK}/64$
1	1	1	$f_{PCLK}/128$

Figure 10-30: Timer Gn Mode Register (TMGMn) (2/2)

Bit Position	Bit Name	Function
7, 6	CCSG5n, CCSG0n	Specifies the mode of the TMGn0 (TMGn1)(CCSG5n for TMGn1, CCSG0n for TMGn0): 0: Free-run mode for TMGn1 (TMGn0), GCCn5 (GCCn0) in capture mode (an detected edge at Pin TIGn5 (TIGn0) stores the value of TMGn1 (TMGn0) in GCCn5 (GCCn0) and an interrupt INTCCGn5 (INTCCGn0) is output) 1: Match and Clear mode of the TMGn1 (TMGn0), GCCn5 (GCCn0) in compare mode (when the data of GCCn5 (GCCn0) match the count value of the TMGn1 (TMGn0), the counter is cleared and the interrupt INTCCGn5 (INTCCGn0) occurs) **Caution: When the POWER bit is set, the rewriting of this Bits are prohibited! Simultaneously writing with the POWER bit is allowed.**
3, 1	CLRGx	Specifies software clear for TMGx (x = 0, 1): 0: Continue TMGx operation 1: Clears (0) the count value of TMGx, the corresponding TOGx is deactivated. **Remark:** TMGx starts 1 peripheral-clock after this bit is set this bit is not readable (always read 0)
2, 0	TMGxE	Specifies TMGx (x = 0, 1) count operation enable/disable 0: Stop count operation the counter holds the immediate preceding value the corresponding TOGx is deactivated 1: Enable count operation **Remarks: 1.** the counter needs at least 1 peripheral-clock (f_{PCLK}) to stop **2.** the counter needs at least 4 peripheral-clocks (f_{PCLK}) to start

(2) Timer Gn Mode Register Low (TMGMnL) (n = 0, 1)

This register is the low byte of the TMGMn register.

This register can be read/written in 8-bit or 1-bit units.

Figure 10-31: Timer Gn Mode Register Low (TMGMnL)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGM0L	CCSG5	CCSG0	0	0	CLRG1	TMG1E	CLRG0	TMG0E	FFFF F640H	00H
TMGM1L	CCSG5	CCSG0	0	0	CLRG1	TMG1E	CLRG0	TMG0E	FFFF F680H	00H

The explanation of the bit 7 to 0 is the same as the bit 7 to 0 of the TMGMn register.

(3) Timer Gn Mode Register High (TMGMnH) (n = 0, 1)

This register is the high byte of the TMGMn register.

This register can be read/written in 8-bit or 1-bit units.

Figure 10-32: Timer Gn Mode Register Low (TMGMnH)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGM0H	POWER	OLDE	CSE12	CSE11	CSE10	CSE02	CSE01	CSE00	FFFF F641H	00H
TMGM1H	POWER	OLDE	CSE12	CSE11	CSE10	CSE02	CSE01	CSE00	FFFF F681H	00H

The explanation of the bit 7 to 0 is the same as the bit 15 to 8 of the TMGMn register.

(4) Timer Gn Channel Mode Register (TMGCMn)

This register specifies the assigned counter (TMGn0 or TMGn1) for the GCCnm register. Furthermore it specifies the edge detection for the TIGy-input-pins (y = 0 to 5).

This register can be read/written in 16-bit, 8-bit or 1-bit units.

Figure 10-33: Timer Gn Channel Mode Register (TMGCMn)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
TMGCM0H	TBG4	TBG3	TBG2	TBG1	IEG51	IEG50	IEG41	IEG40	IEG31	IEG30	IEG21	IEG20	IEG11	IEG10	IEG01	IEG00	FFFF 642H	0000H
TMGCM1H	TBG4	TBG3	TBG2	TBG1	IEG51	IEG50	IEG41	IEG40	IEG31	IEG30	IEG21	IEG20	IEG11	IEG10	IEG01	IEG00	FFFF 682H	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 12</td><td>TBGm</td><td>Assigns Capture/Compare registers GCCn1 to GCCn4 to one of the 2 counters TMGn0 or TMGn1:
0: Set TMGn0 as the corresponding counter to GCCnm register and TIGm/ TOGnm-pin
1: Set TMGn1 as the corresponding counter to GCCnm register and TIGm/ TOGnm-pin</td></tr>
<tr><td>11 to 4</td><td>IEGy1, IEGy0</td><td>Specifies the valid edge of external capture signal input pin (TIGm) for the capture register performing capture-match with the assigned counter TMGn0 or TMGn1:
<table><tr><th>IEGy1</th><th>IEGy0</th><th>Valid Edge</th></tr><tr><td>0</td><td>0</td><td>Falling edge</td></tr><tr><td>0</td><td>1</td><td>Rising edge</td></tr><tr><td>1</td><td>0</td><td>No edge detection performed</td></tr><tr><td>1</td><td>1</td><td>Both rising and falling edges</td></tr></table></td></tr>
</table>

Remarks: 1. y = 0 to 5

2. m = 1 to 4

(5) Timer Gn Channel Mode Register Low (TMGCMnL)

This register is the low byte of the TMGCMn register.

This register can be read/written in 8-bit or 1-bit units.

Figure 10-34: Timer Gn Channel Mode Register (TMGCMnL)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGCM0L	IEG31	IEG30	IEG21	IEG20	IEG11	IEG10	IEG01	IEG00	FFFF F642H	00H
TMGCM1L	IEG31	IEG30	IEG21	IEG20	IEG11	IEG10	IEG01	IEG00	FFFF F682H	00H

The explanation of the bit 7 to 0 is the same as the bit 7 to 0 of the TMGCMnH register.

(6) Timer Gn Channel Mode Register Low (TMGCMnH)

This register is the high byte of the TMGCMnH register.

This register can be read/written in 8-bit or 1-bit units.

Figure 10-35: Timer Gn Channel Mode Register (TMGCMnH)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGCM0H	TBG4	TBG3	TBG2	TBG1	IEG51	IEG50	IEG41	IEG40	FFFF F643H	00H
TMGCM1H	TBG4	TBG3	TBG2	TBG1	IEG51	IEG50	IEG41	IEG40	FFFF F683H	00H

The explanation of the bit 7 to 0 is the same as the bit 15 to 8 of the TMGCMnH register.

(7) Timer Gn output control register (OCTLGn)

This register controls the timer output from the TOGm pin (m = 1 to 4) and the capture or compare modus for the GCCnm register.

This register can be read/written in 16-bit, 8-bit or 1-bit units.

Cautions: 1. When the POWER bit is set, the rewriting of CCSGm is prohibited

2. When the POWER bit and TMG0E bit (TMG1E bit) are set at the same time, the rewriting of the ALVGm bits is prohibited.

Figure 10-36: Timer Gn Output Control Register (OCTLGn)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
OCTLG0H	SWFG4	ALVG4	CCSG4	0	SWFG3	ALVG3	CCSG3	0	SWFG2	ALVG2	CCSG2	0	SWFG1	ALVG1	CCSG1	0	FFFF F644H	4444H
OCTLG1H	SWFG4	ALVG4	CCSG4	0	SWFG3	ALVG3	CCSG3	0	SWFG2	ALVG2	CCSG2	0	SWFG1	ALVG1	CCSG1	0	FFFF F684H	4444H

Bit Position	Bit Name	Function
15, 11, 7, 3	SWFGm	Fixes the TOGnm pin output level according to the setting of ALVGm bit. 0: disable TOGnm to inactive level 1: enable TOGnm
14, 10, 6, 2	ALVGm	Specifies the active level of the TGOm pin output. 0: Active level is 0 1: Active level is 1 **Caution: Don’t write this bit, before ENFG0 or ENFG1 of TMGSTn is 0, so first clear TMG0E or TMG1E bit of the TMGMn register and check ENFG0 or ENFG1 bit before writing.**
13, 9, 5, 1	CCSGm	Specifies Capture/Compare mode selection: 0: Capture mode: if external edge is detected the INTCCGnm interrupt occurs, the corresponding counter value is written to GCCnm 1: Compare mode: if GCCnm matches with corresponding timebase the INTCCGnm interrupt occurs, if SWFGm is set the PWM output mode is set **Caution: Don’t write this bit, before POWER bit of TMGMHn is 0.**

Remark: m = 1 to 4

(8) Timer Gn output control register Low (OCTLGnL)

This register is the low byte of the OCTLGnH register.

This register can be read/written in 8-bit or 1-bit units.

Figure 10-37: Timer Gn Output Control Register Low (OCTLGnL)

	7	6	5	4	3	2	1	0	Address	Initial value
OCTLG0L	SWFG2	ALVG2	CCSG2	0	SWFG1	ALVG1	CCSG1	0	FFFF F644H	44H
OCTLG1L	SWFG2	ALVG2	CCSG2	0	SWFG1	ALVG1	CCSG1	0	FFFF F684H	44H

The explanation of the bit 7 to 0 is the same as the bit 7 to 0 of the OCTLGn register.

(9) Timer Gn output control register High (OCTLGnH)

This register is the low byte of the OCTLGnH register.

This register can be read/written in 8-bit or 1-bit units.

Figure 10-38: Timer Gn Output Control Register High (OCTLGnH)

	7	6	5	4	3	2	1	0	Address	Initial value
OCTLG0H	SWFG4	ALVG4	CCSG4	0	SWFG3	ALVG3	CCSG3	0	FFFF F645H	44H
OCTLG1H	SWFG4	ALVG4	CCSG4	0	SWFG3	ALVG3	CCSG3	0	FFFF F685H	44H

The explanation of the bit 7 to 0 is the same as the bit 15 to 8 of the OCTLGn register.

(10) Time base status register (TMGSTn)

The TMGSTn register indicates the status of TMGn0 and TMGn1. For the CCFGy bit see Chapter 10.3.7 "Operation in Free-run mode" on page 324.

This register can be read in 8-bit or 1-bit units.

Figure 10-39: Timer Gn Status Register (TMGSTn)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGST0	ENFG1	ENFG2	CCFG5	CCFG4	CCFG3	CCFG2	CCFG1	CCFG0	FFFF F646H	00H
TMGST1	ENFG1	ENFG2	CCFG5	CCFG4	CCFG3	CCFG2	CCFG1	CCFG0	FFFF F686H	00H

Bit Position	Bit Name	Function
5 to 0	CCFGy	Indicates TMGn0 or TMGn1 overflow status. 0: No overflow 1: Overflow **Caution: The CCFGy bit is set if a TMGn0 (TMGn1) overflow occurs. This flag is only updated if the corresponding GCCy register was read, so first read the GCCy register and then read this flag if necessary**
7 to 6	ENFG1, ENFG0	Indicates TMGn1 (TMGn0) operation. 0: indicates operation stopped 1: indicates operation

Remarks: 1. y = 0 to 5

2. n = 0, 1

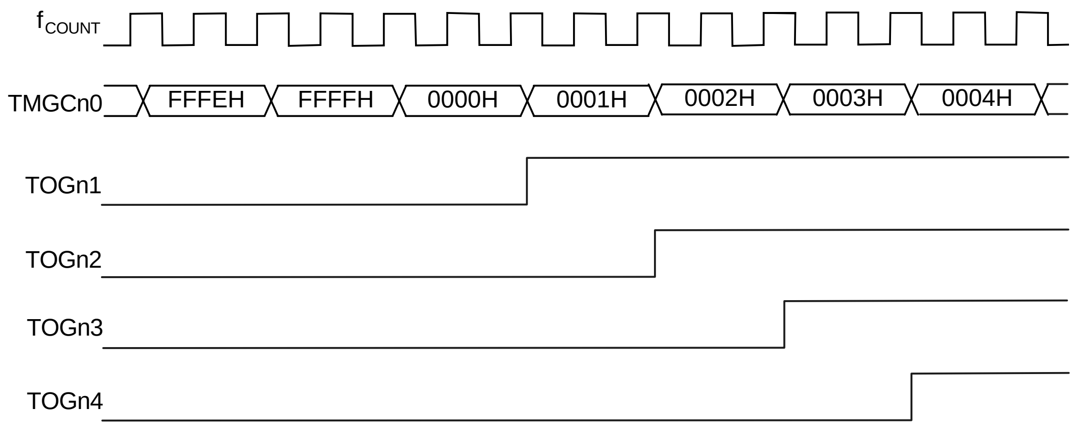
10.3.5 Output delay operation

When the OLDE bit is set, different delays of count clock period are added to the TOGnm pins:

Output-pin	delay $1/f_{COUNT}$
TOGn1	0
TOGn2	1
TOGn3	2
TOGn4	3

The figure below shows the timing for the case where the count clock is set to $f_{PCLK}/2$. However, 0FFFH is set in GCCn0.
Similar delays are added also when a transition is made from the active to inactive level. So, a relative pulse width is guaranteed.

Figure 10-40: Timing of Output delay operation



In this case the count clock is set to $f_{PCLK}/2$.

10.3.6 Explanation of basic operation

(1) Overview of the mode settings

The Timer Gn includes 2 channels of 16-bit counters (TMGn0/TMGn1), which can operate as independently timebases. TMGn0 (TMGn1) can be set by CCSG0 bit (CCSG5 bit) in the following modes:
- free-run mode,
- match and clear mode.

When a timer output (TOGnm) or INTCCGnm interrupt is used, one of the two counters can be selected by setting the TBGm bit (m = 1 to 4) of the TMGCMHn register.
The tables below indicate the interrupt output and timer output states dependent on the register setting values.

Table 10-5: Interrupt output and timer output states dependent on the register setting values

<table>
<tr><th colspan="4">Register setting value</th><th colspan="4">State of each output pin</th></tr>
<tr><th>CCSG0n</th><th>TBGm</th><th>SWFGm</th><th>CCSGm</th><th>INTTMGn0</th><th>INTCCGn0</th><th>INTCCGnm</th><th>TOGnm</th></tr>
<tr><td rowspan="4">0
Free-run mode</td><td rowspan="8">0</td><td rowspan="2">0</td><td>0</td><td rowspan="4">Overflow interrupt</td><td rowspan="4">TI0 edge detection</td><td>TIm edge detection</td><td rowspan="3">Tied to inactive level</td></tr>
<tr><td>1</td><td>CMPGm match</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>TIm edge detection</td></tr>
<tr><td>1</td><td>CMPGm match</td><td>PWM
(free run)</td></tr>
<tr><td rowspan="4">1
Match and clear mode</td><td rowspan="2">0</td><td>0</td><td rowspan="4">Overflow interruptNote 1</td><td rowspan="4">CMPG0 matchNote 2</td><td>TIm edge detection</td><td rowspan="3">Tied to inactive level</td></tr>
<tr><td>1</td><td>CMPGm match</td></tr>
<tr><td rowspan="2">1</td><td>0</td><td>TIm edge detection</td></tr>
<tr><td>1</td><td>CMPGm match</td><td>PWM
(match and clear)</td></tr>
</table>

Notes: 1. An interrupt is generated only when the value of the GCCn0 register is FFFFH.

2. An interrupt is generated only when the value of the GCCn0 register is not FFFFH.

Remark: The setting of the CCSGm bit in combination with the SWFGm bit sets the mode for the timing of the actualization of new compare values.

- In compare mode the new compare value will be immediately active.
- In PWM mode the new compare value will be active first after the next overflow or match & clear of the assigned counter (TMG0, TMG1).

Table 10-6: Interrupt output and timer output states dependent on the register setting values

Register setting value				State of each output pin			
CCSG5n	TBGm	SWFGm	CCSGm	INTTMGn1	INTCCGn5	INTCCGnm	TOGnm
0 Free-run mode	1	0	0	Overflow interrupt	TI5 edge detection	TIm edge detection	Tied to inactive level
			1			CMPGm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (free run)
1 Match and clear mode		0	0	Overflow interrupt[Note 1]	CMPG5 match[Note 2]	TIm edge detection	Tied to inactive level
			1			CMPGm match	
		1	0			TIm edge detection	
			1			CMPGm match	PWM (match and clear)

Notes: 1. An interrupt is generated only when the value of the GCC5 register is FFFFH.

2. An interrupt is generated only when the value of the GCC5 register is not FFFFH.

Remark: The setting of the CCSGm bit in combination with the SWFGm bit sets the mode for the timing of the actualization of new compare values.

- In compare mode the new compare value will be immediately active.
- In PWM mode the new compare value will be active first after the next overflow or match & clear of the assigned counter (TMG0, TMG1).

10.3.7 Operation in Free-run mode

This operation mode is the standard mode for Timer Gn operations. In this mode the 2 counter TMGn0 and TMGn1 are counting up from 0000H to FFFFH, generates an overflow and start again. In the match and clear mode, which is described in Chapter 10.3.8 on page 335 the fixed assigned register GCCn0 (GCCn5) is used to reduce the bit-size of the counter TMGn0 (TMGn1).

(1) Capture operation (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGm	0	disable TOGnm
TBGm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Pulse width or period measurement of the TIGny input signal (free run)

Capture setting method:

(1) When using one of the TOGn1 to TOGn4 pins, select the corresponding counter with the TBGm bit. When TIGn0 is used, the corresponding counter is TMGn0. When TIGn5 is used, the corresponding counter is TMGn1.
(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1) or CSE02 to CSE02 bits (TMGn0).
(3) Select a valid TIGny edge with the IEGy1 and IEGy0 bits. A rising edge, falling edge, or both edges can be selected.
(4) Start timer operation by setting POWER bit and TMG0E bit for TMGn0 or TMG1E bit for TMGn1.

Capture Operation

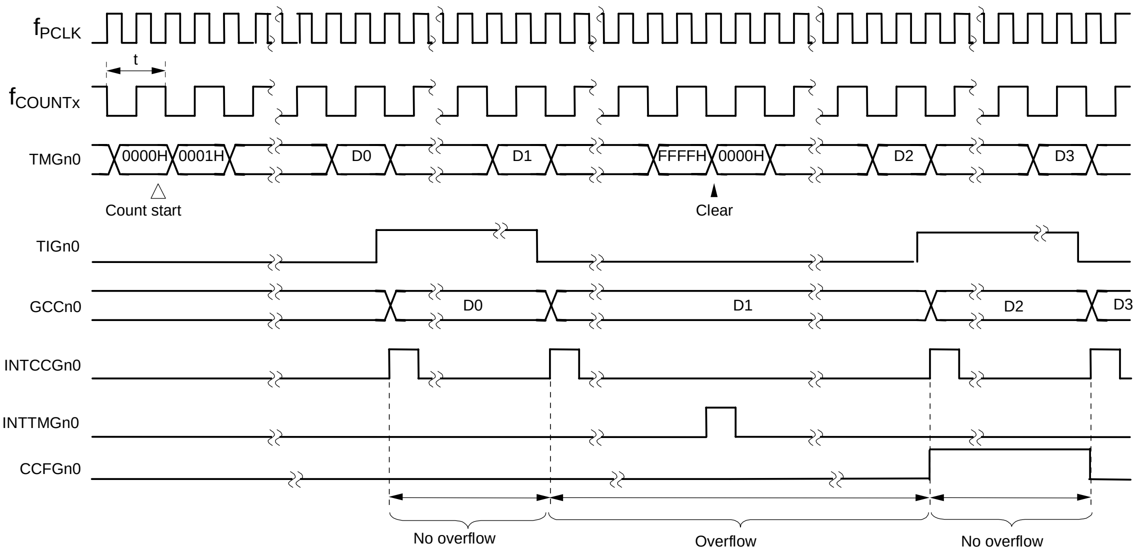
(1) When a specified edge is detected, the value of the counter is stored in GCCny and an edge detection interrupt (INTCCGny) is output.
(2) When the counter overflows, an overflow interrupt (INTTMGn0 or INTTMGn1) is generated.
(3) If an overflow has occurred between capture operations, the CCFGy flag is set when GCCny is read. Correct capture data by checking the value of CCFGy.

Using CCFGy:
When using GCCny as a capture register, use the procedure below.

<1> After INTCCGny (edge detection interrupt) generation, read the corresponding GCCny register.
<2> Check if the corresponding CCFGy bit of the TMGSTn register is set.
<3> If the CCFGy bit is set, the counter was cleared from the previous captured value.
CCFGy is set when GCCny is read. So, after GCCny is read, the value of CCFGy should be read. Using the procedure above, the value of CCFGy corresponding to GCCny can be read normally.

Caution: If two or more overflows occur between captures, a software-based measure needs to be taken to count overflow interrupts (INTTMGn0, INTTMGn1).

Figure 10-41: Timing when both edges of TIGn0 are valid (free run)



Remark: The figure above shows an image. In actual circuitry, 3 to 4 periods of the count-up signal are required from the input of a waveform to TIGn0 until a capture interrupt is output. See Chapter **10.1.3 Basic configuration**, **(1)16-bit counter (TMC0)**, (b)“Synchronous reset” on page 275.

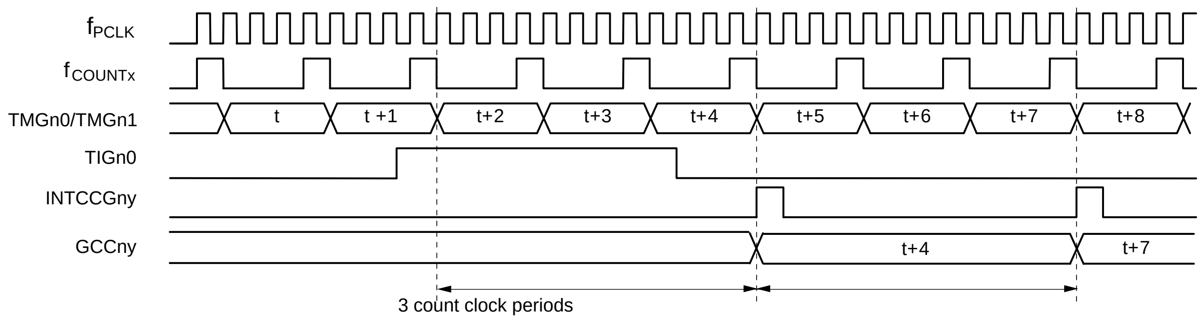
(b) Timing of capture trigger edge detection

The Tin inputs are fitted with an edge-detection and noise-elimination circuit.
Because of this circuit, 3 periods to less than 4 periods of the count clock are required from edge input until an interrupt signal is output and capture operation is performed. The timing chart is shown below.

Basic settings (x= 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{PCLK}/4$
CSEx1	1	
CSEx0	0	
IEGy1	1	detection of both edges
IEGy0	1	

Figure 10-42: Timing of capture trigger edge detection (free run)



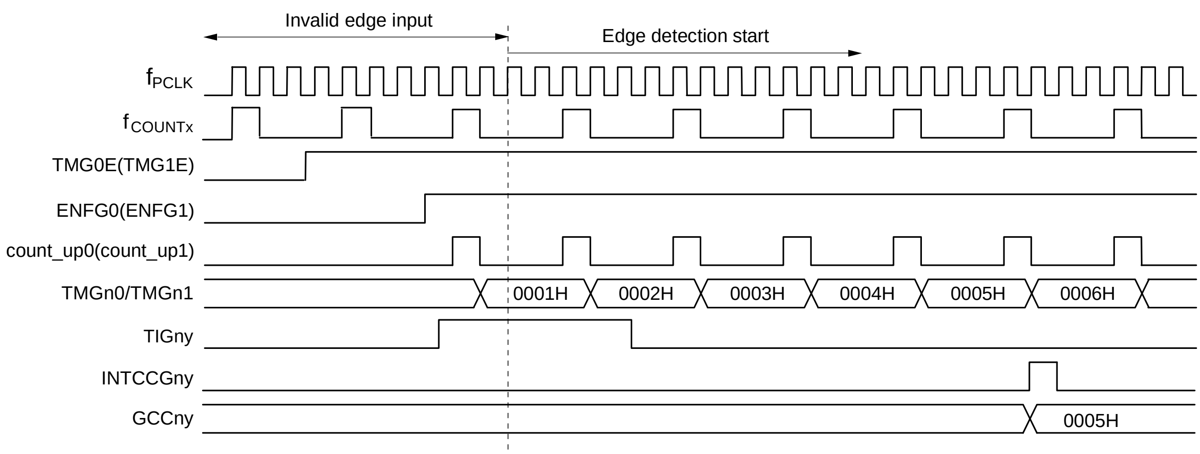
(c) Timing of starting capture trigger edge detection

A capture trigger input signal (TIGny) is synchronized in the noise eliminator for internal use. Edge detection starts when 1 count clock period (f_{COUNT}) has been input after timer count operation starts. (This is because masking is performed to prevent the initial TIGny level from being recognized as an edge by mistake.). The timing chart for starting edge detection is shown below.

Basic settings (x= 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEx2	0	Count clock = $f_{PCLK}/4$
CSEx1	1	
CSEx0	0	
IEGy1	1	detection of both edges
IEGy0	1	

Figure 10-43: Timing of starting capture trigger edge detection



(2) Compare operation (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSG0	0	free run mode
CCSG5	0	
SWFGm	0	disable TOGnm
CCSGm	1	Compare mode for GCCnm
TBGm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Interval timer (free run)

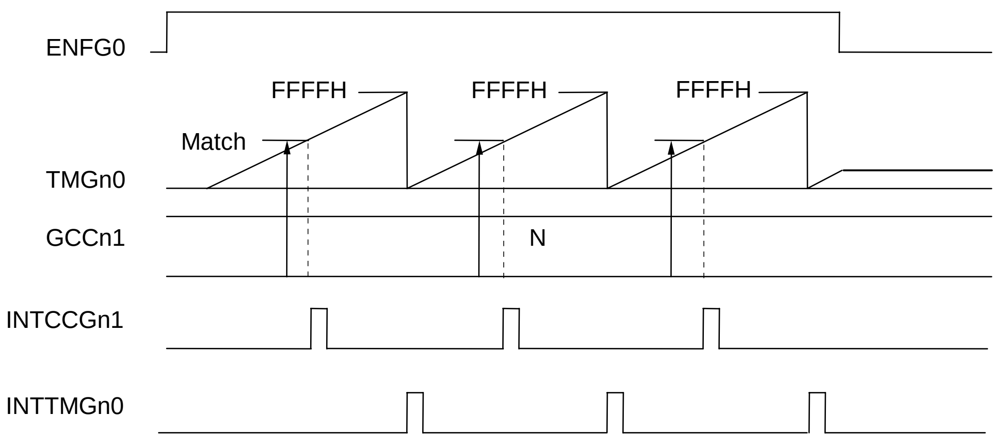
Setting method interval timer:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter (TMGn0 or TMGn1) must be selected with the TBGm bit.
(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1 register) or CSE02 to CSE00 bits (TMGn0 register).
(3) Write data to GCCnm.
(4) Start timer operation by setting POWER and TMG0E (or TMG1E).

Compare Operation:

(1) When the value of the counter matches the value of GCCnm (m = 0 to 4), a match interrupt (INTCCGnm) is output.
(2) When the counter overflows, an overflow interrupt (INTTMGn0/INTTMGn1) is generated.

Figure 10-44: Timing of compare mode (free run)



Data N is set in GCCn1, and the counter TMGn0 is selected.

(b) When the value 0000H is set in GCCnm

INTCCGnm is activated when the value of the counter becomes 0001H.
INTTMGn0/INTTMGn1 is activated when the value of the counter changes from FFFFH to 0000H.
Note, however, that even if no data is set in GCCnm, INTCCGnm is activated immediately after the counter starts.

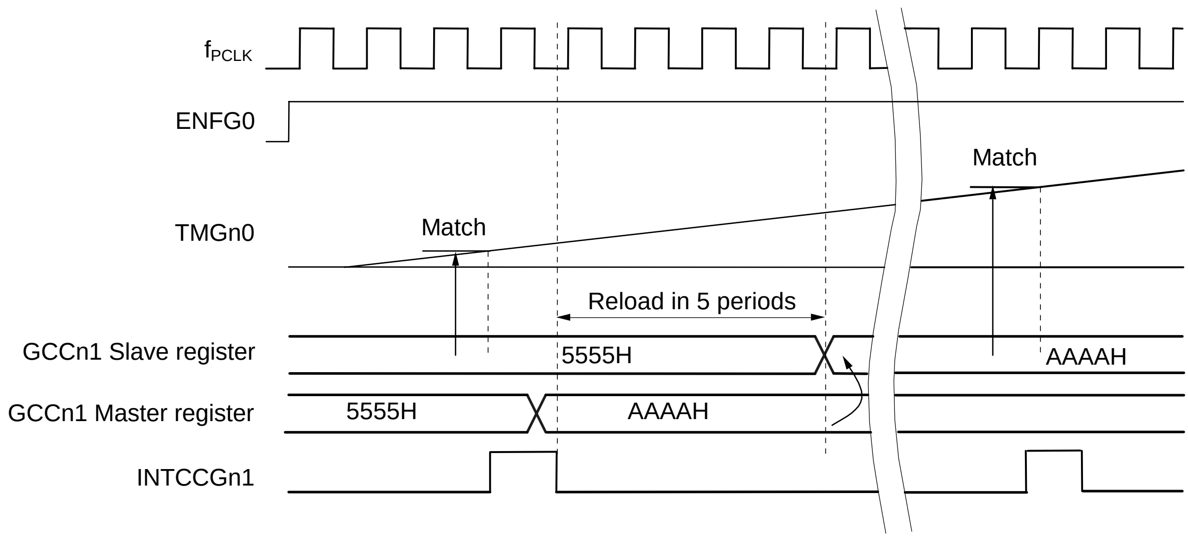
(c) When the value FFFFH is set in GCCnm

INTCCGnm and INTTMGn0/INTTMGn1 are activated when the value of the counter changes from FFFFH to 0000H.

(d) When GCCnm is rewritten during operation

When GCCn1 is rewritten from 5555H to AAAAH. TMGn0 is selected as the counter.
The following operation is performed:

Figure 10-45: Timing when GCCn1 is rewritten during operation (free run)



Caution: To perform successive write access during operation, for rewriting the GCCny register (n = 1 to 4), you have to wait for minimum 7 peripheral clocks periods (f_{PCLK}).

(3) PWM output (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSG0	0	free run mode
CCSG5	0	
SWFGm	1[Note]	enable TOGnm
CCSGm	1[Note]	Compare mode for GCCnm
TBGm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

Note: The PWM mode is activated by setting the SWFGm and the CCSGm bit to "1".

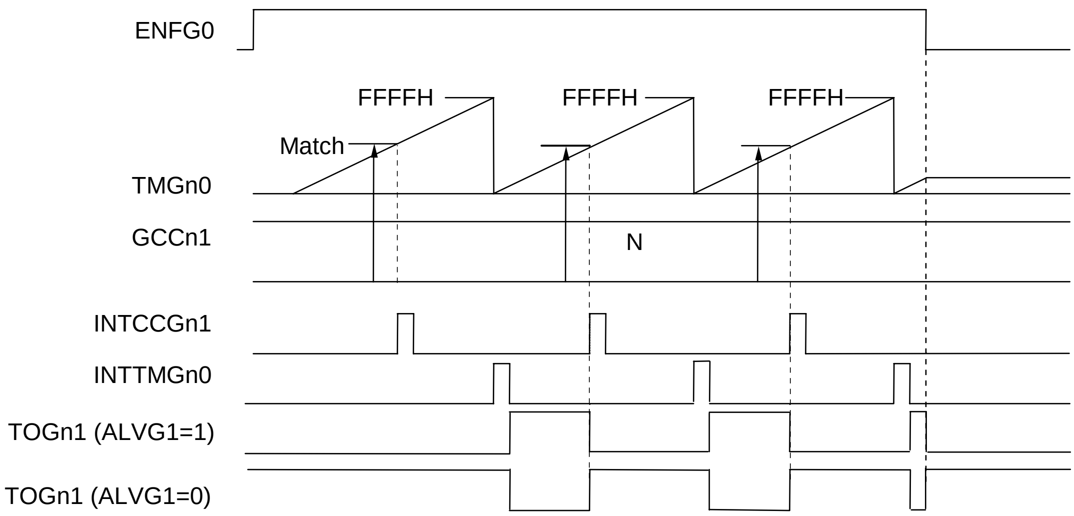
PWM setting method:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter must be selected with the TBGm bit.
(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1 register) or CSE02 to CSE00 bits (TMGn0 register).
(3) Specify the active level of a timer output (TOGnm pin) with the ALVGm bit.
(4) When using multiple timer outputs, the user can prevent TOGnm from becoming active simultaneously by setting the OLDE bit of TMGMHn register to provide step-by-step delays for TOGnm. (This capability is useful for reducing noise and current.)
(5) Write data to GCCnm.
(6) Start timer operation by setting POWER bit and TMG0E bit (or TMG1E bit).

PWM operation:

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.
(2) When the counter overflows, an overflow interrupt (INTTMGn0 or INTTMGn1) is generated.
(3) TOGnm does not make a transition until the first overflow occurs. (Even if the counter is cleared by software, TOGnm does not make a transition until the next overflow occurs. After the first overflow occurs, TOGnm is activated.
(4) When the value of the counter matches the value of GCCnm, TOGnm is deactivated, and a match interrupt (INTCCGnm) is output. The counter is not cleared, but continues count-up operation.
(5) The counter overflows, and INTTMGn0 or INTTMGn1 is output to activate TOGnm. The counter resumes count-up operation starting with 0000H.

Figure 10-46: Timing of PWM operation (free run)

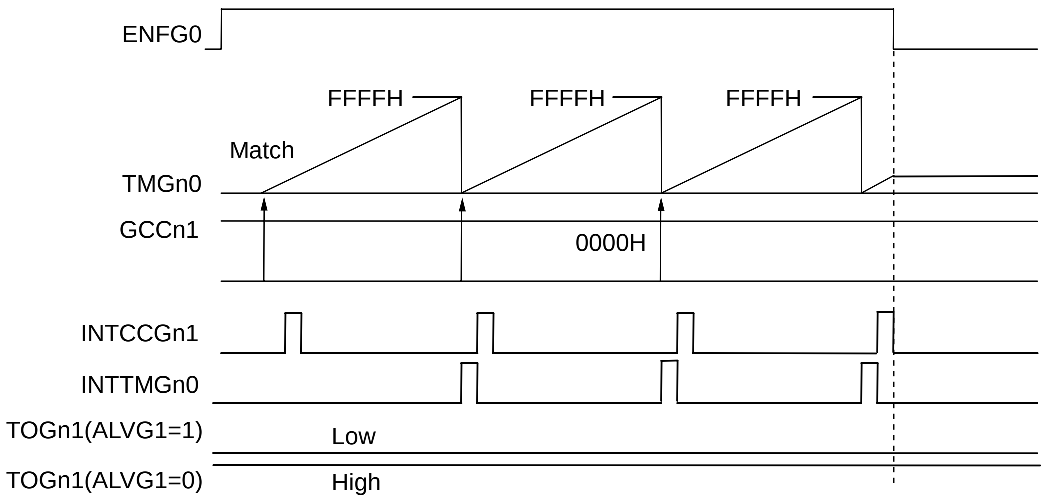


Data N is set in GCCn1, counter TMGn0 is selected.

(a) When 0000H is set in GCCnm (m = 1 to 4)

When 0000H is set in GCCnm, TOGnm is tied to the inactive level.
The figure below shows the state of TOGn1 when 0000H is set in GCCn1, and TMGn0 is selected.

Figure 10-47: Timing when 0000H is set in GCCnm (free run)



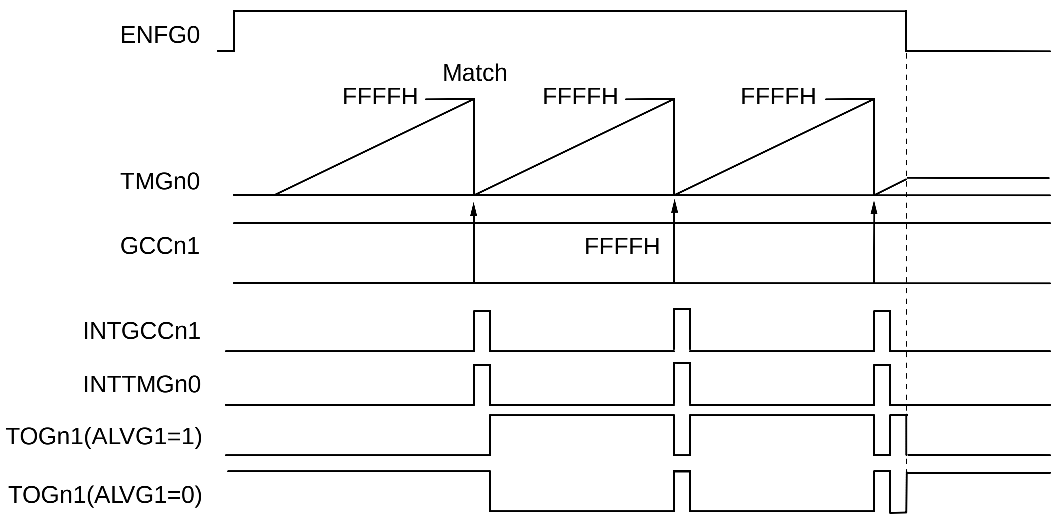
GCCn1 and TMGn0 are selected.

(b) When FFFFH is set in GCCnm (m = 1 to 4)

When FFFFH is set in GCCnm, TOGnm outputs the inactive level for one clock period immediately after each counter overflow (except the first overflow).

The figure shows the state of TOGn1 when FFFFH is set in GCCn1, and TMGn0 is selected.

Figure 10-48: Timing when FFFFH is set in GCCnm (free run)



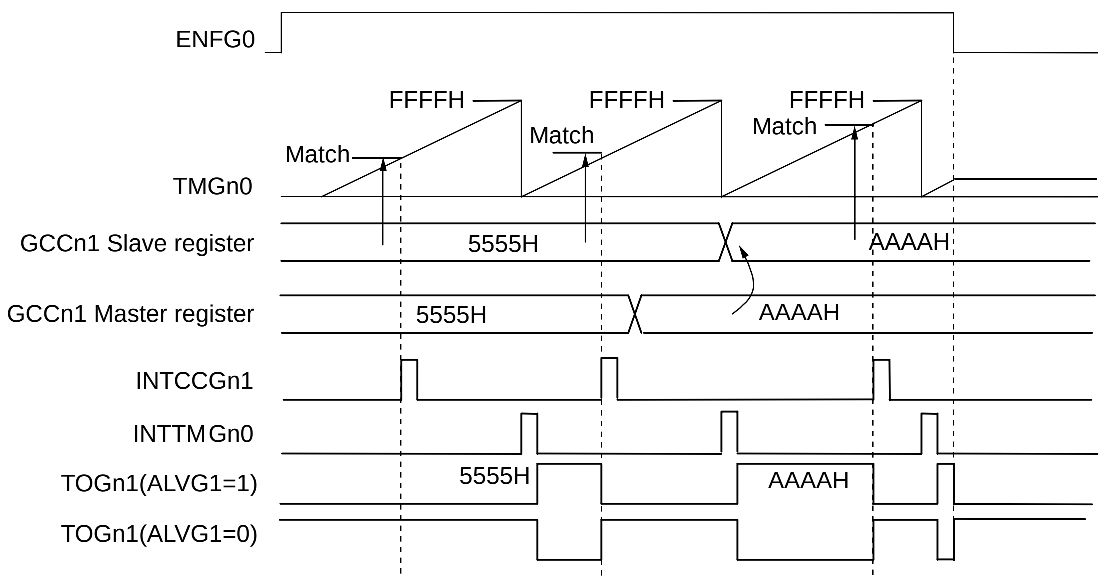
GCCn1 and TMGn0 are selected.

(c) When GCCnm is rewritten during operation (m = 1 to 4)

When GCCn1 is rewritten from 5555H to AAAAH, the operation shown below is performed.

The figure below shows a case where TMGn0 is selected for GCCn1.

Figure 10-49: Timing when GCCnm is rewritten during operation (free run)



GCCn1 and TMGn0 are selected.

If GCCn1 is rewritten to AAAAH after the second INTCCGn1 is generated as shown in the figure above, AAAAH is reloaded to the GCCn1 register when the next overflow occurs.
The next match interrupt (INTCCGn1) is generated when the value of the counter is AAAAH. The pulse width also matches accordingly.

10.3.8 Match and clear mode

The match and clear mode is mainly used reduce the number of valid bits of the counters (TMGn0, TMGn1).
Therefore the fixed assigned register GCCn0 (GCCn1) is used to compare its value with the counter TMGn0 (TMGn1). If the values match, than an interrupt is generated and the counter is cleared. Than the counter starts up counting again.

(1) Capture operation (match ad clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSG0n	1	match and clear mode
CCSG5n	1	
SWFGm	0	disable TOGnm
CCSGm	0	Capture mode for GCCnm
TBGm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Pulse width measurement or period measurement of the TIGm input signal

Setting method:

(1) When using one of TOGn1 to TOGn4-pin, select the corresponding counter with the TBGm bit. When CCSG0n = 1, TI0 cannot be used. When CCSG5n = 1, TIGn5 cannot be used.
(2) Select a count clock cycle with the CSE12 to CSE10 (TMGn1) bits or CSE02 to CSE00 (TMGn0) bits.
(3) Select a valid TIGm edge with the IEGm1 and IEGm0 bit. A rising edge, falling edge, or both edges can be selected.
(4) Set an upper limit on the value of the counter in GCCn0 or GCCn5.
(5) Start timer operation by setting POWER bit and TMG0E bit (or TMG1E bit).

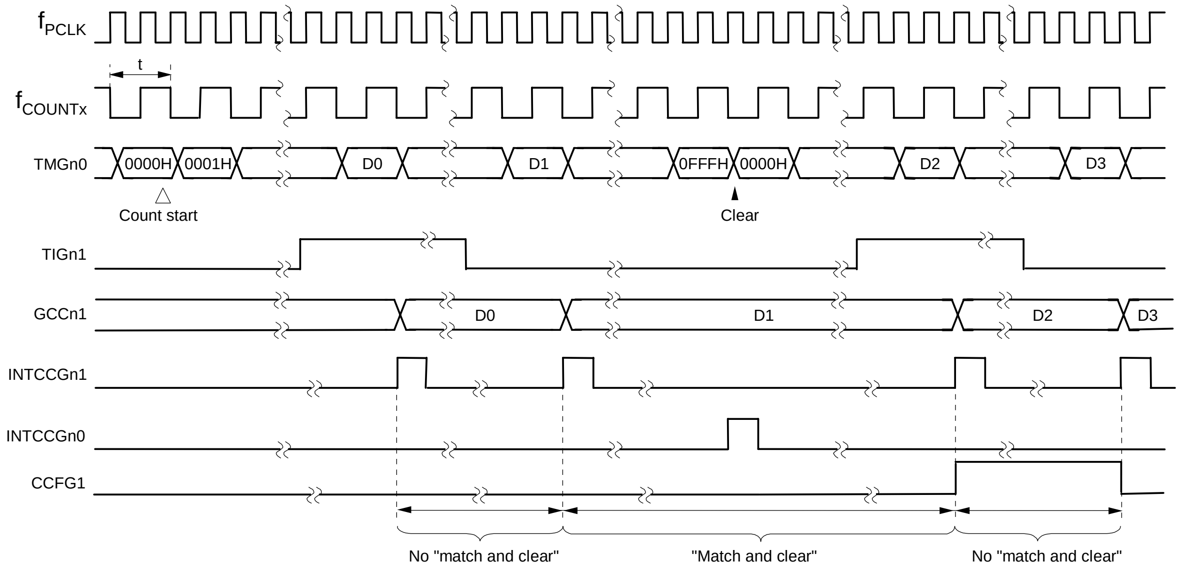
Operation:

(1) When a specified edge is detected, the value of the counter is stored in GCCnm, and an edge detection interrupt (INTCCGnm) is output.
(2) When the value of GCCn0 or GCCn5 matches the value of the counter, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".
(3) If a match and clear event has occurred between capture operations, the CCFGy flag is set when GCCny is read. Correct capture data by checking the value of CCFGy.

(b) Example: Capture where both edges of TIGm are valid (match and clear)

For the timing chart TMGn0 is selected as the counter corresponding to TOGn1, and 0FFFH is set in GCCn0.

Figure 10-50: Timing when both edges of TIGm are valid (match and clear)



Remark: The figure above shows an image. In actual circuitry, 3 to 4 periods of the count-up signal (f_{COUNT}) are required from the input of a waveform to TOGn1 until a capture interrupt is output. (See Figure 10-42, “Timing of capture trigger edge detection (free run),” on page 327.)

Caution: If two or more match and clear events occur between captures, a software-based measure needs to be taken to count INTCCGn0 or INTCCGn5.

(c) When 0000H is set in GCCn0 or GCCn5 (match and clear)

When 0000H is set in GCCn0 (GCCn5), the value of the counter is fixed at 0000H, and does not operate. Moreover, INTCCGn0 (INTCCGn5) continues to be active.

(d) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 (GCCn5), operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (INTTMGn1) is generated, but INTCCGn0 (INTCCGn5) is not generated.

(2) Compare operation (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSG0n	1	match and clear mode
CCSG5n	1	
SWFGm	0	disable TOGnm
CCSGm	1	Compare mode for GCCnm
TBGm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Interval timer (match and clear)

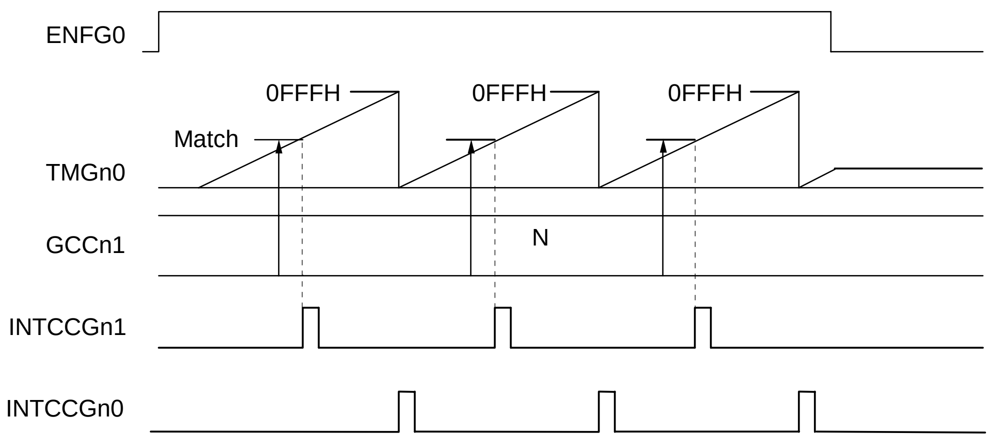
Setting Method

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter must be selected with the TBGm bit.
(2) Select a count clock cycle with the CSE12 to CSE10 bits (TMGn1) or CSE02 to CSE00 bits (TMGn0).
(3) Set an upper limit on the value of the counter in GCCn0 or GCCn5.
(4) Write data to GCCnm.
(5) Start timer operation by setting the POWER bit and TMGxE bit (x = 0, 1).

Operation:

(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.
(2) When the value of GCCn0 or GCCn5 matches the value of the counter, INTCCGn0 (or INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".
(3) The counter resumes count-up operation starting with 0000H.

Figure 10-51: Timing of compare operation (match and clear)



In this example, the data N is set in GCCn1, and TMGn0 is selected.
0FFFH is set in GCCn0. Here, N < 0FFFH.

(b) When 0000H is set in GCCn0 or GCCn5 (match and clear)

When 0000H is set in GCCn0 or GCCn5, the value of the counter is fixed at 0000H, and does not operate. Moreover, INTCCGn0 (or INTCCGn5) continues to be active.

(c) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 or GCCn5, operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (or INTTMGn1) is generated, but INTCCGn0 (or INTCCGn5) is not generated.

(d) When 0000H is set in GCCnm (m = 1 to 4) (match and clear)

INTCCGnm is activated when the value of the counter becomes 0001H.
Note, however, that even if no data is set in GCCnm, INTCCGnm is activated immediately after the counter starts.

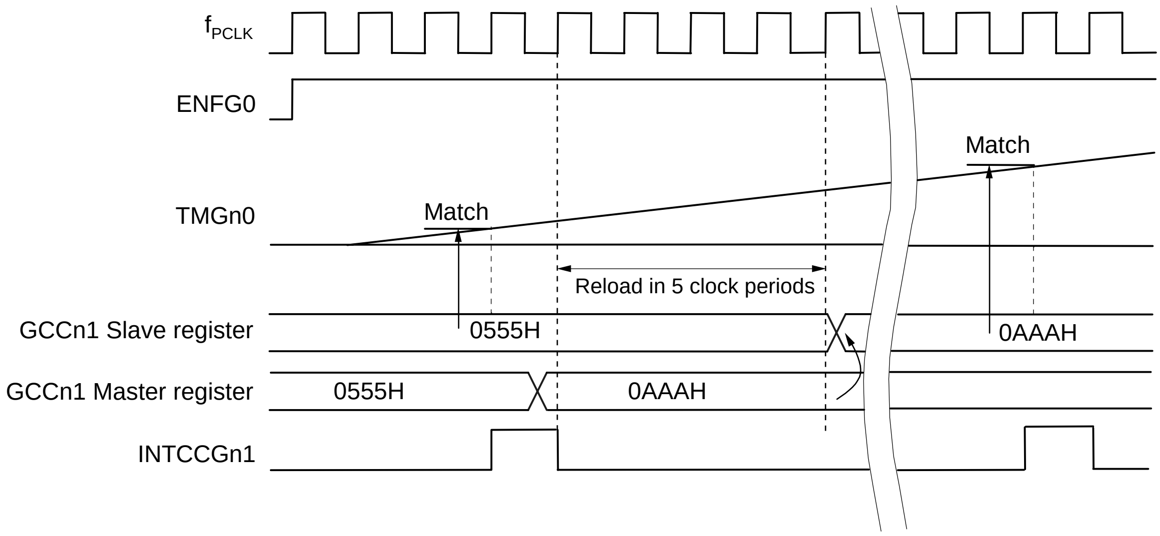
(e) When a value exceeding the value of GCCn0 or GCCn5 is set in GCCnm (m = 1 to 4) (match and clear)

INTCCGnm is not generated.

(f) When GCCnm (m = 1 to 4) is rewritten during operation (match and clear)

When the value of GCCn1 is changed from 0555H to 0AAAH, the operation described below is performed.
TMGn0 is selected as the counter, and 0FFFH is set in GCCn0.

Figure 10-52: Timing when GCCnm is rewritten during operation (match and clear)



Caution: To perform successive write access during operation, for rewriting the GCCny register (n = 1 to 4), you have to wait for minimum 7 peripheral clocks periods (f_{PCLK}).

(3) PMW output (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSG0	1	match and clear mode
CCSG5	1	
SWFGm	1[Note]	enable TOGnm
CCSGm	1[Note]	Compare mode for GCCnm
TBGm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

Note: The PWM mode is activated by setting the SWFGm and the CCSGm bit to "1".

Setting Method:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counters TMGn0 or TMGn1 must be selected with the TBGm bit (m = 1 to 4).
(2) Select a count clock cycle with the CSE12 to CSE10 (TMGn1) bits or CSE02 to CSE00 (TMGn0) bits.
(3) Specify the active level of a timer output (TOGnm) with the ALVGm bit.
(4) When using multiple timer outputs, the user can prevent TOGnm from making transitions simultaneously by setting the OLDE bit of TMGMHn register. (This capability is useful for reducing noise and current.)
(5) Set an upper limit on the value of the counter in GCCn0 or GCCn5. (Timer Dn 0000H is forbidden)
(6) Write data to GCCnm.
(7) Start count operation by setting POWER bit and TMG0E bit (or TMG1E bit).

Operation of PWM (match and clear):

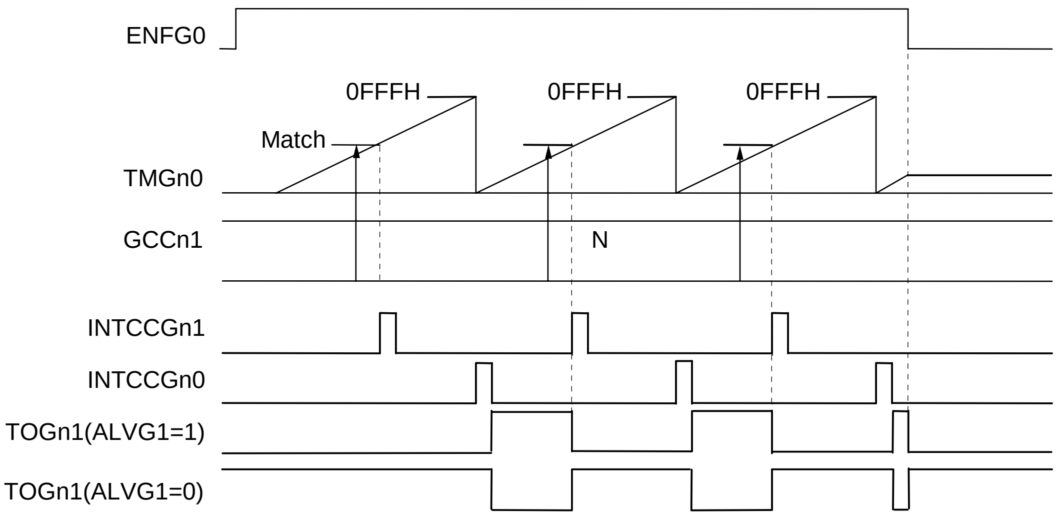
(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

Caution: Do not set 0000H in GCCn0 or GCCn5 in match and clear modus.

(2) When the value of GCCn0 (GCCn5) matches the value of the counter, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. This operation is referred to as "match and clear".
(3) TOGnm does not make a transition until the first match and clear event.
(4) TOGnm makes a transition to the active level after the first match and clear event.
(5) When the value of the counter matches the value of GCCnm, TOGnm makes a transition to the inactive level, and a match interrupt (INTCCGnm) is output.
(6) When the next match and clear event occurs, INTCCGn0 (INTCCGn5) is output, and the counter is cleared. The counter resumes count-up operation starting with 0000H.

Example where the data N is set, and the counter TMGn0 is selected.
0FFFH is set in GCCn0 and N < 0FFFH.

Figure 10-53: Timing of PWM operation (match and clear)



When 0000H is set in GCCn0 (GCCn5), the value of the counter is fixed at 0000H, and the counter does not operate. The waveform of INTCCGn0 (INTCCGn5) varies, depending on whether the count clock is the reference clock or the sampling clock.

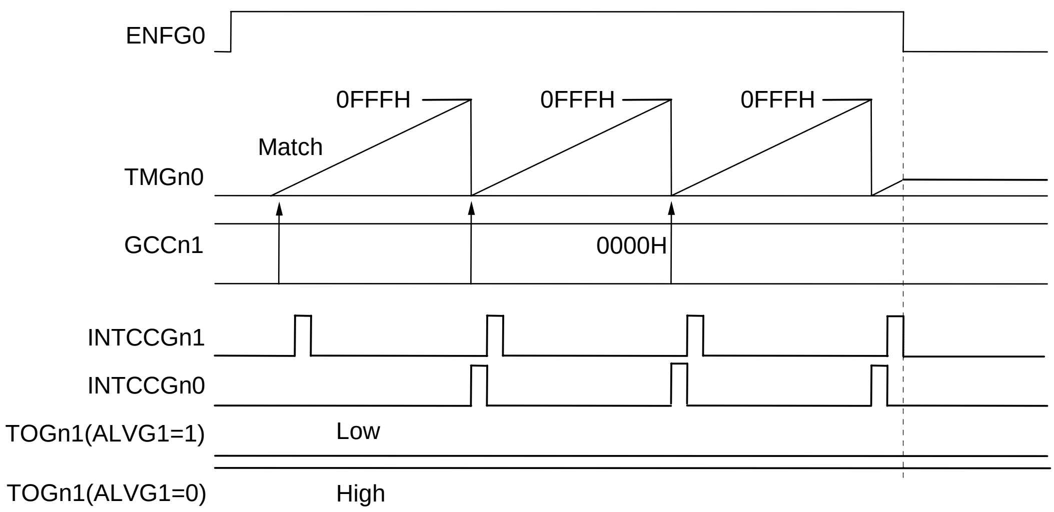
(a) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

When FFFFH is set in GCCn0 (GCCn5), operation equivalent to the free-run mode is performed. When an overflow occurs, INTTMGn0 (INTTMGn1) is generated, but INTCCGn0 (INTCCGn5) is not generated.

(b) When 0000H is set in GCCnm (match and clear)

When 0000H is set in GCCnm, TOGnm is tied to the inactive level.
The figure below shows the state of TOGn1 when 0000H is set in GCCn1, and TMGn0 is selected. Note, however, that 0FFFH is set in GCCn0.

Figure 10-54: Timing when 0000H is set in GCCnm (match and clear)

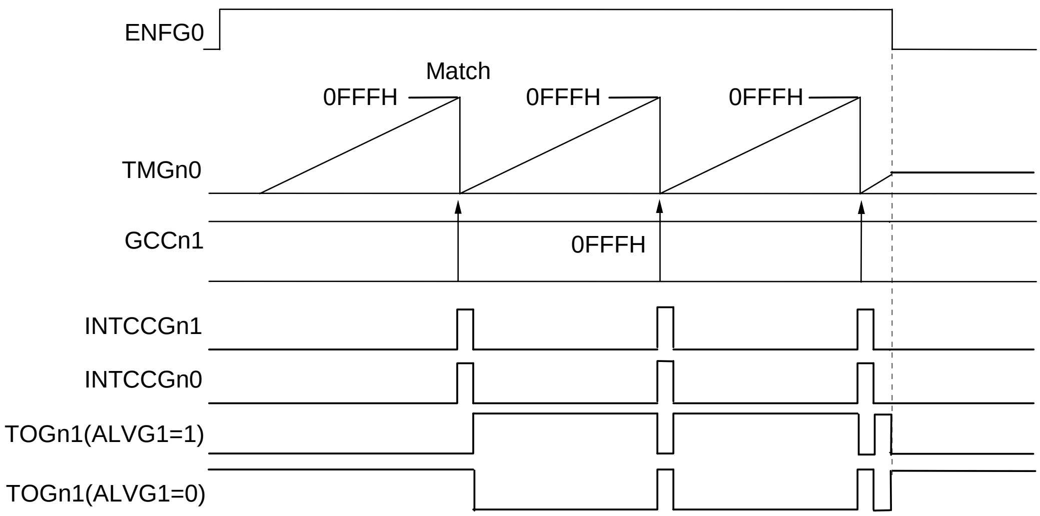


(c) When the same value as set in GCCn0 or GCCn5 is set in GCCnm (match and clear)

When the same value as set in GCCn0 (GCCn5) is set in GCCnm, TOGnm outputs the inactive level for only one clock period immediately after each match and clear event (excluding the first match and clear event).

The figure below shows the state of TOGn1 when 0FFFH is set in GCCn0 and GCCn1, and TMGn0 is selected.

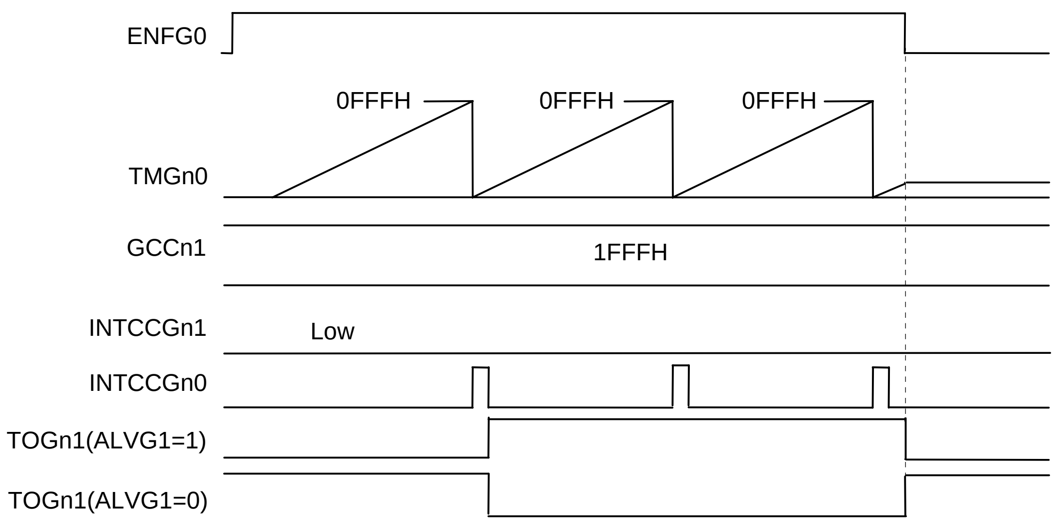
Figure 10-55: Timing when the same value as set in GCCn0/GCCn5 is set in GCCnm (match and clear)



(d) When a value exceeding the value set in GCCn0 or GCCn5 is set in GCCnm (match and clear)

When a value exceeding the value set in GCCn0 (GCCn5) is set in GCCnm, TOGnm starts and continues outputting the active level immediately after the first match and clear event (until count operation stops.)
The figure shows the state of TOGn1 when 0FFFH is set in GCCn0, 1FFFH is set in GCCn1, and TMGn0 is selected.

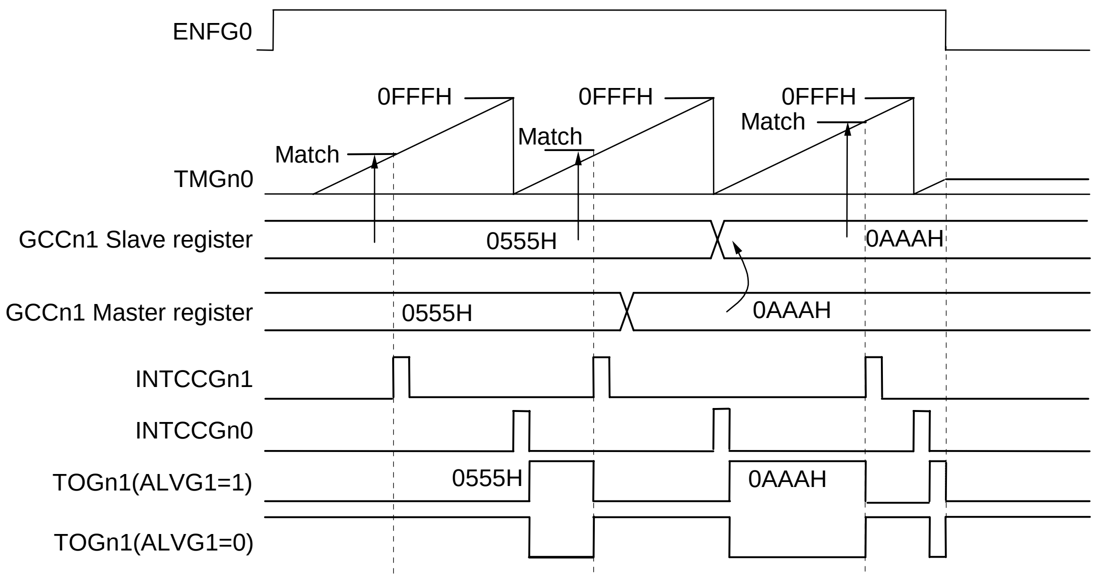
Figure 10-56: Timing when the value of GCCnm exceeding GCCn0 or GCCn5 (match and clear)



(e) When GCCnm is rewritten during operation (match and clear)

When GCCn1 is rewritten from 0555H to 0AAAH, the operation shown below is performed.
The figure below shows a case where 0FFFH is set in GCCn0, and TMGn0 is selected for GCCn1.

Figure 10-57: Timing when GCCnm is rewritten during operation (match and clear)



If GCCn1 is rewritten to 0AAAH after the second INTCCGn1 is generated as shown in the figure above, 0AAAH is reloaded to the GCCn1 register when the next overflow occurs.
The next match interrupt (INTCCGn1) is generated when the value of the counter is 0AAAH. The pulse width also matches accordingly.

10.3.9 Edge noise elimination

The edge detection circuit has a noise elimination function. This function regards:

- a pulse **not wider than 1 count clock** period as a **noise**, and does not detect it as an edge.
- a pulse **not shorter than 2 count clock** periods is detected normally as an **edge**.
- a pulse **wider than 1 count clock period but shorter than 2 count clock** periods may be **detected as an edge or may be eliminated as noise**, depending on the timing.

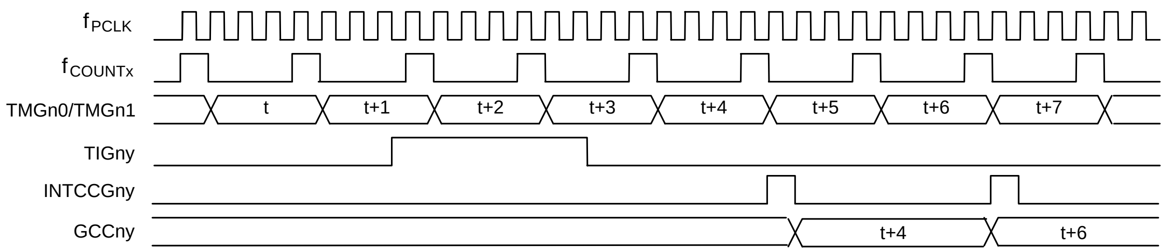
(This is because the count-up signal of the counter is used for sampling timing.) The upper figure below shows the timing chart for performing edge detection. The lower figure below shows the timing chart for not performing edge detection.

Basic settings (x = 0, 1 and y = 0 to 5):

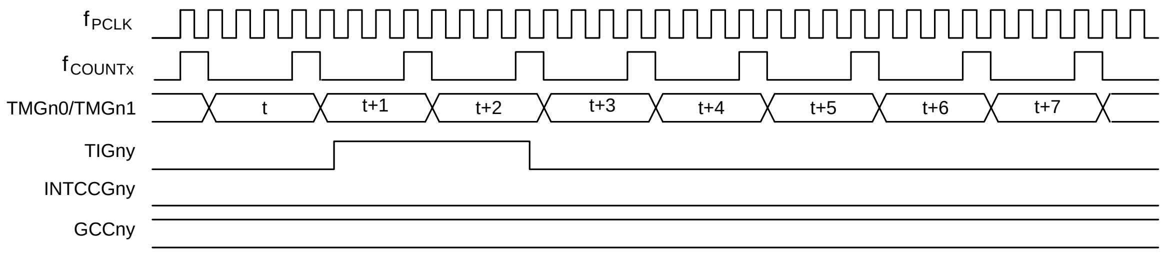
Bit	Value	Remark
CSEx2	0	Count clock = $f_{PCLK}/4$
CSEx1	1	
CSEx0	0	
IEGy1	1	detection of both edges
IEGy0	1	

Figure 10-58: Timing of Edge detection noise elimination

<Timing chart for performing edge detection>



<Timing chart for noise elimination>



10.3.10 Precautions Timer Gn

(1) When POWER bit of TMGMHn register is set

The rewriting of the CSEn2 to CSEn0 bits (n = 0, 1) of TMGMHn register is prohibited.
These bits set the prescaler for the Timer Gn counter.

The rewriting of the CCSGy bits (y = 0 to 5) is prohibited.
This bits (OCTLGnL and OCTLGnH registers) set the capture mode or the compare mode to the GCCy register. For the GCCn0 register and the GCCn5 register these bits (TMGMLn register) set the “free run” or “match and clear” mode of the TMGn0 and TMGn1 counter.

The rewriting of the TMGCMnL and the TMGCMHn register is prohibited.
These registers configure the counter (TMGn0 or TMGn1) for the GCCnm register (m = 1 to 4) and define the edge detection for the TIGm input pins (falling, rising, both).

Even when POWER bit is set, TOGnm output is switched by switching the ALVGm bit of OCTLGnL and OCTLGnH registers.
These bits configure the active level of the TOGnm-pins (m = 1 to 4).

(2) When POWER bit and TMGxE bit are set (x = 0, 1)

The rewriting of ALVGm is prohibited (m = 1 to 4).
These bits configure the active level of the TOGnm-pins (m = 1 to 4).

When in compare-mode the rewriting of the GCCn0 or GCCn5 register is prohibited.
In compare mode these registers set the value for the “match and clear” mode of the TMGn0 and TMGn1 counter.

(3) Functionality

When the POWER bit is set to “0”, regardless of the SWFGm bit (OCTLGnL and OCTLGnH registers), the TOGnm pins are tied to the inactive level.
The SWFGm bit enables or disables the output of the TOGnm pins. This bit can be rewritten during timer operation.

The CLRGx bit (x = 0, 1) is a flag. If this bit is read, a "0" is read at all times.
This bit clears the corresponding counter (TMGn0 or TMGn1)

When GCCnm register (m = 1 to 4) are used in capture operation:
If two or more overflows of TMGn0 or TMGn1 occur between captures, a software-based measure needs to be taken to count overflow interrupts (INTTMGn0 or INTTMGn1).
If only one overflow is necessary, the CCFGy bits (y = 0 to 5) can be used for overflow detection.

Only the overflow of the TMGn0 or TMGn1counter clears the CCFGy bit (TMGSTn register). The software-based clearing via CLRG0 or CLRG1 bit (TMGMLn register) doesn’t affect these bits.
The CCFGy bit is set if a TMGn0 (TMGn1) overflow occurs. This flag is only updated if the corresponding GCCny register was read, so first read the GCCny register and then read this flag if necessary.

(4) Timing

The delay of each timer output TOGnm (m = 1 to 4) varies according to the setting of the count clock with the CSEx2 to CSEx0 bits (x = 0, 1).

In capture operation 3 to 4 periods of the count-clock (f_{COUNT}) signal are required from the TIGny pin (y = 0 to 5) until a capture interrupt is output.

when TMGxE (x = 0, 1) is set earlier or simultaneously with POWER bit, than the Timer Gn needs 7 peripheral clocks periods (f_{PCLK}) to start counting.
when TMGxE (x = 0, 1) is set later than POWER bit, than the Timer Gn needs 4 peripheral clocks periods (f_{PCLK}) to start counting.

When a capture register (GCCny) is read, the capturing is disable during read operation. This is intended to prevent undefined data during reading. So, if a contention occurs between an external trigger signal and the read operation, capture operation may be cancelled, and old data may be read.

GCCnm register (m = 1 to 4) in Compare mode:
After setting the POWER bit you have to wait for 10 peripheral clocks periods (f_{PCLK}) to perform write access to the GCCnm register (m = 1 to 4).
To perform successive write access during operation, for rewriting the GCCnm register (n = 1 to 4), you have to wait for minimum 7f peripheral clocks periods (f_{PCLK}).

Chapter 11 Watch Timer

11.1 Function

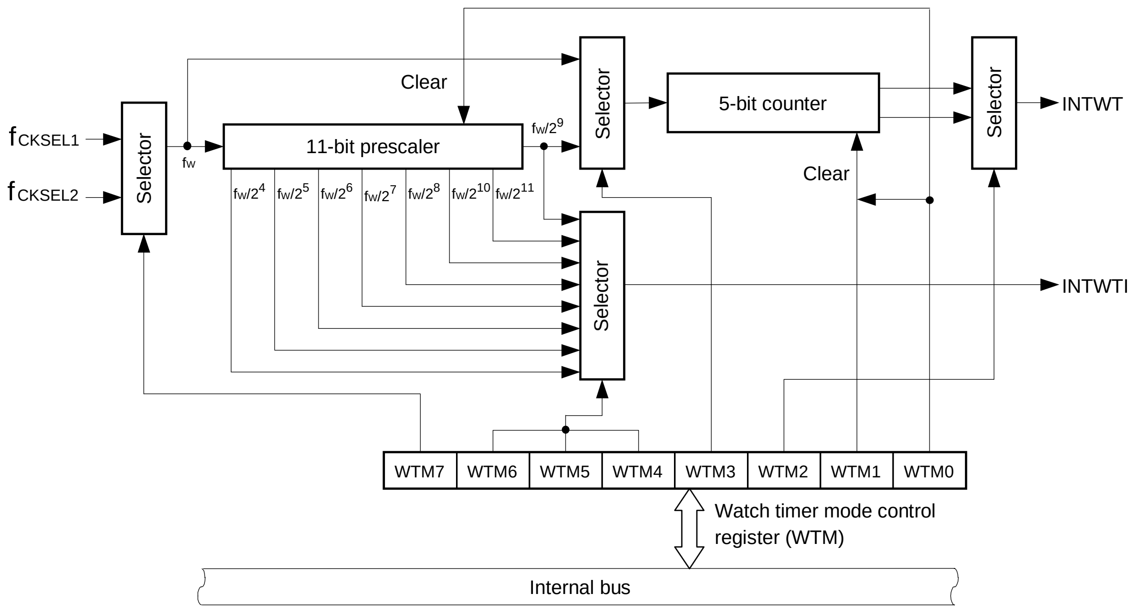
The watch timer has the following functions:

- Watch timer
- Interval timer

The watch timer and interval timer functions can be used at the same time.

Figure 11-1 shows the block diagram of the watch timer.

Figure 11-1: Block Diagram of Watch Timer



(1) Watch timer

The watch timer generates an interrupt request (INTWT) at time intervals of 500 µs to 16.4 s by using the clock selector for the Watch Timer (see Chapter **8.2 "Configuration" on page 198**).

(2) Interval timer

The interval timer generates an interrupt request (INTWTI) at time intervals of 500 µs to 2.1 s.

11.2 Configuration

The watch timer consists of the following hardware:

Table 11-1: Configuration of Watch Timer

Item	Configuration
Counter	5 bits × 1
Prescaler	11 bits × 1
Control register	Watch timer mode control register (WTM)

11.3 Watch Timer Control Register

The watch timer mode control register (WTM) controls the watch timer.

(1) Watch timer mode control register (WTM)

This register enables or disables the count clock and operation of the watch timer, sets the interval time of the prescaler, controls the operation of the 5-bit counter, and sets the set time of the watch flag.

WTM is set by a 1-bit or 8-bit memory manipulation instruction.

Figure 11-2: Watch Timer Mode Control Register (WTM) (1/2)

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
WTM	WTM7	WTM6	WTM5	WTM4	WTM3	WTM2	WTM1	WTM0	FFFF F560H	R/W	00H

WTM7	Selects main input frequency from prescaler
0	Clock input f_{CKSEL1} is selected
1	Clock input f_{CKSEL2} is selected

WTM6	WTM5	WTM4	Selects Interval Time of Prescaler (f_X = 4 MHz)
0	0	0	$2^4/f_W$
0	0	1	$2^5/f_W$
0	1	0	$2^6/f_W$
0	1	1	$2^7/f_W$
1	0	0	$2^8/f_W$
1	0	1	$2^9/f_W$
1	1	0	$2^{10}/f_W$
1	1	1	$2^{11}/f_W$

Figure 11-2: Watch Timer Mode Control Register (WTM) (2/2)

WTM3	WTM2	Selects Set Time of Watch Flag
0	0	$2^{14}/f_W$
0	1	$2^{13}/f_W$
1	0	$2^5/f_W$
1	1	$2^4/f_W$

WTM1	Controls Operation of 5-bit Counter
0	Clears after operation stops
1	Starts

WTM0	Enables Operation of Watch Timer
0	Stops operation (clears both prescaler and timer)
1	Enables operation

Remark: f_W: Watch timer clock frequency

11.4 Operations

11.4.1 Selection of the Watch Timer Clock

With the settings of the clock generator different clocks can be assigned to the Watch Timer. With the WTSELn bits (n = 0, 1) of the CKC register 6 different clocks can be switched as the Watch Timer clock.

Table 11-2: Selection of the Watch Timer Clock

CKC Register			WTM Register	f_W	
WDTSEL1	f_{XXT}	WDTSEL0	WTM7	f_X = 4 MHz	f_X = 5 MHz
0	f_X/128	0	1	31250 Hz	39063 Hz
		1		7813 Hz	9766 Hz
		X	0	977 Hz	1221 Hz
1	f_{XT}	0	1	32000 Hz	
		1		8000 Hz	
		X	0	1000 Hz	

Note: WTSEL1 bit of CKC register

Remark: X = don't care

11.4.2 Control of the watch timer

The watch timer operates with time intervals from 500 µs to 16.4 s.
The watch timer generates at its overflow the INTWT interrupt request at fixed time intervals.

With the WTM1 bit and the WTM0 bit the watch timer function can be started.
With the WTM1 bit the watch timer function can be stopped independently from the interval timer function.

For synchronous watch and interval timer function operation:
The count operation of the watch timer is started when the WTM0 bit and the WTM1 bit of the watch timer mode control register (WTM) are set to “1”.
Both prescalers are stopped and cleared if the WTM0 bit is set to “0“.

For independent start or stop of watch timer function operation:
This functionality is only available, when the 11-bit Prescaler is running, too.
The count operation of the watch timer is started when the WTM1 bit and the WTM0 bit of the watch timer mode control register (WTM) are set to “1”.
The WTM0 bit has to be set to “1” either it was “1” before. In that case the frequency of the running 11-bit prescaler is not influenced.
The 5-bit watch timer function prescaler is stopped and cleared if the WTM1 bit is set to “0“.

Caution: If the 5-bit watch timer function prescaler is clocked by $f_W/2^9$ (WTM3 = 0).
This prescaler is started with the next edge of the $f_W/2^9$ clock. Therefore if the 11-bit prescaler was running before the 5-bit prescaler watch timer is started, the INTWT interrupt is generated up to one $f_W/2^9$ period later then the time that the WTM1 bit was set to “1”.

For example: if $f_W = f_{XXT}/32$ (i.e. f_{XXT} = 32000 Hz) this can be up to 1 ms.
This happens only at the first starting edge of the $f_W/2^9$ clock.

Table 11-3: Example for Interval Time of Watch Timer

WTM3	WTM2	Interval Time	$f_W = f_{CKSEL2} = f_X/4096$	$f_W = f_{CKSEL1} = f_{XT}$ (32 KHz)
			WTSEL1 = 0	WTSEL1 = 1
1	1	$2^4 \times 1/f_W$	16.4 ms	0.5 ms
1	0	$2^5 \times 1/f_W$	32.8 ms	1.0 ms
0	1	$2^{13} \times 1/f_W$	8.4 s	256 ms
0	0	$2^{14} \times 1/f_W$	16.8 s	512 ms

Remarks:
1. f_X = 4 MHz
2. WTSEL0 bit = 1(CKC register) → $f_{CKSEL2} = f_{XXT}/32$
3. f_W: Watch timer clock frequency
4. interval times change accordingly if f_X = 4 MHz

11.4.3 Operation as interval timer

The watch timer can also be used as an interval timer that repeatedly generates an interrupt at intervals specified by a count value set in advance.
The Interval Timer and the Watch Timer can used at the same time. The interval time of the interval timer is smaller than the interval time of the Watch Timer, every time. It is a sub interval of the Watch Timer.
The interval time can be selected by the WTM4 through WTM6 bits of the watch timer mode control register (WTM).

Table 11-4: Example for Interval Time of Interval Timer

WTM6	WTM5	WTM4	Interval Time	$f_W = f_{CKSEL2} = f_X/4096$	$f_W = f_{CKSEL1} = f_{XT}$ (32 KHz)
				WTSEL1 = 0	WTSEL1 = 1
0	0	0	$2^4 \times 1/f_W$	16.4 ms	0.5 ms
0	0	1	$2^5 \times 1/f_W$	32.8 ms	1 ms
0	1	0	$2^6 \times 1/f_W$	65.5 ms	2 ms
0	1	1	$2^7 \times 1/f_W$	131 ms	4 ms
1	0	0	$2^8 \times 1/f_W$	262 ms	8 ms
1	0	1	$2^9 \times 1/f_W$	524 ms	16 ms
1	1	0	$2^{10} \times 1/f_W$	1048 ms	32 ms
1	1	1	$2^{11} \times 1/f_W$	2096 ms	65 ms

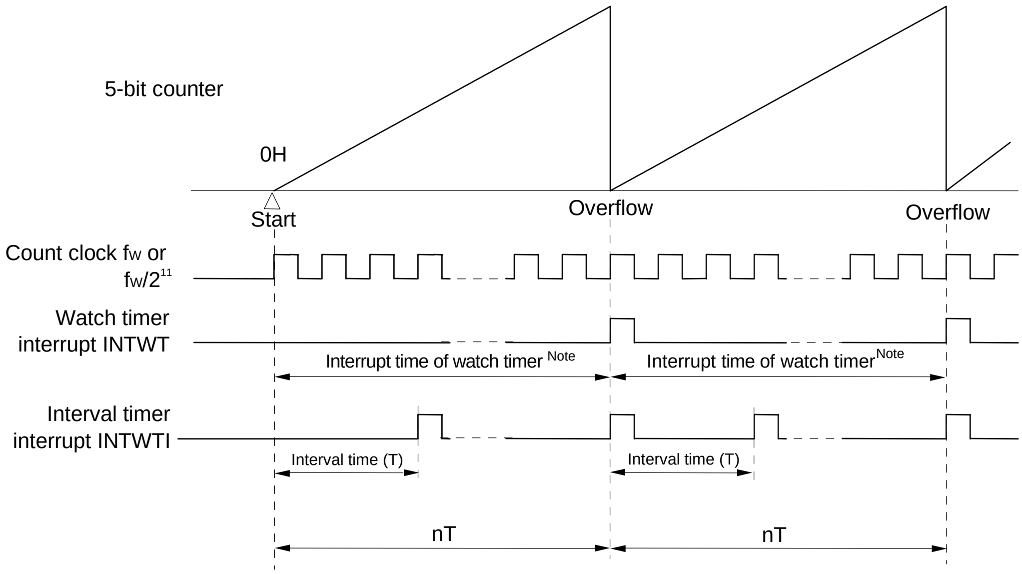
Remarks: **1.** f_X = 4 MHz

2. WTSEL0 bit = 1(CKC register) $\rightarrow f_{CKSEL2} = f_{XXT}/32$

3. f_W: Watch timer clock frequency

4. interval times change accordingly if f_X = 4 MHz

Figure 11-3: Operation Timing of Watch Timer/Interval Timer



Note: The Watch Timer frequency depends of the CKC register (Clock Generator) and the WTM (Watch Timer) register.

Remarks: 1. f_W : Watch timer clock frequency

2. n : Interval timer operation numbers

[MEMO]

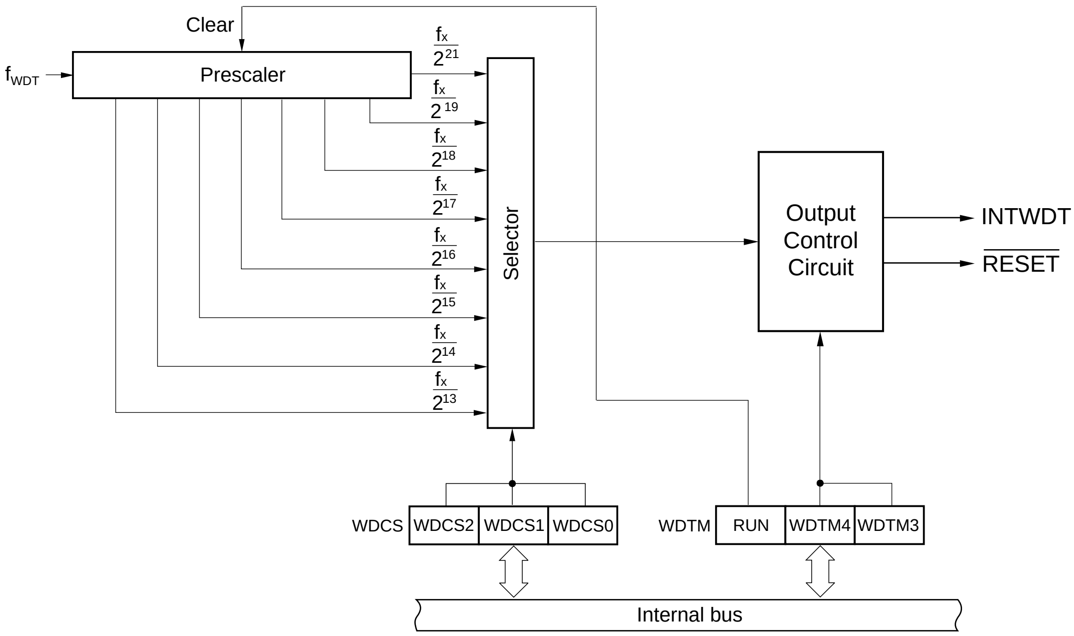
Chapter 12 Watchdog Timer Function

12.1 Functions

The watchdog timer has the following functions.

- Watchdog Timer with non maskable interrupt INTWDT
- Watchdog Timer with hardware $\overline{RESET}$.

Figure 12-1: Block Diagram of Watchdog Timer



Remark: f_{WD}: Watchdog Timer clock frequency (depends on clock tree settings)

(1) Interrupt mode

This mode detects program runaway. When runaway is detected, a non-maskable interrupt can be generated.

(2) $\overline{\text{RESET}}$ mode

This mode detects program runaway. When runaway is detected, a hardware $\overline{RESET}$ is generated.

12.2 Configuration

The watchdog timer consists of the following hardware.

Table 12-1: Watchdog Timer Configuration

Item	Configuration
Control registers	Watchdog timer clock selection register (WDCS) Watchdog timer mode register (WDTM) Watch Dog Timer command register (WCMD) Watch Dog Timer command status register (WPHS)

12.3 Watchdog Timer Control Register

The registers to control the watchdog timer is shown below.

- Watchdog timer clock selection register (WDCS)
- Watchdog timer mode register (WDTM)

(1) Watchdog timer clock selection register (WDCS)

This register selects the overflow times of the watchdog timer.

WDCS is set by an 8-bit memory manipulation instruction.

Figure 12-2: Watchdog Timer Clock Selection Register (WDCS)

	7	6	5	4	3	2	1	0	Address	R/W	After reset
WDCS	0	0	0	0	0	WDCS2	WDCS1	WDCS0	FFFF F571H	R/W	00H

WDCS2	WDCS1	WDCS0	Clock	Overflow Time[Note]	
				$f_{WD} = f_X$ 4 MHz (main clock)	$f_{WD} = f_{XT}$ 32 KHz (sub clock)
0	0	0	$2^{13}/f_{WD}$	2 ms	256 ms
0	0	1	$2^{14}/f_{WD}$	4.1 ms	512 ms
0	1	0	$2^{15}/f_{WD}$	8.2 ms	1.02 s
0	1	1	$2^{16}/f_{WD}$	16.4 ms	2.05 s
1	0	0	$2^{17}/f_{WD}$	32.8 ms	4.10 s
1	0	1	$2^{18}/f_{WD}$	65.5 ms	8.20 s
1	1	0	$2^{19}/f_{WD}$	131 ms	16.38 s
1	1	1	$2^{21}/f_{WD}$	524 ms	65.54 s

Note: This are only 2 examples for f_{WD}. The clock depends on the clock tree settings and the external main oscillator resonators (4 or 5 MHz).

(2) Watchdog timer mode register (WDTM)

This register sets the operating mode of the watchdog timer, and enables and disables counting. Data can be written to it only in a sequence of specific instructions so that its contents are not easily rewritten in case of program hang-up. See also WCMD register.

Once the watchdog timer is started (RUN = 1) after reset, this register cannot be changed.

WDTM is set by an 8/1-bit memory manipulation instruction.

Cautions: 1. The WDTM4 bit has to set to “1” at the initialisation of the WDT

2. WDTM register setting by DMA transfer is prohibited. This registers should be written with STORE instruction execution by CPU only.

Figure 12-3: Watchdog Timer Mode Register (WDTM)

	7	6	5	4	3	2	1	0	Address	R/W	After reset
WDTM	RUN	0	0	WDTM4	WDTM3	0	0	0	FFFF F572H	R/W	00H

RUN	Operating Mode Selection for the Watchdog Timer Note 1
0	After Reset: disable count (writing during counting: clear old count)
1	After Reset: start counting (writing during counting: clear old count)

WDTM4 Note 2	WDTM3	Watchdog Timer Operation Mode Selection
1	0	Watchdog timer mode 1 (Non-maskable interrupt occurs upon generation of an overflow)
1	1	Watchdog timer mode 2 ($\overline{\text{RESET}}$ operation is activated upon generation of an overflow)

Notes: 1. If RUN is set to “1” once, the register cannot be cleared to “0” by software. Therefore, when the count starts, the count cannot be stopped except by $\overline{\text{RESET}}$ input.

2. The WDTM4 bit has to set to “1” at the initialisation of the WDT.

Cautions: 1. Data is set to the CKC register by the following sequence:

- **Write any 8-bit data to the command register (WCMD)**
- **Write the set data to the destination register (WDTM)**

2. If RUN is set to “1” and the watchdog timer is cleared, the actual overflow time may be up to $2^{12}/f_{XX}$ seconds less than the set time.

(3) Watchdog timer command register (WCMD)

This command register WCMD is used to protect the WDTM register from unintended writing. Writing to WDTM register is possible only immediately after writing to WCMD register. Data written into WCMD register are ignored. Data read from WCMD register are undefined, too.

WDTM is set by an 8-bit memory manipulation instruction.

Caution: WCMD register setting by DMA transfer is prohibited. This registers should be written with STORE instruction execution by CPU only.

Figure 12-4: Watchdog Timer Mode Register (WCMD)

	7	6	5	4	3	2	1	0	Address	R/W	After reset
WCMD	X	X	X	X	X	X	X	X	FFFF F580H	R/W	undefined

(4) Watchdog timer command status register (WPHS)

The WPHS register monitors the success of a write instruction to the WDTM register. If the write to WDTM fails because of violating the special instruction sequence writing WCMD immediately before WDTM, the WPRERR flag is set.

WPHS can be accessed by 8-bit or 1-bit memory instructions.

Caution: The WPERR bit can only be reset by software. Setting the WPERR by software is not possible.

Figure 12-5: Watchdog Timer Mode Register (WPHS)

	7	6	5	4	3	2	1	0	Address	R/W	After reset
WPHS	0	0	0	0	0	0	0	WPRERR	FFFF F582H	R/W	00H

WPRERR	WDTM register Protection Error Flag
0	no WDTM register writing error has occurred
1	an WDTM register writing error has occurred

12.4 Operation

12.4.1 Operating as watchdog timer

Once the watchdog timer is started (RUN = 1) after reset, the RUN, WDTM4, and WDTM3 bits cannot be changed. These bits can be cleared only by reset input.

(1) Watchdog Timer Mode 1 (Interrupt)

Set WDTM4 bit of the watchdog timer mode register (WDTM) to “1” and WDTM3 bit to “0” to operate as a watchdog timer in interrupt-request-mode to detect program runaway.
Setting RUN bit of WDTM register to “1” starts the count. After counting starts, if RUN bit is set to “1” again within the set time interval for runaway detection, the watchdog timer is cleared and counting starts again.
If RUN is not set to “1” and the runaway detection time has elapsed, a non-maskable interrupt (INTWDT) is generated (no reset functions).
The watchdog timer stops running in the STOP mode. Consequently, set RUN to “1” and clear the watchdog timer before entering the STOP mode. Do not set the watchdog timer when operating the HALT mode since the watchdog timer is running in HALT mode.

For details of the possible time settings please refer to Figure 12-2, “Watchdog Timer Clock Selection Register (WDCS),” on page 359.

(2) Watchdog Timer Mode 2 ($\overline{\text{RESET}}$)

Set WDTM4 bit and WDTM3 bit of the watchdog timer mode register (WDTM) to “1” to operate as a watchdog timer in $\overline{\text{RESET}}$-request-mode to detect program runaway.

Setting RUN bit of WDTM to “1” starts the count. After counting starts, if RUN bit is set to “1” again within the set time interval for runaway detection, the watchdog timer is cleared and counting starts again.

If RUN bit is not set to “1” and the runaway detection time has elapsed, a $\overline{\text{RESET}}$ functions is generated.
The watchdog timer stops running in the STOP mode. Consequently, set RUN to “1” and clear the watchdog timer before entering the STOP mode. Do not set the watchdog timer when operating the HALT mode since the watchdog timer running in HALT mode.

For details of the possible time settings please refer to Figure 12-2, “Watchdog Timer Clock Selection Register (WDCS),” on page 359.

Caution: The actual runaway detection time may be up to $2^{12}/f_{XX}$ seconds less than the set time.

Chapter 13 Serial Interface Function

13.1 Features

The serial interface function provides three types of serial interfaces combining a total of 9 transmit/receive channels. All channels can be used simultaneously.
The three interface formats are as follows.

- Asynchronous serial interfaces (UART50, UART51): 2 channels
- Clocked serial interfaces (CSI00 to CSI02): 3 channels
- FCAN controller: 4 channels

Remark: For details about the FCAN controller, refer to Chapter 13 FCAN Interface Function.

UART50 and UART51 transmit/receive 1-byte serial data following a start bit and support full-duplex communication.
CSI00 to CSI02 perform data transfer according to three types of signals, namely serial clocks ($\overline{\text{SCK00}}$ to $\overline{\text{SCK02}}$), serial inputs (SI00 to SI02), and serial outputs (SO00 to SO02) (3-wire serial I/O).
FCAN conforms to CAN specification Version 2.0 Part B, and provides 64-message buffers.

13.2 Asynchronous Serial Interfaces UART5n (UART50, UART51)

13.2.1 Features

- Transfer rate: 300 bps to 625 Kbps
  (using a dedicated baud rate generator and an internal peripheral clock of 20 MHz)

- Full-duplex communications
  - On-chip reception buffer register (RXBn)
  - On-chip transmission buffer register (TXBn)

- Two-pin configuration
  - TXD5n: Transmit data output pin
  - RXD5n: Receive data input pin

- Reception error detection functions
  - Parity error
  - Framing error
  - Overrun error

- Interrupt sources: 3 types
  - Reception error interrupt (INTSERn): Interrupt is generated according to the logical OR of the three types of reception errors.
  - Reception completion interrupt (INTSRn): Interrupt is generated when receive data is transferred from the shift register to the reception buffer register after serial transfer is completed during a reception enabled state.
  - Transmission completion interrupt (INTSTn): Interrupt is generated when the serial transmission of transmit data (8 or 7 bits) from the shift register is completed.

- Character length: 7 or 8 bits

- Parity functions: Odd, even, 0, or none

- Transmission stop bits: 1 or 2 bits

- On-chip dedicated baud rate generator

Remark: n = 0, 1

13.2.2 Configuration

UART5n is controlled by the asynchronous serial interface mode register (ASIMn), asynchronous serial interface status register (ASISn), and asynchronous serial interface transmission status register (ASIFn). Receive data is maintained in the reception buffer register (RXBn), and transmit data is written to the transmission buffer register (TXBn).
Figure 13-1, “Asynchronous Serial Interfaces Block Diagram,” on page 366 shows the configuration of the asynchronous serial interface (UART5n) (n = 0, 1).

(1) Asynchronous serial interface mode registers (ASIM0, ASIM1)

The ASIMn register is an 8-bit register for specifying the operation of the asynchronous serial interface.

(2) Asynchronous serial interface status registers (ASIS0, ASIS1)

The ASISn register consists of a set of flags that indicate the error contents when a reception error occurs. The various reception error flags are set (1) when a reception error occurs and are reset (0) when the ASISn register is read.

(3) Asynchronous serial interface transmission status registers (ASIF0, ASIF1)

The ASIFn register is an 8-bit register that indicates the status when a transmit operation is performed.
This register consists of a transmission buffer data flag, which indicates the hold status of TXBn data, and the transmission shift register data flag, which indicates whether transmission is in progress.

(4) Reception control parity check

The receive operation is controlled according to the contents set in the ASIMn register. A check for parity errors is also performed during a receive operation, and if an error is detected, a value corresponding to the error contents is set in the ASISn register.

(5) Reception shift register

This is a shift register that converts the serial data that was input to the RXD5n pin to parallel data. One byte of data is received, and if a stop bit is detected, the receive data is transferred to the reception buffer register (RXBn).
This register cannot be directly manipulated.

(6) Reception buffer registers (RXB0, RXB1)

RXBn is an 8-bit buffer register for holding receive data. When 7 characters are received, 0 is stored in the MSB.
During a reception enabled state, receive data is transferred from the reception shift register to the RXBn, synchronized with the end of the shift-in processing of one frame.
Also, the reception completion interrupt request (INTSRn) is generated by the transfer of data to the RXBn.

(7) Transmission shift register

This is a shift register that converts the parallel data that was transferred from the transmission buffer register (TXBn) to serial data.
When one byte of data is transferred from the TXBn, the shift register data is output from the TXDn pin.
This register cannot be directly manipulated.

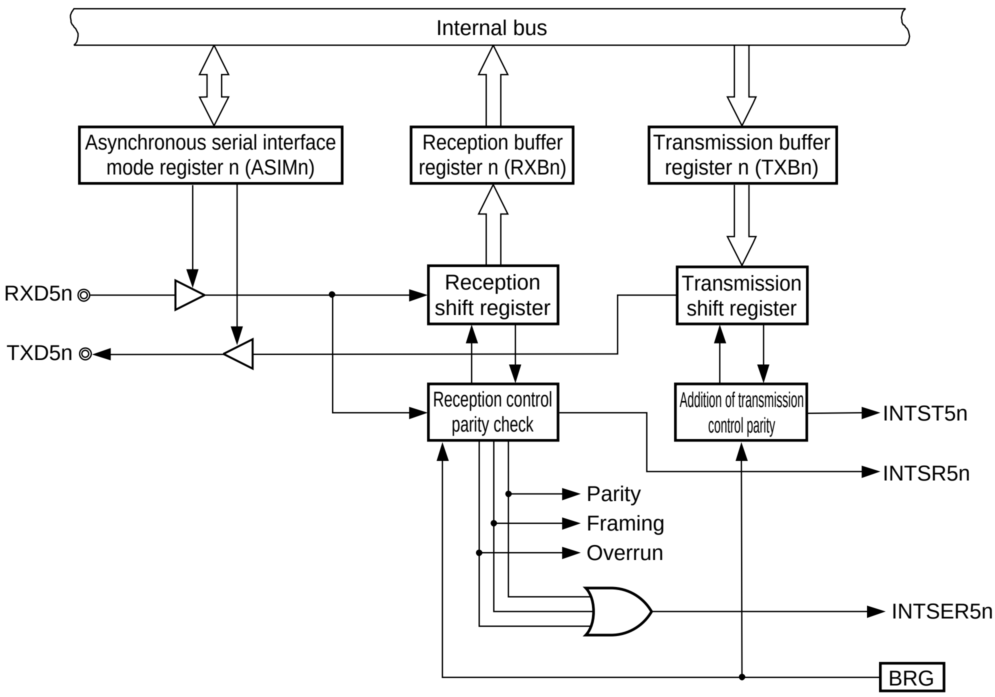
(8) Transmission buffer registers (TXB0, TXB1)

TXBn is an 8-bit buffer for transmit data. A transmit operation is started by writing transmit data to TXBn. The transmission completion interrupt request (INTSTn) is generated synchronized with the completion of transmission of one frame.

(9) Addition of transmission control parity

A transmit operation is controlled by adding a start bit, parity bit, or stop bit to the data that is written to the TXBn register, according to the contents that were set in the ASIMn register.

Figure 13-1: Asynchronous Serial Interfaces Block Diagram



Remark: n = 0, 1

13.2.3 Control registers

(1) Asynchronous serial interface mode registers (ASIM0, ASIM1)

The ASIMn register is an 8-bit register that controls the UART5n transfer operation.
This register can be read/written in 8 bit or 1-bit units (n = 0, 1).

Figure 13-2: Asynchronous Serial Interface Mode Registers (ASIM0, ASIM1) (1/3)

	7	6	5	4	3	2	1	0	Address	Initial value
ASIM0	Power	TXE	RXE	PS1	PS0	CL	SL	ISRM	FFFF FA00H	01H
ASIM1	Power	TXE	RXE	PS1	PS0	CL	SL	ISRM	FFFF FA40H	01H

Bit Position	Bit Name	Function
7	Power	Enables/disables clock operation. 0: Disable clock operation (reset internal circuit asynchronously.) 1: Enable clock operation UART5n operation clock control and asynchronous reset of the internal circuit are performed with the Power bit. When the Power bit is set to 0, the UART5n operation clock stops (fixed to low level), and an asynchronous reset is applied to internal UART5n latch. The TXDn pin output is low level when the Power bit = 0, and high level when the Power bit = 1. Therefore, perform Power setting in combination with port mode register (PM1, PM2, PM6) so as to avoid malfunction on the other side at start-up (Set the port to the output mode after setting the Power bit to 1). Input from the RXDn pin is fixed to high level with Power bit = 0.
6	TXE	Enables/disables transfer. 0: Disable transfer (Perform synchronized reset of transfer circuit.) 1: Enable transfer **Cautions: 1. Set the TXE bit to 1 after setting the Power bit to 1 when starting transfer. Set the Power bit to 0 after setting the TXE bit to 0 when stopping transfer.** **2. To initialize the transfer unit, clear (0) the TXE bit, and after letting 2 Clock cycles (base clock) elapse, set (1) the TXE bit again. If the TXE bit is not set again, initialization may not be successful. (For details about the base clock, refer to 13.2.6 “Dedicated baud rate generators (BRG) of UART5n (n = 0, 1)” on page 384.)**

Figure 13-2: Asynchronous Serial Interface Mode Registers (ASIM0, ASIM1) (2/3)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>5</td><td>RXE</td><td>Enables/disables reception.
0: Disable reception (Perform synchronous reset of reception circuit)
1: Enable reception
Cautions: 1. Set the RXE bit to 1 after setting the Power bit to 1 when starting transfer. Set the Power bit to 0 after setting the RXE bit to 0 when stopping transfer.
2. To initialize the reception unit status, clear (0) the RXE bit, and after letting 2 Clock cycles (base clock) elapse, set (1) the RXE bit again. If the RXE bit is not set again, initialization may not be successful. (For details about the base clock, refer to 13.2.6 “Dedicated baud rate generators (BRG) of UART5n (n = 0, 1)” on page 384</td></tr>
<tr><td>4, 3</td><td>PS1, PS0</td><td>Controls parity bit.
<table>
<tr><th>PS1</th><th>PS0</th><th>Transmit Operation</th><th>Receive Operation</th></tr>
<tr><td>0</td><td>0</td><td>Don’t output parity bit</td><td>Receive with no parity</td></tr>
<tr><td>0</td><td>1</td><td>Output 0 parity</td><td>Receive as 0 parity</td></tr>
<tr><td>1</td><td>0</td><td>Output odd parity</td><td>Judge as odd parity</td></tr>
<tr><td>1</td><td>1</td><td>Output even parity</td><td>Judge as even parity</td></tr>
</table>
Cautions: 1. To overwrite the PS1 and PS0 bits, first clear (0) the TXE and RXE bits.
2. If "0 parity" is selected for reception, no parity judgment is performed. Therefore, no error interrupt is generated because the PE bit of the ASISn register is not set.
• Even parity
If the transmit data contains an odd number of bits with the value “1”, the parity bit is set (1). If it contains an even number of bits with the value “1”, the parity bit is cleared (0). This controls the number of bits with the value “1” contained in the transmit data and the parity bit so that it is an even number.
During reception, the number of bits with the value “1” contained in the receive data and the parity bit is counted, and if the number is odd, a parity error is generated.
• Odd parity
In contrast to even parity, odd parity controls the number of bits with the value “1” contained in the transmit data and the parity bit so that it is an odd number. During reception, the number of bits with the value “1” contained in the receive data and the parity bit is counted, and if the number is even, a parity error is generated.</td></tr>
</table>

Remark: When reception is disabled, the reception shift register does not detect a start bit. No shift-in processing or transfer processing to the reception buffer register (RXBn) is performed, and the contents of the RXBn register are retained.
When reception is enabled, the reception shift operation starts, synchronized with the detection of the start bit, and when the reception of one frame is completed, the contents of the reception shift register are transferred to the RXBn register. A reception completion interrupt (INTSRn) is also generated in synchronization with the transfer to the RXBn register.

Figure 13-2: Asynchronous Serial Interface Mode Registers (ASIM0, ASIM1) (3/3)

Bit Position	Bit Name	Function
4, 3	PS1, PS0	• 0 parity During transmission, the parity bit is cleared (0) regardless of the transmit data. During reception, no parity error is generated because no parity bit is checked. • No parity No parity bit is added to transmit data. During reception, the receive data is considered to have no parity bit. No parity error is generated because there is no parity bit.
2	CL	Specifies character length of transmit/receive data. 0: 7 bits 1: 8 bits **Caution: To overwrite the CL bit, first clear (0) the TXE and RXE bits.**
1	SL	Specifies stop bit length of transmit data. 0: 1 bit 1: 2 bits **Caution: To overwrite the SL bit, first clear (0) the TXE bit. Since reception is always done using a single stop bit, the SL bit setting does not affect receive operations.**
0	ISRM	Enables/disables generation of reception completion interrupt requests when an error occurs. 0: Generate a reception error interrupt request (INTSERn) as an interrupt when an error occurs. In this case, no reception completion interrupt request (INTSRn) is generated. 1: Generate a reception completion interrupt request (INTSRn) as an interrupt when an error occurs. In this case, no reception error interrupt request (INTSERn) is generated. **Caution: To overwrite the ISRM bit, first clear (0) the RXE bit.**

(2) Asynchronous serial interface status registers (ASIS0 to ASIS2)

The ASISn register, which consists of 3-bit error flags (PE, FE and OVE), indicates the error status when UART5n reception is completed.
The status flag, which indicates a reception error, always indicates the status of the error that occurred most recently. That is, if the same error occurred several times before the receive data was read, this flag would hold only the status of the error that occurred last.
The ASISn register is cleared to 00H by a read operation. When a reception error occurs, the reception buffer register (RXBn) should be read and the error flag should be cleared after the ASISn register is read.
This register is read-only in 8-bit or 1-bit units (n = 0, 1).

Caution: When the Power bit or RXE bit of the ASIMn register is set to 0, or when the ASIS0 register is read, the PE, FE, and OVE bits of the ASISn register are cleared (0).

Figure 13-3: Asynchronous Serial Interface Status Registers (ASIS0, ASIS1)

	7	6	5	4	3	2	1	0	Address	Initial value
ASIS0	0	0	0	0	0	PE	FE	OVE	FFFF FA03H	00H
ASIS1	0	0	0	0	0	PE	FE	OVE	FFFF FA43H	00H

Bit Position	Bit Name	Function
2	PE	This is a status flag that indicates a parity error. 0: When the ASIMn register's Power and RXE bits are both set to 0, or when the ASISn register has been read 1: When reception was completed, the transmit data parity did not match the parity bit **Caution: The operation of the PE bit differs according to the settings of the PS1 and PS0 bits of the ASIMn register.**
1	FE	This is a status flag that indicates a framing error. 0: When the ASIMn register's Power and RXE bits are both set to 0, or when the ASISn register has been read 1: When reception was completed, no stop bit was detected **Caution: For receive data stop bits, only the first bit is checked regardless of the number of stop bits.**
0	OVE	This is a status flag that indicates an overrun error. 0: When the ASIMn register's Power and RXE bits are both 0, or when the ASISn register has been read. 1: UART5n completed the next receive operation before reading the RXBn receive data. **Caution: When an overrun error occurs, the next receive data value is not written to the RXBn register and the data is discarded.**

(3) Asynchronous serial interface transmission status registers (ASIF0, ASIF1)

The ASIFn register, which consists of 2-bit status flags, indicates the status during transmission. By writing the next data to the TXBn register after data is transferred from the TXBn register to the transmission shift register, transmit operations can be performed continuously without suspension even during an interrupt interval. When transmission is performed continuously, data should be written after referencing the ASIFn register to prevent writing to the TXBn register by mistake.
This register is read-only in 8-bit or 1-bit units (n = 0, 1).

Figure 13-4: Asynchronous Serial Interface Transmit Status Registers (ASIF0, ASIF1)

	7	6	5	4	3	2	1	0	Address	Initial value
ASIF0	0	0	0	0	0	0	TXBF0	TXSF0	FFFF FA05H	00H
ASIF1	0	0	0	0	0	0	TXBF0	TXSF0	FFFF FA45H	00H

Bit Position	Bit Name	Function
1	TXBF	This is a transmission buffer data flag. 0: When the ASIMn register’s Power or TXE bits is 0, or when data has been transferred to the transmission shift register (Data to be transferred next to TXBn register does not exist). 1: Data exists in TXBn register when the TXBn register has been written to (Data to be transferred next exists in TXBn register).
0	TXSF	This is a transmission shift register data flag. It indicates the transmission status of UART5n. 0: When the ASIMn register’s Power or TXE bits is set to 0, or when following transfer completion, the next data transfer from the TXBn register is not performed (waiting transmission) 1: When data has been transferred from the TXBn register (Transmission in progress)

The following table shows relationships between the transmission status and write operations to TXBn register.

TXBF	TXSF	Transmission Status	Write Operation to TXBn
0	0	Initial status or transmission completed	Writing is permitted
0	1	Transmission in progress (no data is in TXBn)	Writing is permitted
1	0	Waiting transmission (data is in TXBn)	Writing is not permitted
1	1	Transmission in progress (data is in TXBn)	Writing is not permitted

Caution: When transmission is performed continuously, data should be written to TXBn register after confirming the TXBF bit value. If writing is not permitted, transmit data cannot be guaranteed when data is written to TXBn register.

(4) Reception buffer registers (RXB0, RXB1)

The RXBn register is an 8-bit buffer register for storing parallel data that had been converted by the reception shift register.

When reception is enabled (RXE bit = 1 in the ASIMn register), receive data is transferred from the reception shift register to the RXBn register, synchronized with the completion of the shift-in processing of one frame. Also, a reception completion interrupt request (INTSRn) is generated by the transfer to the RXBn register. For information about the timing for generating this interrupt request, refer to 13.2.5 (4)"Receive operation" on page 380.

If reception is disabled (RXE bit = 0 in the ASIMn register), the contents of the RXBn register are retained, and no processing is performed for transferring data to the RXBn register even when the shift-in processing of one frame is completed. Also, no reception completion interrupt is generated.

When 7 bits is specified for the data length, bits 6 to 0 of the RXBn register are transferred for the receive data and the MSB (bit 7) is always 0. However, if an overrun error (OVE) occurs, the receive data at that time is not transferred to the RXBn register.

Except when a reset is input, the RXBn register becomes FFH even when Power bit = 0 in the ASIMn register.

This register is read-only in 8-bit or 1-bit units (n = 0, 1).

Figure 13-5: Reception Buffer Registers (RXB0, RXB1)

	7	6	5	4	3	2	1	0	Address	Initial value
RXB0	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	FFFF FA02H	FFH
RXB1	RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0	FFFF FA42H	FFH

Bit Position	Bit Name	Function
7 to 0	RXB7 to RXB0	Stores receive data. 0 can be read for RXB7 when 7-bit or character data is received.

(5) Transmission buffer registers (TXB0, TXB1)

The TXBn register is an 8-bit buffer register for setting transmit data.
When transmission is enabled (TXE bit = 1 in the ASIMn register), the transmit operation is started by writing data to TXBn register.
When transmission is disabled (TXE bit = 0 in the ASIMn register), even if data is written to TXBn register, the value is ignored.
The TXBn register data is transferred to the transmission shift register, and a transmission completion interrupt request (INTSTn) is generated, synchronized with the completion of the transmission of one frame from the transmission shift register. For information about the timing for generating this interrupt request, refer to 13.2.5 (2)“Transmit operation” on page 376.
When TXBF bit = 1 in the ASIFn register, writing must not be performed to TXBn register.
This register can be read or written in 8-bit or 1-bit units (n = 0, 1).

Figure 13-6: Transmission Buffer Registers (TXB0, TXB1)

	7	6	5	4	3	2	1	0	Address	Initial value
TXB0	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0	FFFF FA04H	FFH
TXB1	TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0	FFFF FA44H	FFH

Bit Position	Bit Name	Function
7 to 0	TXB7 to TXB0	Writes transmit data.

13.2.4 Interrupt requests

The following three types of interrupt requests are generated from UART50, UART51.

- Reception error interrupt (INTSERn)
- Reception completion interrupt (INTSRn)
- Transmission completion interrupt (INTSTn)

The default priorities among these three types of interrupt requests is, from high to low, reception error interrupt, reception completion interrupt, and transmission completion interrupt (n = 0, 1).

Table 13-1: Generated Interrupts and Default Priorities

Interrupt	Priority
Reception error	1
Reception completion	2
Transmission completion	3

(1) Reception error interrupt (INTSER0, INTSER1)

When reception is enabled, a reception error interrupt is generated according to the logical OR of the three types of reception errors explained for the ASISn register. Whether a reception error interrupt (INTSERn) or a reception completion interrupt (INTSRn) is generated when an error occurs can be specified according to the ISRM bit of the ASIMn register.
When reception is disabled, no reception error interrupt is generated.

(2) Reception completion interrupt (INTSR0, INTSR1)

When reception is enabled, a reception completion interrupt is generated when data is shifted in to the reception shift register and transferred to the reception buffer register (RXBn).
A reception completion interrupt request can be generated in place of a reception error interrupt according to the ISRM bit of the ASIMn register even when a reception error has occurred.
When reception is disabled, no reception completion interrupt is generated.

(3) Transmission completion interrupt (INTST0, INTST1)

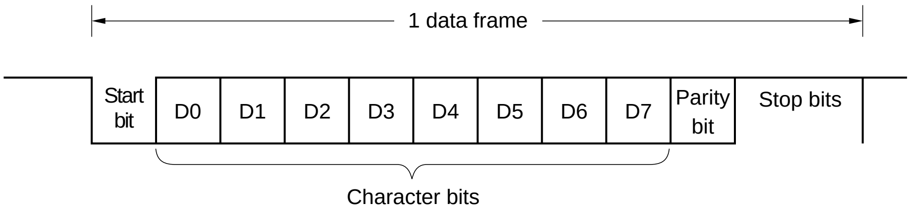
A transmission completion interrupt is generated when one frame of transmit data containing 7-bit or 8-bit characters is shifted out from the transmission shift register.

13.2.5 Operation

(1) Data format

Full-duplex serial data transmission and reception can be performed.
The transmit/receive data format consists of one data frame containing a start bit, character bits, a parity bit, and stop bits as shown in Figure 13-7.
The character bit length within one data frame, the type of parity, and the stop bit length are specified according to the asynchronous serial interface mode register (ASIMn) (n = 0, 1).
Also, data is transferred with LSB first.

Figure 13-7: Asynchronous Serial Interface Transmit/Receive Data Format



- Start bit ··· 1 bit
- Character bits ··· 7 bits or 8 bits
- Parity bit ··· Even parity, odd parity, 0 parity, or no parity
- Stop bits ··· 1 bit or 2 bits

(2) Transmit operation

When Power bit is set to 1 in the ASIMn register, a high level is output from the TXD5n pin.
Then, when TXE bit is set to 1 in the ASIMn register, transmission is enabled, and the transmit operation is started by writing transmit data to transmission buffer register (TXBn) (n = 0, 1).

(a) Transmission enabled state

This state is set by the TXE bit in the ASIMn register.

- TXE = 1: Transmission enabled state
- TXE = 0: Transmission disabled state

Since UART5n does not have a CTS (transmission enabled signal) input pin, a port should be used to confirm whether the destination is in a reception enabled state.

(b) Starting a transmit operation

In transmission enabled state, a transmit operation is started by writing transmit data to transmission buffer register (TXBn). When a transmit operation is started, the data in TXBn is transferred to transmission shift register. Then, the transmission shift register outputs data to the TXD5n pin (the transmit data is transferred sequential starting with the start bit). The start bit, parity bit, and stop bits are added automatically.

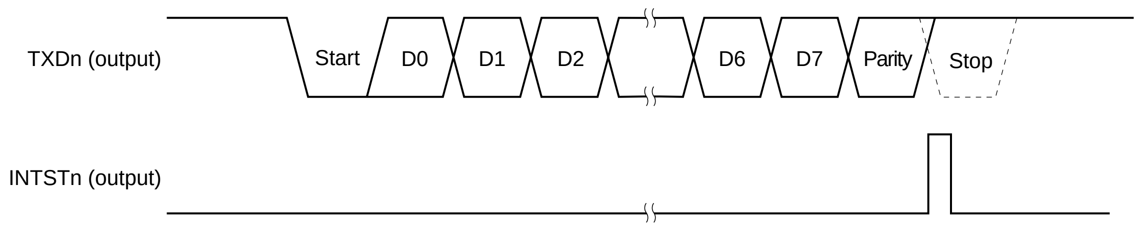
(c) Transmission interrupt request

When the transmission shift register becomes empty, a transmission completion interrupt request (INTSTn) is generated. The timing for generating the INTSTn interrupt differs according to the specification of the number of stop bits. The INTSTn interrupt is generated at the same time that the last stop bit is output.
If the data to be transmitted next has not been written to the TXBn register, the transmit operation is suspended.

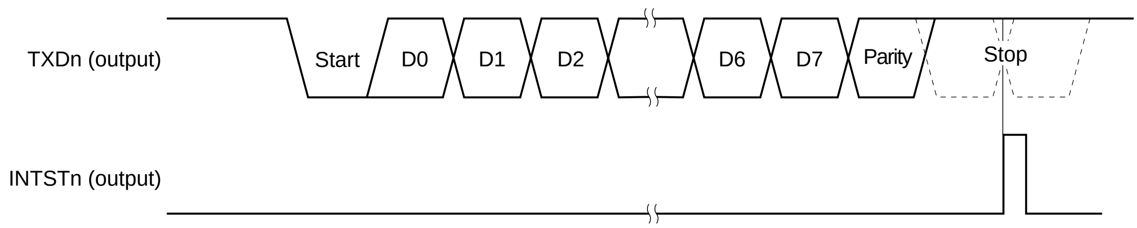
Caution: Normally, when the transmission shift register becomes empty, a transmission completion interrupt (INTSTn) is generated. However, no transmission completion interrupt (INTSTn) is generated if the transmission shift register becomes empty due to the input of a $\overline{\text{RESET}}$.

Figure 13-8: Asynchronous Serial Interface Transmission Completion Interrupt Timing

(a) Stop bit length: 1



(b) Stop bit length: 2



(3) Continuous transmission operation

UART5n can write the next transmit data to the TXBn register at the time that the transmission shift register starts the shift operation. This enables an efficient transmission rate to be realized by continuously transmitting data even during the INTSTn interrupt service after the transmission of one data frame.

When continuous transmission is performed, data should be written after referencing the ASIFn register to confirm the transmission status and whether or not data can be written to the TXBn register (n = 0, 1).

Caution: Transmit data should be written when the TXBF bit is 0. The transmission unit should be initialized when the TXSF bit is 0. If these actions are performed at other times, the transmit data cannot be guaranteed.

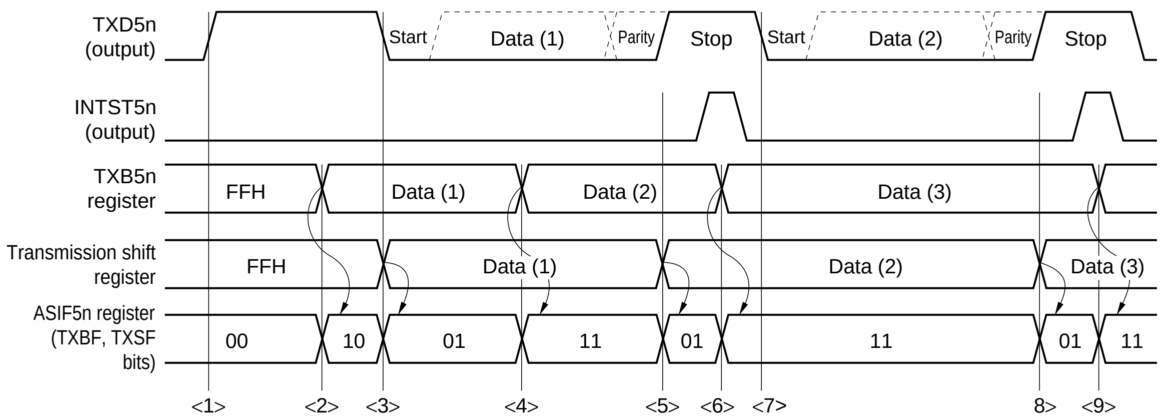
Table 13-2: Transmission Status and Whether or Not Writing Is Enabled

TXBF	TXSF	Transmission Status	Whether or not Write Operation to TXBn is Enabled
0	0	Initial status or transmission completed	Writing is enabled
0	1	Transmission in progress (no data is in TXBn register)	Writing is enabled
1	0	Awaiting transmission (data is in TXBn register)	Writing is not enabled
1	1	Transmission in progress (data is in TXBn register)	Writing is not enabled

(a) Starting procedure

Figure 13-9 shows the procedure to start continuous transmission.

Figure 13-9: Continuous Transmission Starting Procedure

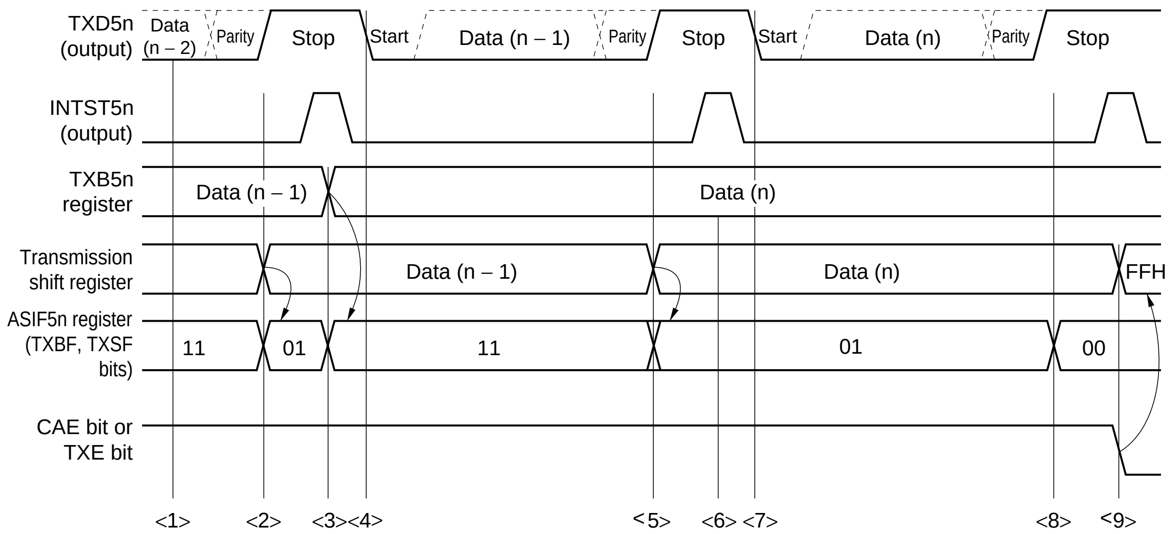


Transmission Starting Procedure	Internal Operation	ASIFn Register	
		TXBF bit	TXSF bit
<1> Set transmission mode		0	0
<2> Write data (1) to TXBn register		1	0
	<3> Generate start bit Start data (1) transmission[Note]	0	1
<4> Read ASIFn register (confirm that TXBF bit = 0)			
Write data (2)		1	1
	<<Transmission in progress>> <5> Generate transmission completion interrupt (INTSTn)	0	1
<6> Read ASIFn register (confirm that TXBF bit = 0)			
Write data (3)		1	1
	<7> Generate start bit		
	Start data (2) transmission		
	<<Transmission in progress>> <8> Generate transmission completion interrupt (INTSTn)	0	1
<9> Read ASIFn register (confirm that TXBF bit = 0)			
Write data (4)		1	1

Note: For a certain time it may happen that the bit combinations of TXBF and TXSF bits 00B or 11B can be read.

(b) Ending procedure

Figure 13-10: Continuous Transmission End Procedure



Transmission End Procedure	Internal Operation	ASIFn Register TXBF Bit	ASIFn Register TXSF Bit
	<1> Transmission of data (n - 2) is in progress	1	1
<3> Read ASIFn register (confirm that TXBF bit = 0)	<2> Generate transmission completion interrupt (INTSTn)	0	1
Write data (n)	<4> Generate start bit Start data (n - 1) transmission <<Transmission in progress>>	1	1
<6> Read ASIFn register (confirm that TXSF bit = 1) There is no write data	<5> Generate transmission completion interrupt (INTSTn) <7> Generate start bit Start data (n) transmission <<Transmission in progress>>	0	1
<9> Read ASIFn register (confirm that TXSF bit = 0) Clear (0) the Power bit or TXE bit of ASIMn register	<8> Generate transmission completion interrupt (INTSTn) Initialize internal circuits	0	0

(4) Receive operation

An awaiting reception state is set by setting Power bit to 1 in the ASIMn register and then setting RXE bit to 1 in the ASIMn register. To start a receive operation, detects a start bit first. The start bit is detected by sampling RXD5n pin. When the receive operation begins, serial data is stored sequential in the reception shift register according to the baud rate that was set. A reception completion interrupt (INTSRn) is generated each time the reception of one frame of data is completed. Normally, the receive data is transferred from the reception buffer register (RXBn) to memory by this interrupt servicing (n = 0, 1).

(a) Reception enabled state

The receive operation is set to reception enabled state by setting the RXE bit in the ASIM0 register to 1.

- RXE bit = 1: Reception enabled state
- RXE bit = 0: Reception disabled state

In reception disabled state, the reception hardware stands by in the initial state. At this time, the contents of the reception buffer register (RXBn) are retained, and no reception completion interrupt or reception error interrupt is generated.

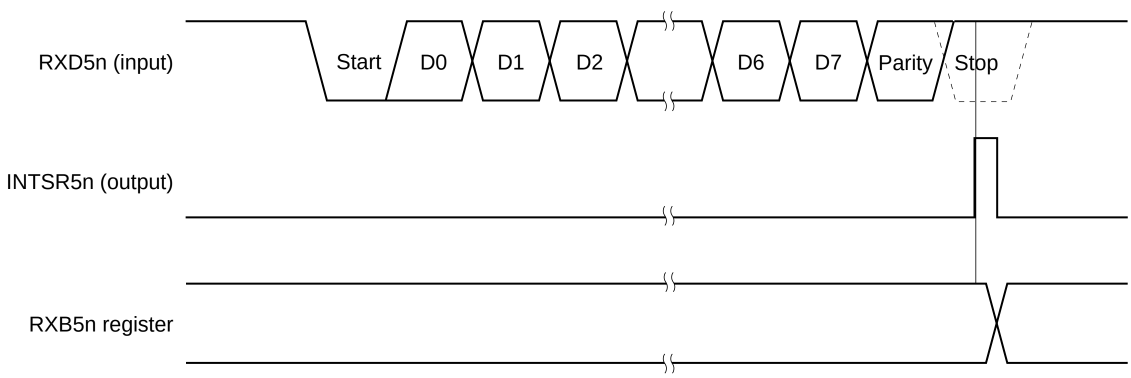
(b) Starting a receive operation

A receive operation is started by the detection of a start bit.
The RXDn pin is sampled according to the serial clock from the dedicated baud rate generator (BRG) of UART5n (n = 0, 1).

(c) Reception completion interrupt

When RXE bit = 1 in the ASIMn register and the reception of one frame of data is completed (the stop bit is detected), a reception completion interrupt (INTSRn) is generated and the receive data within the reception shift register is transferred to RXBn at the same time.
Also, if an overrun error (OVE) occurs, the receive data at that time is not transferred to the reception buffer register (RXBn), and either a reception completion interrupt (INTSRn) or a reception error interrupt (INTSERn) is generated (the receive data within the reception shift register is transferred to RXBn) according to the ISRM bit setting in the ASIMn register.
Even if a parity error (PE) or framing error (FE) occurs during a reception operation, the receive operation continues until stop bit is received, and after reception is completed, either a reception completion interrupt (INTSRn) or a reception error interrupt (INTSERn) is generated according to the ISRM bit setting in the ASIMn register.
If the RXE bit is reset (0) during a receive operation, the receive operation is immediately stopped. The contents of the reception buffer register (RXBn) and of the asynchronous serial interface status register (ASISn) at this time do not change, and no reception completion interrupt (INTSRn) or reception error interrupt (INTSERn) is generated.
No reception completion interrupt is generated when RXE bit = 0 (reception is disabled).

Figure 13-11: Asynchronous Serial Interface Reception Completion Interrupt Timing



(5) Reception error

The three types of error that can occur during a receive operation are a parity error, framing error, or overrun error. The data reception result is that the various flags of the ASISn register are set (1), and a reception error interrupt (INTSERn) or a reception completion interrupt (INTSRn) is generated at the same time. The ISRM bit of the ASIMn register specifies whether INTSERn or INTSRn is generated.

The type of error that occurred during reception can be detected by reading the contents of the ASISn register during the INTSERn or INTSRn interrupt servicing (n = 0, 1).

The contents of the ASISn register are reset (0) by reading it.

Table 13-3: Reception Error Causes

Error Flag	Reception Error	Cause
PE	Parity error	The parity specification during transmission did not match the parity of the reception data
FE	Framing error	No stop bit was detected
OVE	Overrun error	The reception of the next data was completed before data was read from the reception buffer register (RXBn)

(a) Separation of reception error interrupt

A reception error interrupt can be separated from the INTSRn interrupt and generated as an INTSERn interrupt by clearing the ISRM bit of the ASIMn register to 0.

Figure 13-12: When Reception Error Interrupt Is Separated from INTSRn Interrupt (ISRM Bit = 0)

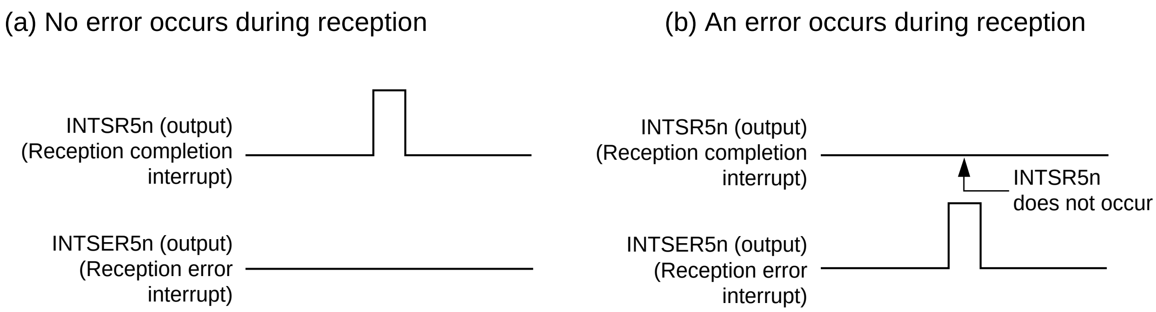
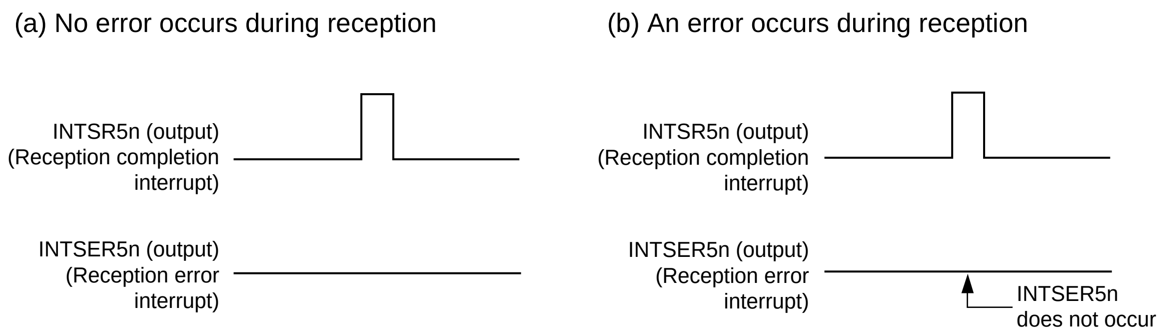


Figure 13-13: When Reception Error Interrupt Is Included in INTSRn Interrupt (ISRM Bit = 1)



(6) Parity types and corresponding operation

A parity bit is used to detect a bit error in communication data. Normally, the same type of parity bit is used at the transmission and reception sides.

(a) Even parity

- During transmission
The parity bit is controlled so that the number of bits with the value “1” within the transmit data including the parity bit is even. The parity bit value is as follows.

- If the number of bits with the value “1” within the transmit data is odd: 1
- If the number of bits with the value “1” within the transmit data is even: 0

- During reception
The number of bits with the value “1” within the receive data including the parity bit is counted, and a parity error is generated if this number is odd.

(b) Odd parity

- During transmission
In contrast to even parity, the parity bit is controlled so that the number of bits with the value “1” within the transmit data including the parity bit is odd. The parity bit value is as follows.

- If the number of bits with the value “1” within the transmit data is odd: 0
- If the number of bits with the value “1” within the transmit data is even: 1

- During reception
The number of bits with the value “1” within the receive data including the parity bit is counted, and a parity error is generated if this number is even.

(c) 0 parity

- During transmission the parity bit is set to “0” regardless of the transmit data.
- During reception, no parity bit check is performed. Therefore, no parity error is generated regardless of whether the parity bit is “0” or “1”.

(d) No parity

- No parity bit is added to the transmit data.
- During reception, the receive operation is performed as if there were no parity bit. Since there is no parity bit, no parity error is generated.

(7) Receive data noise filter

The RXD5n signal is sampled at the rising edge of the prescaler output basic clock (Clock). If the same sampling value is obtained twice, the match detector output changes, and this output is sampled as input data. Therefore, data not exceeding one clock width is judged to be noise and is not delivered to the internal circuit (see Figure 12-15). Refer to 12.2.6 (1) (a) Basic clock (Clock) regarding the basic clock.

Also, since the circuit is configured as shown in Figure 12-14, internal processing during a receive operation is delayed by up to 2 clocks according to the external signal status.

Figure 13-14: Noise Filter Circuit

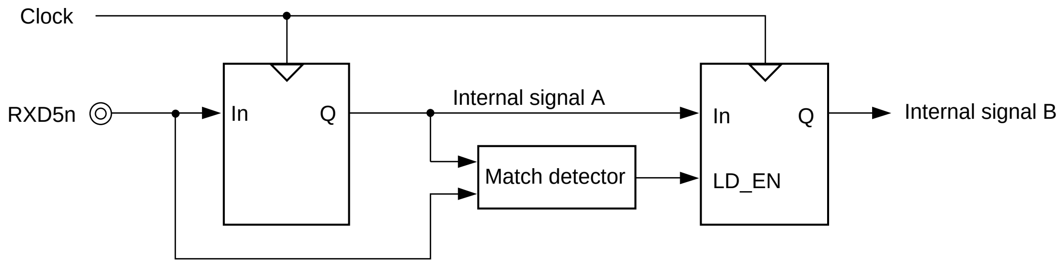
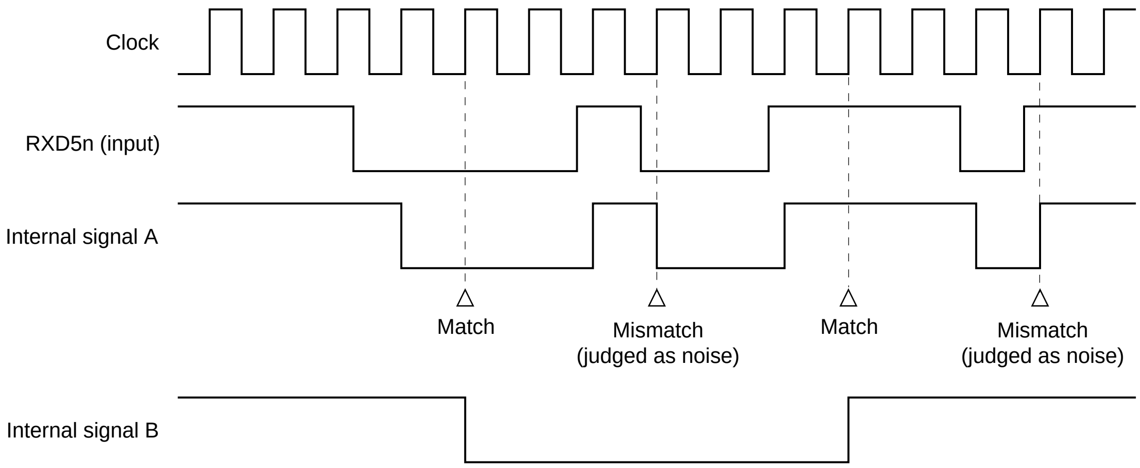


Figure 13-15: Timing of RXD5n Signal Judged as Noise



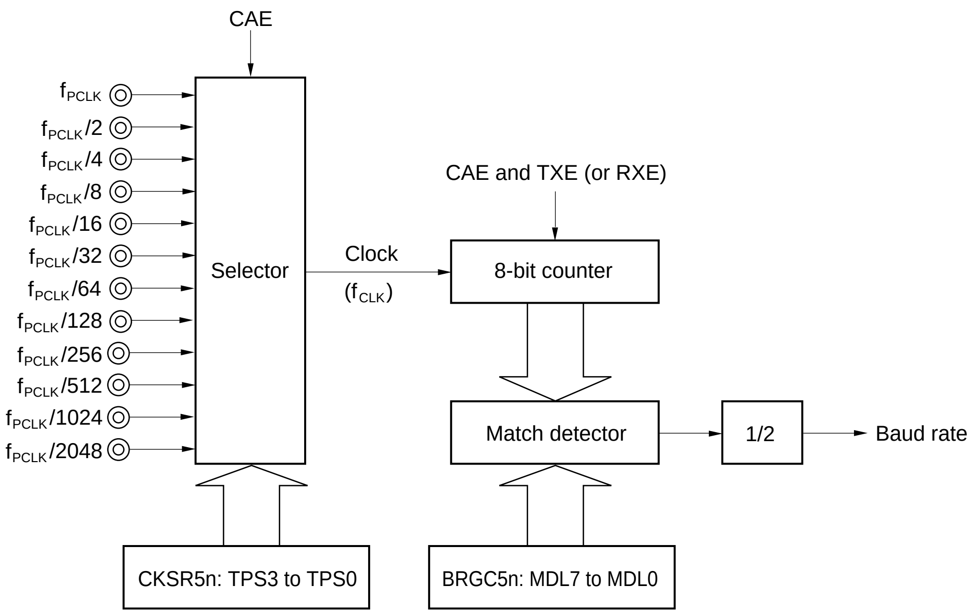
Remark: n = 0, 1

13.2.6 Dedicated baud rate generators (BRG) of UART5n (n = 0, 1)

A dedicated baud rate generator, which consists of a source clock selector and an 8-bit programmable counter, generates serial clocks during transmission/reception at UART5n (n = 0, 1). The dedicated baud rate generator output can be selected as the serial clock for each channel.
Separate 8-bit counters exist for transmission and for reception.

(1) Baud rate generator configuration

Figure 13-16: Baud Rate Generator (BRG) Configuration of UART5n (n = 0, 1)



Remark: n = 0, 1

(a) Basic clock (Clock)

When Power bit = 1 in the ASIMn register, the clock selected according to the TPS3 to TPS0 bits of the CKSRm register is supplied to the transmission/reception unit. This clock is called the basic clock (Clock), and its frequency is referred to as f_{CLK}. When Power bit = 0, Clock is fixed at low level.

(2) Serial clock generation

A serial clock can be generated according to the settings of the CKSRm and BRGCm registers.
The basic clock to the 8-bit counter is selected according to the TPS3 to TPS0 bits of the CKSRm register.
The 8-bit counter divisor value can be set according to the MDL7 to MDL0 bits of the BRGCm register (m = 0, 1).

(a) Clock select registers (CHKSR0, CHKSR1)

The CKSRm register is an 8-bit register for selecting the basic block according to the TPS3 to TPS0 bits. The clock selected by the TPS3 to TPS0 bits becomes the basic clock (Clock) of the transmission/ reception module. Its frequency is referred to as f_{CLK}.
This register can be read or written in 8-bit or 1-bit units.

Figure 13-17: Clock Select Registers (CHKSR0, CHKSR1)

	7	6	5	4	3	2	1	0	Address	Initial value
CHKSR0	0	0	0	0	TPS3	TPS2	TPS1	TPS0	FFFF FA06H	00H
CHKSR1	0	0	0	0	TPS3	TPS2	TPS1	TPS0	FFFF FA46H	00H

Bit Position	Bit Name	Function
3 to 0	TPS3 to TPS0	Specifies the basic clock

TPS3	TPS2	TPS1	TPS0	Basic Clock (f_{CLK})
0	0	0	0	f_{PCLK}
0	0	0	1	$f_{PCLK}/2$
0	0	1	0	$f_{PCLK}/4$
0	0	1	1	$f_{PCLK}/8$
0	1	0	0	$f_{PCLK}/16$
0	1	0	1	$f_{PCLK}/32$
0	1	1	0	$f_{PCLK}/64$
0	1	1	1	$f_{PCLK}/128$
1	0	0	0	$f_{PCLK}/256$
1	0	0	1	$f_{PCLK}/512$
1	0	1	0	$f_{PCLK}/1024$
1	0	1	1	$f_{PCLK}/2048$
1	1	Arbitrary	Arbitrary	Setting prohibited

Remark: f_{PCLK}: internal peripheral clock.

(b) Baud rate generator control registers (BRGC0, BRGC1)

The BRGCm register is an 8-bit register that controls the baud rate (serial transfer speed) of UART5n.

This register can be read or written in 8-bit or 1-bit units (m = 0, 1).

Figure 13-18: Baud Rate Generator Control Registers (BRGC0, BRGC1)

	7	6	5	4	3	2	1	0	Address	Initial value
BRGC0	MDL7	MDL6	MDL5	MDL4	MDL3	MDL2	MDL1	MDL0	FFFF FA07H	FFH
BRGC1	MDL7	MDL6	MDL5	MDL4	MDL3	MDL2	MDL1	MDL0	FFFF FA47H	FFH

Bit Position: 7 to 0 — Bit Name: MDL7 to MDL0 — Function:

MDL7	MDL6	MDL5	MDL4	MDL3	MDL2	MDL1	MDL0	Divisor Value (k)	Serial Clock
0	0	0	0	0	x	x	x	–	Setting prohibited
0	0	0	0	1	0	0	0	8	$f_{CLK}/8$
0	0	0	0	1	0	0	1	9	$f_{CLK}/9$
0	0	0	0	1	0	1	0	10	$f_{CLK}/10$
:	:	:	:	:	:	:	:	:	:
1	1	1	1	1	0	1	0	250	$f_{CLK}/250$
1	1	1	1	1	0	1	1	251	$f_{CLK}/251$
1	1	1	1	1	1	0	0	252	$f_{CLK}/252$
1	1	1	1	1	1	0	1	253	$f_{CLK}/253$
1	1	1	1	1	1	1	0	254	$f_{CLK}/254$
1	1	1	1	1	1	1	1	255	$f_{CLK}/255$

Caution: If the MDL7 to MDL0 bits are to be overwritten, TXE bit and RXE bit should be set to 0 in the ASIMn register first.

Remarks: 1. f_{CLK}:Frequency [Hz] of basic clock selected according to TPS3 to TPS0 bits of CKSRm register

2. k: Value set according to MDL7 to MDL0 bits (k = 8, 9, 10,..., 255)

3. The baud rate is the output clock for the 8-bit counter divided by 2

4. x: don't care

(c) Baud rate

The baud rate is the value obtained according to the following formula.

$$\text{Baud rate} = \frac{f_{CLK}}{2 \cdot k} \quad [bps]$$

f_{CLK} = Frequency [Hz] of basic clock selected according to TPS3 to TPS0 bits of CKSRm register.

k = Value set according to MDL7 to MDL0 bits of BRGCm register (k = 8, 9, 10,..., 255)

(d) Baud rate error

The baud rate error is obtained according to the following formula.

$$\boldsymbol{Error} = \left(\frac{\textbf{Actual baud rate (baud rate with error)}}{\textbf{Desired baud rate (normal baud rate)}} - \mathbf{1}\right) \times \mathbf{100} \quad [\%]$$

Cautions: 1. Make sure that the baud rate error during transmission does not exceed the allowable error of the reception destination.

2. Make sure that the baud rate error during reception is within the allowable baud rate range during reception, which is described in chapter 13.2.6 (3)"Allowable baud rate range during reception" on page 389.

Example: Basic clock frequency = 10 MHz
Settings of MDL7 to MDL0 bits in BRGC0 register = 01000001B (k = 65)
Target baud rate = 76800 bps

Baud rate = $10 \times 10^6 / (2 \times 65)$
= 76923 bps

Error = (76923/76800 - 1) × 100
= 0.160%

(e) Baud rate setting example

Table 13-4: Baud Rate Generator Setting Data

Baud Rate [bps]	f_{PCLK} = 20 MHz			f_{PCLK} = 16 MHz			f_{PCLK} = 5 MHz			f_{PCLK} = 4 MHz		
	f_{CLK}	k	ERR	f_{CLK}	k	ERR	f_{CLK}	k	ERR	f_{CLK}	k	ERR
300	$f_{PCLK}/256$	130	0.16	$f_{PCLK}/256$	104	0.16	$f_{PCLK}/64$	130	0.16	$f_{PCLK}/64$	104	0.16
600	$f_{PCLK}/128$	130	0.16	$f_{PCLK}/128$	104	0.16	$f_{PCLK}/32$	130	0.16	$f_{PCLK}/32$	104	0.16
1200	$f_{PCLK}/64$	130	0.16	$f_{PCLK}/64$	104	0.16	$f_{PCLK}/16$	130	0.16	$f_{PCLK}/16$	104	0.16
2400	$f_{PCLK}/32$	130	0.16	$f_{PCLK}/32$	104	0.16	$f_{PCLK}/8$	130	0.16	$f_{PCLK}/8$	104	0.16
4800	$f_{PCLK}/16$	130	0.16	$f_{PCLK}/16$	104	0.16	$f_{PCLK}/4$	130	0.16	$f_{PCLK}/4$	104	0.16
9600	$f_{PCLK}/8$	130	0.16	$f_{PCLK}/8$	104	0.16	$f_{PCLK}/2$	130	0.16	$f_{PCLK}/2$	104	0.16
19200	$f_{PCLK}/4$	130	0.16	$f_{PCLK}/4$	104	0.16	$f_{PCLK}/2$	65	0.16	$f_{PCLK}/2$	64	0.16
31250	$f_{PCLK}/2$	160	0	$f_{PCLK}/2$	128	0	$f_{PCLK}/2$	40	0	$f_{PCLK}/2$	32	0
38400	$f_{PCLK}/2$	130	0.16	$f_{PCLK}/2$	104	0.16	$f_{PCLK}/2$	33	-1.38	$f_{PCLK}/2$	26	0.16
76800	$f_{PCLK}/2$	65	0.16	$f_{PCLK}/2$	52	0.16	$f_{PCLK}/2$	16	1,70	$f_{PCLK}/2$	13	0.16
153600	$f_{PCLK}/2$	33	-1.38	$f_{PCLK}/2$	26	0.16	$f_{PCLK}/2$	8	1.70	$f_{PCLK}/2$	7	-7.00

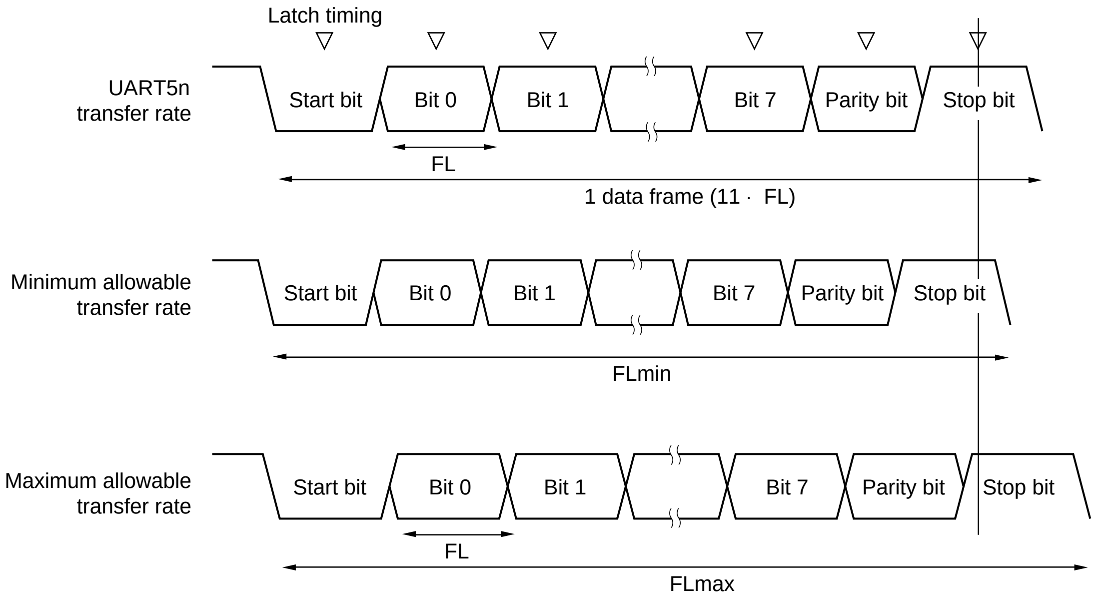
Remark: f_{PCLK}: System clock frequency
f_{CLK}: Basic clock frequency
k: Setting values of MDL7 to MDL0 bits in BRGCm register
ERR: Baud rate error [%]

(3) Allowable baud rate range during reception

The degree to which a discrepancy from the transmission destination's baud rate is allowed during reception is shown below.

Caution: The equations described below should be used to set the baud rate error during reception so that it always is within the allowable error range.

Figure 13-19: Allowable Baud Rate Range During Reception



As shown in Figure 13-19, after the start bit is detected, the receive data latch timing is determined according to the counter that was set by the BRGCm register. If all data up to the final data (stop bit) is in time for this latch timing, the data can be received normally.
Applying this to 11-bit reception is, theoretically, as follows.

$$FL = BR^{-1}$$

BR: UART5n baud rate
k: BRGCm register setting value
FL: 1-bit data length

When the latch timing margin is made 2 basic clocks (Clock), the minimum allowable transfer rate (FLmin) is as follows.

$$\text{FLmin} = 11 \times \text{FL} - \frac{k-2}{2k} \times \text{FL} = \frac{21k+2}{2k} \times FL$$

Therefore, the transfer destination's maximum baud rate (BRmax) that can be received is as follows.

$$\text{BRmax} = \left(\frac{\text{FLmin}}{11}\right)^{-1} = \frac{22k}{21k+2} \times BR$$

Similarly, the maximum allowable transfer rate (FLmax) can be obtained as follows.

$$\frac{10}{11} \times \text{FLmax} = 11 \times \text{FL} - \frac{k+2}{2k} \times FL = \frac{21k-2}{2k} \times FL$$

$$\text{FLmax} = \frac{21k-2}{20k} \times FL \times 11$$

Therefore, the transfer destination's minimum baud rate (BRmin) that can be received is as follows.

$$\text{BRmin} = \left(\frac{\text{FLmax}}{11}\right)^{-1} = \frac{20k}{21k+2} \times BR$$

The allowable baud rate error of UART5n and the transfer destination can be obtained as follows from the expressions described above for computing the minimum and maximum baud rate values.

Table 13-5: Maximum and Minimum Allowable Baud Rate Error

Division Ratio (k)	Maximum Allowable Baud Rate Error	Minimum Allowable Baud Rate Error
8	+3.53%	–3.61%
20	+4.26%	–4.31%
50	+4.56%	–4.58%
100	+4.66%	–4.67%
255	+4.72%	–4.73%

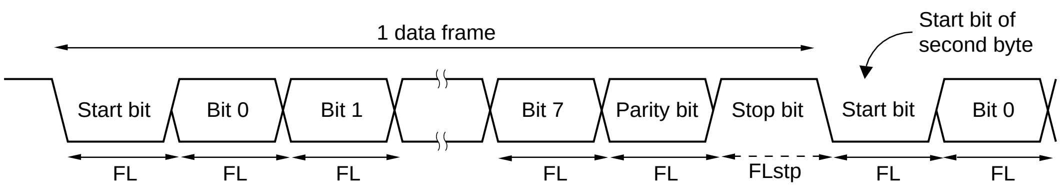
Remarks: 1. The reception precision depends on the number of bits in one frame, the basic clock frequency, and the division ratio (k). The higher the basic clock frequency and the larger the division ratio (k), the higher the precision.

2. k: BRGCm setting value

(4) Transfer rate during continuous transmission

During continuous transmission, the transfer rate from a stop bit to the next start bit is extended two clocks of basic clock (Clock) longer than normal. However, on the reception side, the transfer result is not affected since the timing is initialized by the detection of the start bit.

Figure 13-20: Transfer Rate During Continuous Transmission



Representing the 1-bit data length by FL, the stop bit length by FLstp, and the basic clock frequency by f_{CLK} fields the following equation.

$$FLstp = FL + 2/f_{CLK}$$

Therefore, the transfer rate during continuous transmission is as follows.

$$\text{Transfer rate} = 11 \times FL = 2 / f_{CLK}$$

13.2.7 Precautions

When the supply of clocks to UART5n (n = 0, 1) is stopped (for example, IDLE or STOP mode), operation stops with each register retaining the value it had immediately before the supply of clocks was stopped. The TXD5n pin output also holds and outputs the value it had immediately before the supply of clocks was stopped. However, operation is not guaranteed after the supply of clocks is restarted. Therefore, after the supply of clocks is restarted, the circuits should be initialized by setting Power bit = 0, RXE bit = 0, and TXE bit = 0 in the ASIMn register.

13.3 Clocked Serial Interfaces (CSI00 to CSI02)

13.3.1 Features

- High-speed transfer: Maximum 5 Mbps
- Master mode or slave mode can be selected
- Transmission data length: 8 bits or 16 bits
- Transfer data direction can be switched between MSB first and LSB first
- Eight clock signals can be selected (7 master clocks and 1 slave clock)
- 3-wire type
  - SO0n :Serial transmit data output
  - SI0n :Serial transmit data input
  - $\overline{\text{SCK0n}}$:Serial clock input/output
- Interrupt sources: 1 type
  - Transmission/reception completion interrupt (INTCSI0n)
- Transmission/reception mode and reception-only mode can be specified
- Two transmission buffers (SOTBFn/SOTBFLn, SOTBn/SOTBLn) and two reception buffers (SIRBn/SIRBLn, SIRBEn/SIRBELn) are provided on chip
- Single transfer mode and repeat transfer mode can be specified

Remark: n = 0 to 2

13.3.2 Configuration

CSI0n is controlled via the clocked serial interface mode register (CSIMn) (n = 0 to 2).
Transmission/reception of data is performed with reading SIOn register (n = 0 to 2).

(1) Clocked serial interface mode registers (CSIM0 to CSIM2)

The CSIMn register is an 8-bit register that specifies the operation of CSI0n.

(2) Clocked serial interface clock selection registers (CSIC0 to CSIC2)

The CSICn register is an 8-bit register that controls the CSI0n serial transfer operation.

(3) Serial I/O shift registers (SIO0 to SIO2)

The SIOn register is a 16-bit shift register that converts parallel data into serial data.
The SIOn register is used for both transmission and reception.
Data is shifted in (reception) and shifted out (transmission) from the MSB or LSB side.
The actual transmission/reception operations are started up by access of the buffer register.

(4) Serial I/O shift registers Low (SIOL0 to SIOL2)

The SIOLn register is an 8-bit shift register that converts parallel data into serial data.
The SIOLn register is used for both transmission and reception.
Data is shifted in (reception) and shifted out (transmission) from the MSB or LSB side.
The actual transmission/reception operations are started up by access of the buffer register.

(5) Clocked serial interface reception buffer registers (SIRB0 to SIRB2)

The SIRBn register is a 16-bit buffer register that stores receive data.

(6) Clocked serial interface reception buffer registers Low (SIRBL0 to SIRBL2)

The SIRBLn register is an 8-bit buffer register that stores receive data.

(7) Clocked serial interface read-only reception buffer registers (SIRBE0 to SIRBE2)

The SIRBEn register is a 16-bit buffer register that stores receive data.
The SIRBEn register is the same as the SIRBn register. It is used to read the contents of the SIRBn register.

(8) Clocked serial interface read-only reception buffer registers Low (SIRBEL0 to SIRBEL2)

The SIRBELn register is an 8-bit buffer register that stores receive data.
The SIRBELn register is the same as the lower bytes of the SIRBn register. It is used to read the contents of the SIRBLn register.

(9) Clocked serial interface transmission buffer registers (SOTB0 to SOTB2)

The SOTBn register is a 16-bit buffer register that stores transmit data.

(10) Clocked serial interface transmission buffer registers Low (SOTBL0 to SOTBL2)

The SOTBLn register is an 8-bit buffer register that stores transmit data.

(11) Clocked serial interface initial transmission buffer registers (SOTBF0 to SOTBF2)

The SOTBFn register is a 16-bit buffer register that stores the initial transmit data in the repeat transfer mode.

(12) Clocked serial interface initial transmission buffer registers Low (SOTBFL0 to SOTBFL2)

The SOTBFLn register is an 8-bit buffer register that stores initial transmit data in the repeat transfer mode.

(13) Selector

The selector selects the serial clock to be used.

(14) Serial clock control circuit

Controls the serial clock supply to the shift register. Also controls the clock output to the $\overline{\text{SCK0n}}$ pin when the internal clock is used.

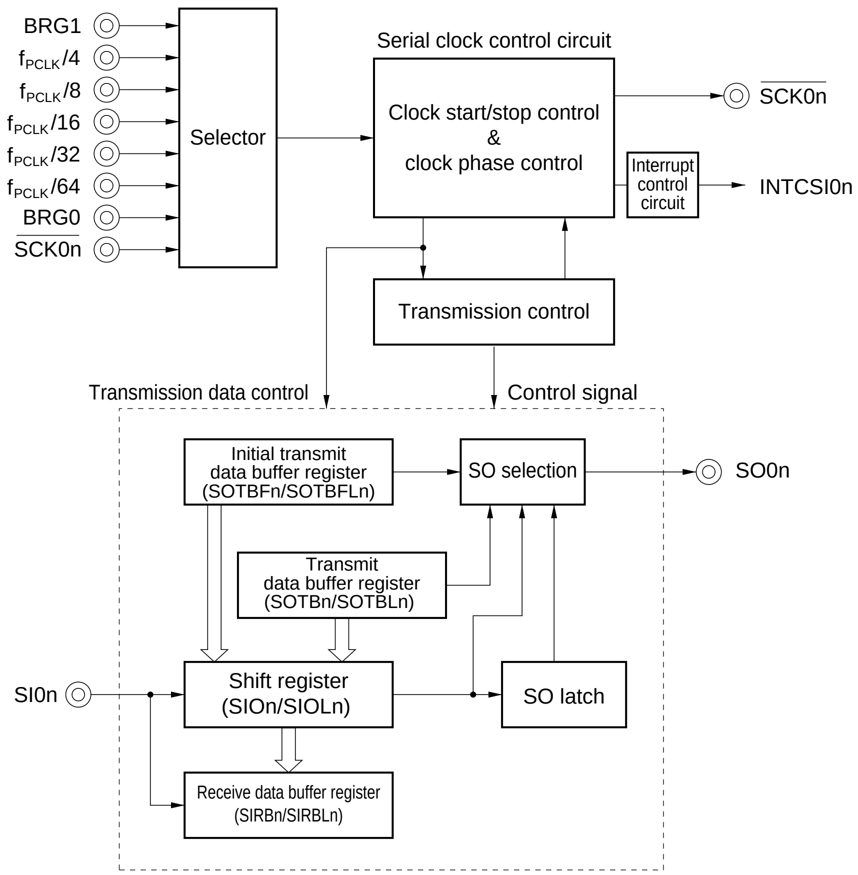
(15) Serial clock counter

Counts the serial clock output or input during transmission/reception operation, and checks whether 8-bit data transmission/reception has been performed.

(16) Interrupt control circuit

Controls the interrupt request timing.

Figure 13-21: Block Diagram of Clocked Serial Interfaces



Remark: n = 0 to 2

13.3.3 Control registers

(1) Clocked serial interface mode registers (CSIM0 to CSIM2)

The CSIMn register controls the CSI0n operation (n = 0 to 2).
These registers can be read/written in 8-bit or 1-bit units (however, bit 0 is read-only).

Figure 13-22: Clocked Serial Interface Mode Registers (CSIM0 to CSIM2)

	7	6	5	4	3	2	1	0	Address	Initial value
CSIM0	CSIE	TRMD	CCL	DIR	CSIT	AUTO	0	CSOT	FFFF FD00H	00H
CSIM1	CSIE	TRMD	CCL	DIR	CSIT	AUTO	0	CSOT	FFFF FD40H	00H
CSIM2	CSIE	TRMD	CCL	DIR	CSIT	AUTO	0	CSOT	FFFF FD80H	00H

Bit Position	Bit Name	Function
7	CSIE	Enables/disables CSI0n operation. 0: Disable CSI0n operation. 1: Enable CSI0n operation. The internal CSI0n circuit can be reset asynchronously by setting the CSIE bit to 0. For the $\overline{\text{SCK0n}}$ and SO0n pin output status when the CSIE bit = 0, refer to 13.3.5 "Output pins" on page 422.
6	TRMD	Specifies transmission/reception mode. 0: Receive-only mode 1: Transmission/reception mode When the TRMD bit = 0, receive-only transfer is performed and the SO0n pin output is fixed to low level. Data reception is started by reading the SIRBn register. When the TRMD bit = 1, transmission/reception is started by writing data to the SOTBn register.
5	CCL	Specifies data length. 0: 8 bits 1: 16 bits
4	DIR	Specifies transfer direction mode (MSB/LSB). 0: First bit of transfer data is MSB 1: First bit of transfer data is LSB
3	CSIT	Controls delay of interrupt request signal. 0: No delay 1: Delay mode (interrupt request signal is delayed 1/2 cycle). **Caution: The delay mode (CSIT bit = 1) is effective only in the master mode (CKS2 to CKS0 bits of the CSICn register are not 111B). In the slave mode (CKS2 to CKS0 bits are 111B), do not set the delay mode.**
2	AUTO	Specifies single transfer mode or repeat transfer mode. 0: single transfer mode 1: Repeat transfer mode
0	CSOT	Flag indicating transfer status. 0: Idle status 1: Transfer execution status **Caution: The CSOT bit is cleared (0) by writing 0 to the CSIE bit.**

Remark: n = 0 to 2

Caution: Overwriting the TRMD, CCL, DIR, CSIT, and AUTO bits of the CSIMn register can be done only when the CSOT bit = 0. If these bits are overwritten at any other time, the operation cannot be guaranteed.

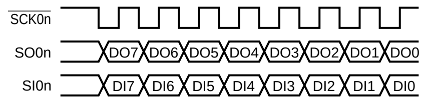
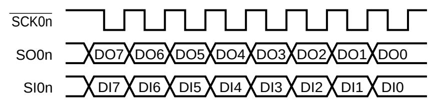
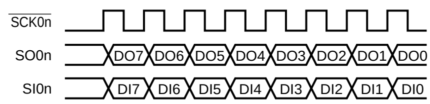
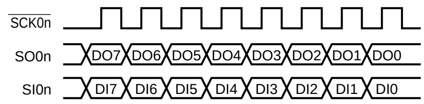
(2) Clocked serial interface clock selection registers (CSIC0 to CSIC2)

The CSICn register is an 8-bit register that controls the CSI0n transfer operation (n = 0 to 2).
This register can be read/written in 8-bit or 1-bit units.

Figure 13-23: Clocked Serial Interface Clock Selection Registers (CSIC0 to CSIC2) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
CSIC0	0	0	0	CKP	DAP	CKS2	CKS1	CKS0	FFFF FD01H	00H
CSIC1	0	0	0	CKP	DAP	CKS2	CKS1	CKS0	FFFF FD41H	00H
CSIC2	0	0	0	CKP	DAP	CKS2	CKS1	CKS0	FFFF FD81H	00H

Bit Position	Bit Name	Function
4, 3	CKP, DAP	Specifies operation mode

CKP	DAP	Operation Mode
0	0	
0	1	
1	0	
1	1	

Remark: n = 0 to 2

Figure 13-23: Clocked Serial Interface Clock Selection Registers (CSIC0 to CSIC2) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>2 to 0</td><td>CKS2 to CKS0</td><td>Specifies input clock
<table>
<tr><th>CKS2</th><th>CKS1</th><th>CKS0</th><th>Input Clock</th><th>Mode</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>f_{PCLK} /4</td><td>Master mode</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Internal BRG Channel 0</td><td>Master mode</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Internal BRG Channel 1</td><td>Master mode</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>f_{PCLK} /8</td><td>Master mode</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>f_{PCLK} /16</td><td>Master mode</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>f_{PCLK} /32</td><td>Master mode</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>f_{PCLK} /64</td><td>Master mode</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>External clock ($\overline{SCK0n}$)</td><td>Slave mode</td></tr>
</table>
Remarks: 1. f_{PCLK}: internal peripheral clock frequency.
2. n = 0 to 2</td></tr>
</table>

Caution: The CSICn register can be overwritten only when the CSIE bit of the CSIMn register = 0.

(3) Clocked serial interface reception buffer registers (SIRB0 to SIRB2)

The SIRBn register is a 16-bit buffer register that stores receive data (n = 0 to 2).
When the receive-only mode is set (TRMD bit of CSIMn register = 0), the reception operation is started by reading data from the SIRBn register.
These registers are read-only, in 16-bit units.
In addition to reset input, these registers can also be initialized by clearing (0) the CSIE bit of the CSIMn register.

Figure 13-24: Clocked Serial Interface Reception Buffer Registers (SIRB0 to SIRB2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIRB0	SIRB15	SIRB14	SIRB13	SIRB12	SIRB11	SIRB10	SIRB9	SIRB8	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFF FD02H	0000H
SIRB1	SIRB15	SIRB14	SIRB13	SIRB12	SIRB11	SIRB10	SIRB9	SIRB8	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFF FD42H	0000H
SIRB2	SIRB15	SIRB14	SIRB13	SIRB12	SIRB11	SIRB10	SIRB9	SIRB8	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFF FD82H	0000H

Bit Position	Bit Name	Function
15 to 0	SIRB15 to SIRB0	Store receive data.

Cautions: 1. Read the SIRBn register only when the 16-bit data length has been set (CCL bit of CSIMn register = 1).

2. When the single transfer mode has been set (AUTO bit of CSIMn register = 0), perform read operation only in the idle state (CSOT bit of CSIMn register = 0). If the SIRBn register is read during data transfer, the data cannot be guaranteed.

(4) Clocked serial interface reception buffer registers Low (SIRBL0 to SIRBL2)

The SIRBLn register is an 8-bit buffer register that stores receive data (n = 0 to 2).
When the receive-only mode is set (TRMD bit of CSIMn register = 0), the reception operation is started by reading data from the SIRBLn register.
These registers are read-only, in 8-bit units.
In addition to reset input, these registers can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIRBLn register is the same as the lower bytes of the SIRBn register.

Figure 13-25: Clocked Serial Interface Reception Buffer Registers Low (SIRBL0 to SIRBL2)

	7	6	5	4	3	2	1	0	Address	Initial value
SIRBL0	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFF FD02H	00H
SIRBL1	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFF FD42H	00H
SIRBL2	SIRB7	SIRB6	SIRB5	SIRB4	SIRB3	SIRB2	SIRB1	SIRB0	FFFF FD82H	00H

Bit Position	Bit Name	Function
7 to 0	SIRB7 to SIRB0	Stores receive data.

Cautions: 1. Read the SIRBLn register only when the 8-bit data length has been set (CCL bit of CSIMn register = 0).

2. When the single transfer mode is set (AUTO bit of CSIMn register = 0), perform read operation only in the idle state (CSOT bit of CSIMn register = 0). If the SIRBLn register is read during data transfer, the data cannot be guaranteed.

(5) Clocked serial interface read-only reception buffer registers (SIRBE0 to SIRBE2)

The SIRBEn register is a 16-bit buffer register that stores receive data (n = 0 to 2).
These registers are read-only, in 16-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIRBEn register is the same as the SIRBn register. It is used to read the contents of the SIRBn register.

Figure 13-26: Clocked Serial Interface Read-Only Reception Buffer Registers (SIRBE0 to SIRBE2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIRBE0	SIRBE15	SIRBE14	SIRBE13	SIRBE12	SIRBE11	SIRBE10	SIRBE9	SIRBE8	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFF FD06H	0000H
SIRBE1	SIRBE15	SIRBE14	SIRBE13	SIRBE12	SIRBE11	SIRBE10	SIRBE9	SIRBE8	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFF FD46H	0000H
SIRBE2	SIRBE15	SIRBE14	SIRBE13	SIRBE12	SIRBE11	SIRBE10	SIRBE9	SIRBE8	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFF FD86H	0000H

Bit Position	Bit Name	Function
15 to 0	SIRBE15 to SIRBE0	Store receive data.

Cautions: 1. The receive operation is not started even if data is read from the SIRBEn register

2. The SIRBEn register can be read only if the 16-bit data length is set (CCL bit of CSIMn register = 1).

(6) Clocked serial interface read-only reception buffer registers Low (SIRBEL0 to SIRBEL2)

The SIRBELn register is an 8-bit buffer register that stores receive data (n = 0 to 2).
These registers are read-only, in 8-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIRBELn register is the same as the lower byte of the SIRBn register. It is used to read the contents of the SIRBLn register.

Figure 13-27: Clocked Serial Interface Read-Only Reception Buffer Registers Low (SIRBEL0 to SIRBEL1)

	7	6	5	4	3	2	1	0	Address	Initial value
SIRBEL0	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFF FD06H	00H
SIRBEL1	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFF FD46H	00H
SIRBEL2	SIRBE7	SIRBE6	SIRBE5	SIRBE4	SIRBE3	SIRBE2	SIRBE1	SIRBE0	FFFF FD86H	00H

Bit Position	Bit Name	Function
7 to 0	SIRBE7 to SIRBE0	Store receive data.

Cautions: 1. The receive operation is not started even if data is read from the SIRBELn register.

2. The SIRBELn register can be read only if the 8-bit data length has been set (CCL bit of CSIMn register = 0).

(7) Clocked serial interface transmission buffer registers (SOTB0 to SOTB2)

The SOTBn register is a 16-bit buffer register that stores transmit data (n = 0 to 2).
When the transmission/reception mode is set (TRMD bit of CSIMn register = 1), the transmission operation is started by writing data to the SOTBn register.
This register can be read/written in 16-bit units.

Figure 13-28: Clocked Serial Interface Transmission Buffer Registers (SOTB0 to SOTB2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTB0	SOTB15	SOTB14	SOTB13	SOTB12	SOTB11	SOTB10	SOTB9	SOTB8	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFF FD04H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTB1	SOTB15	SOTB14	SOTB13	SOTB12	SOTB11	SOTB10	SOTB9	SOTB8	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFF FD44H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTB2	SOTB15	SOTB14	SOTB13	SOTB12	SOTB11	SOTB10	SOTB9	SOTB8	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFF FD84H	0000H

Bit Position	Bit Name	Function
15 to 0	SOTB15 to SOTB0	Store transmit data.

Cautions: 1. Access the SOTBn register only when the 16-bit data length is set (CCL bit of CSIMn register = 1).

2. When the single transfer mode is set (AUTO bit of CSIMn register = 0), perform access only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBn register is accessed during data transfer, the data cannot be guaranteed.

(8) Clocked serial interface transmission buffer registers Low (SOTBL0 to SOTBL2)

The SOTBLn register is an 8-bit buffer register that stores transmit data (n = 0 to 2).
When the transmission/reception mode is set (TRMD bit of CSIMn register = 1), the transmission operation is started by writing data to the SOTBLn register.
These registers can be read/written in 8-bit units.
The SOTBLn register is the same as the lower bytes of the SOTBn register.

Figure 13-29: Clocked Serial Interface Transmission Buffer Registers Low (SOTBL0 to SOTBL2)

	7	6	5	4	3	2	1	0	Address	Initial value
SOTBL0	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFF FD04H	00H
SOTBL1	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFF FD44H	00H
SOTBL2	SOTB7	SOTB6	SOTB5	SOTB4	SOTB3	SOTB2	SOTB1	SOTB0	FFFF FD84H	00H

Bit Position	Bit Name	Function
7 to 0	SOTB7 to SOTB0	Store transmit data.

Cautions: 1. Access the SOTBLn register only when the 8-bit data length has been set (CCL bit of CSIMn register = 0).

2. When the single transfer mode is set (AUTO bit of CSIMn register = 0), perform access only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBLn register is accessed during data transfer, the data cannot be guaranteed.

(9) Clocked serial interface initial transmission buffer registers (SOTBF0 to SOTBF2)

The SOTBFn register is a 16-bit buffer register that stores initial transmission data in the repeat transfer mode (n = 0 to 2).
The transmission operation is not started even if data is written to the SOTBFn register.
These registers can be read/written in 16-bit units.

Figure 13-30: Clocked Serial Interface Initial Transmission Buffer Registers (SOTBF0 to SOTBF2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SOTBF0	SOTBF15	SOTBF14	SOTBF13	SOTBF12	SOTBF11	SOTBF10	SOTBF9	SOTBF8	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFF FD08H	0000H
SOTBF1	SOTBF15	SOTBF14	SOTBF13	SOTBF12	SOTBF11	SOTBF10	SOTBF9	SOTBF8	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFF FD48H	0000H
SOTBF2	SOTBF15	SOTBF14	SOTBF13	SOTBF12	SOTBF11	SOTBF10	SOTBF9	SOTBF8	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFF FD88H	0000H

Bit Position	Bit Name	Function
15 to 0	SOTBF15 to SOTBF0	Stores initial transmission data in repeat transfer mode.

Caution: Access the SOTBFn register only when the 16-bit data length has been set (CCL bit of CSIMn register = 1), and only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBFn register is accessed during data transfer, the data cannot be guaranteed.

(10) Clocked serial interface initial transmission buffer registers Low (SOTBFL0 to SOTBFL2)

The SOTBFLn register is an 8-bit buffer register that stores initial transmission data in the repeat transfer mode (n = 0 to 2).
The transmission operation is not started even if data is written to the SOTBFLn register.
These registers can be read/written in 8-bit units.
The SOTBFLn register is the same as the lower bytes of the SOTBFn register.

Figure 13-31: Clocked Serial Interface Initial Transmission Buffer Registers Low (SOTBFL0 to SOTBFL2)

	7	6	5	4	3	2	1	0	Address	Initial value
SOTBFL0	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFF FD08H	00H
SOTBFL1	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFF FD48H	00H
SOTBFL2	SOTBF7	SOTBF6	SOTBF5	SOTBF4	SOTBF3	SOTBF2	SOTBF1	SOTBF0	FFFF FD88H	00H

Bit Position	Bit Name	Function
7 to 0	SOTBF7 to SOTBF0	Store initial transmission data in repeat transfer mode.

Caution: Access the SOTBFLn register only when the 8-bit data length has been set (CCL bit of CSIM0 register = 0), and only in the idle state (CSOT bit of CSIMn register = 0). If the SOTBFLn register is accessed during data transfer, the data cannot be guaranteed.

(11) Serial I/O shift registers (SIO0 to SIO2)

The SIOn register is a 16-bit shift register that converts parallel data into serial data (n = 0 to 2).
The transfer operation is not started even if the SIOn register is read.
These registers are read-only, in 16-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.

Figure 13-32: Serial I/O Shift Registers (SIO0 to SIO2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
SIO0	SIO15	SIO14	SIO13	SIO12	SIO11	SIO10	SIO9	SIO8	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFF FD0AH	0000H
SIO1	SIO15	SIO14	SIO13	SIO12	SIO11	SIO10	SIO9	SIO8	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFF FD4AH	0000H
SIO2	SIO15	SIO14	SIO13	SIO12	SIO11	SIO10	SIO9	SIO8	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFF FD8AH	0000H

Bit Position	Bit Name	Function
15-0	SIO15 to SIO0	Data is shifted in (reception) or shifted out (transmission) from the MSB or LSB side.

Caution: Access the SIOn register only when the 16-bit data length has been set (CCL bit of CSIMn register = 1), and only in the idle state (CSOT bit of CSIMn register = 0). If the SIOn register is accessed during data transfer, the data cannot be guaranteed.

(12) Serial I/O shift registers Low (SIOL0 to SIOL2)

The SIOLn register is an 8-bit shift register that converts parallel data into serial data (n = 0 to 2).
The transfer operation is not started even if the SIOLn register is read.
These registers are read-only, in 8-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE bit of the CSIMn register.
The SIOLn register is the same as the lower bytes of the SIOn register.

Figure 13-33: Serial I/O Shift Registers Low (SIOL0 to SIOL2)

	7	6	5	4	3	2	1	0	Address	Initial value
SIOL0	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFF FD0AH	00H
SIOL1	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFF FD4AH	00H
SIOL2	SIO7	SIO6	SIO5	SIO4	SIO3	SIO2	SIO1	SIO0	FFFF FD8AH	00H

Bit Position	Bit Name	Function
7 to 0	SIO7 to SIO0	Data is shifted in (reception) or shifted out (transmission) from the MSB or LSB side.

Caution: Access the SIOLn register only when the 8-bit data length has been set (CCL bit of CSIMn register = 0), and only in the idle state (CSOT bit of CSIMn register = 0). If the SIOLn register is accessed during data transfer, the data cannot be guaranteed.

13.3.4 Operation

(1) Single transfer mode

(a) Usage

In the receive-only mode (TRMD bit of CSIMn register = 0), transfer is started by reading[Note 1] the receive data buffer register (SIRBn/SIRBLn) (n = 0 to 2).
In the transmission/reception mode (TRMD bit of CSIMn register = 1), transfer is started by writing[Note 2] to the transmit data buffer register (SOTBn/SOTBLn).
In the slave mode, the operation must be enabled beforehand (CSIE bit of CSIMn register = 1).
When transfer is started, the value of the CSOT bit of the CSIMn register becomes 1 (transmission execution status).
Upon transfer completion, the transmission/reception completion interrupt (INTCSI0n) is set (1), and the CSOT bit is cleared (0). The next data transfer request is then waited for.

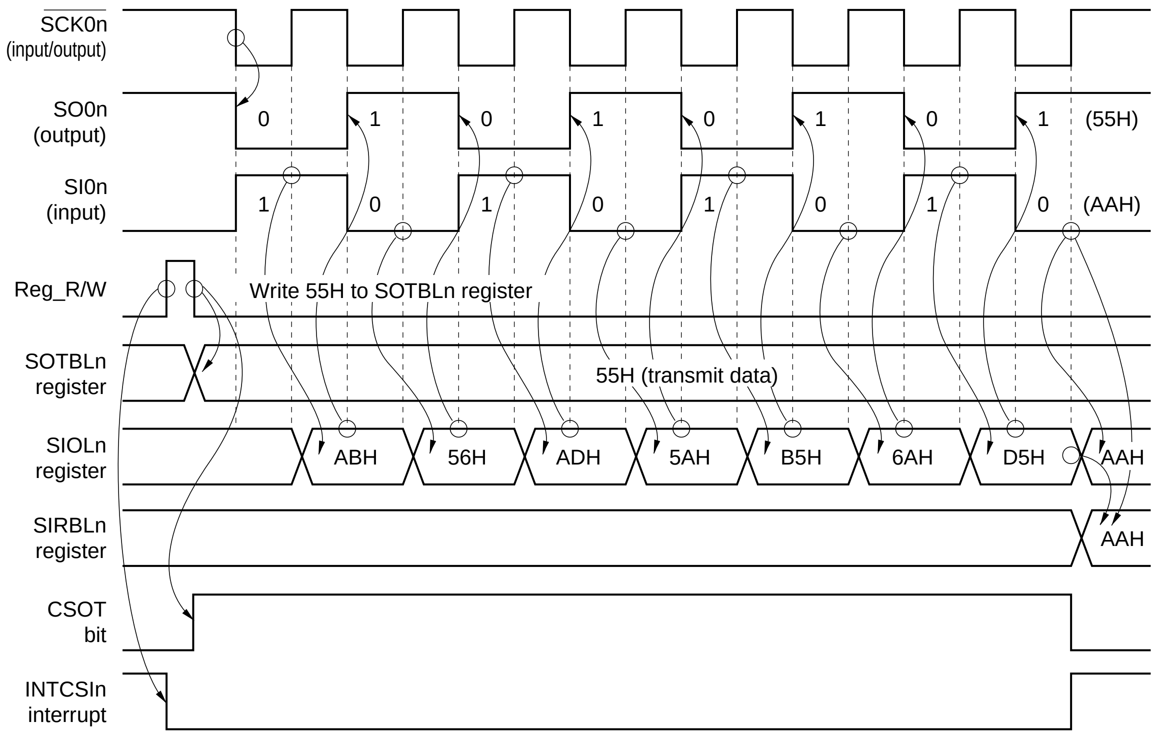
Notes: 1. When the 16-bit data length (CCL bit of CSIMn register = 1) has been set, read the SIRBn register. When the 8-bit data length (CCL bit of CSIMn register = 0) has been set, read the SIRBLn register.

2. When the 16-bit data length (CCL bit of CSIMn register = 1) has been set, write to the SOTBn register. When the 8-bit data length (CCL bit of CSIMn register = 0) has been set, write to the SOTBLn register.

Caution: When the CSOT bit of the CSIMn register = 1, do not manipulate the CSI0n register.

Figure 13-34: Timing Chart in single Transfer Mode (1/2)

(a) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKP bit = 0, DAP bit = 0

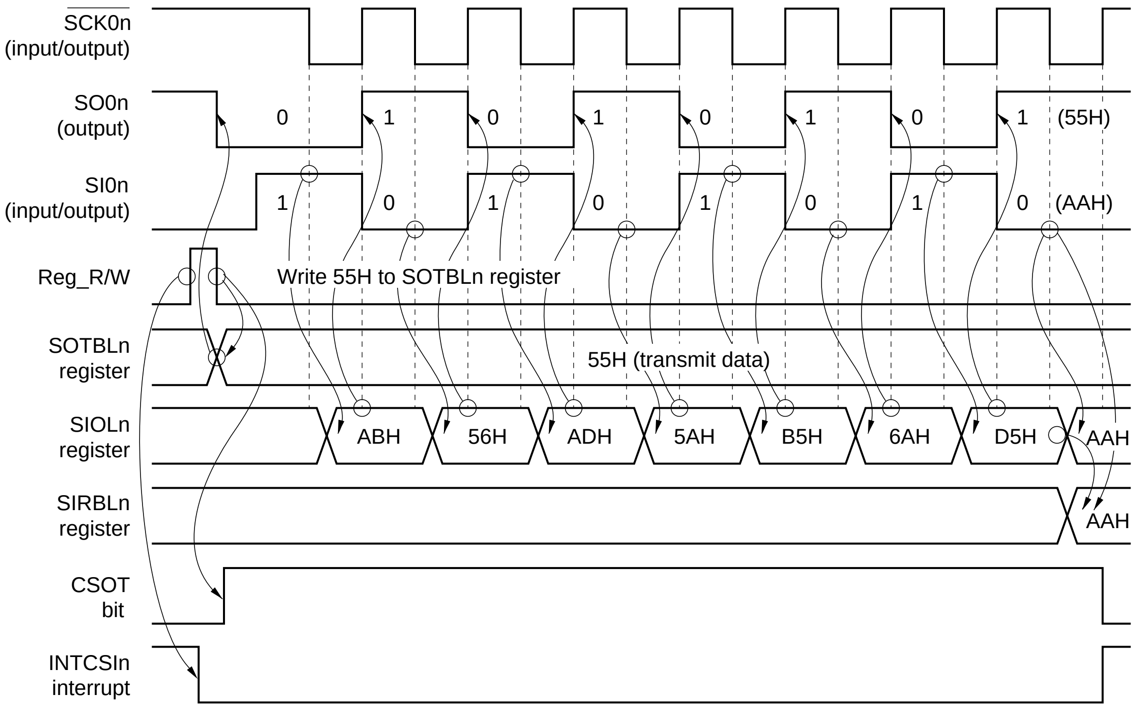


Remarks: 1. n = 0 to 2

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/ SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

Figure 13-34: Timing Chart in single Transfer Mode (2/2)

(b) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKP bit = 0, DAP bit = 1



Remarks: 1. n = 0 to 2

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

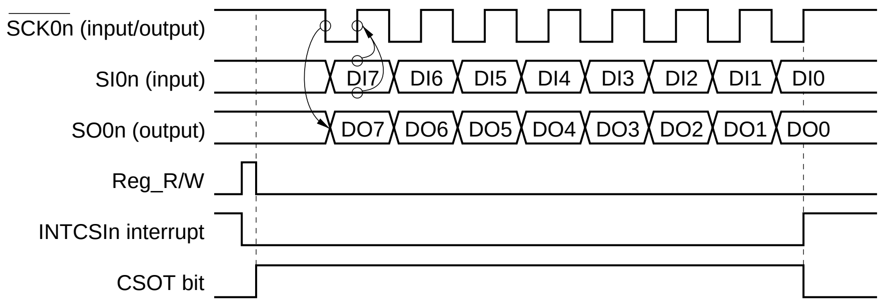
(b) Clock phase selection

The following shows the timing when changing the conditions for clock phase selection (CKP bit of CSICn register) and data phase selection (DAP bit of CSICn register) under the following conditions.

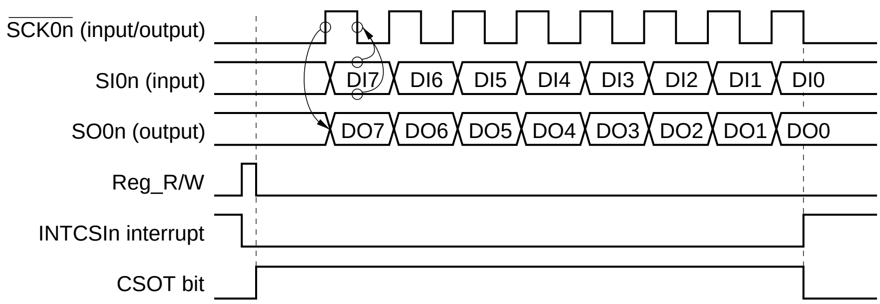
- Data length = 8 bits (CCL bit of CSIMn register = 0)
- First bit of transfer data = MSB (DIR bit of CSIMn register = 0)
- No interrupt request signal delay control (CSIT bit of CSIMn register = 0)

Figure 13-35: Timing Chart According to Clock Phase Selection (1/2)

(a) When CKP bit = 0, DAP bit = 0



(b) When CKP bit = 1, DAP bit = 0

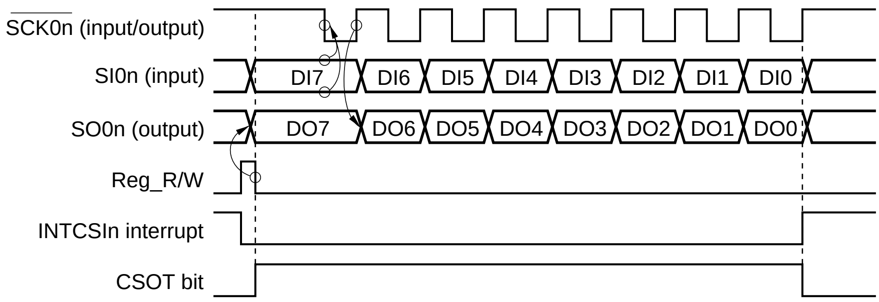


Remarks: 1. n = 0 to 2

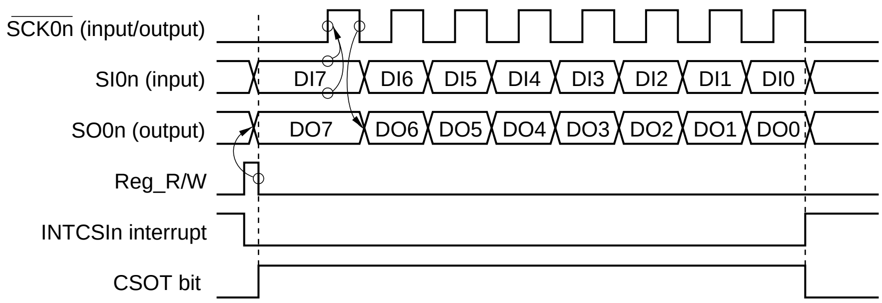
2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

Figure 13-35: Timing Chart According to Clock Phase Selection (2/2)

(c) When CKP bit = 0, DAP bit = 1



(d) When CKP bit = 1, DAP bit = 1



Remarks: 1. n = 0 to 2

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/ SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

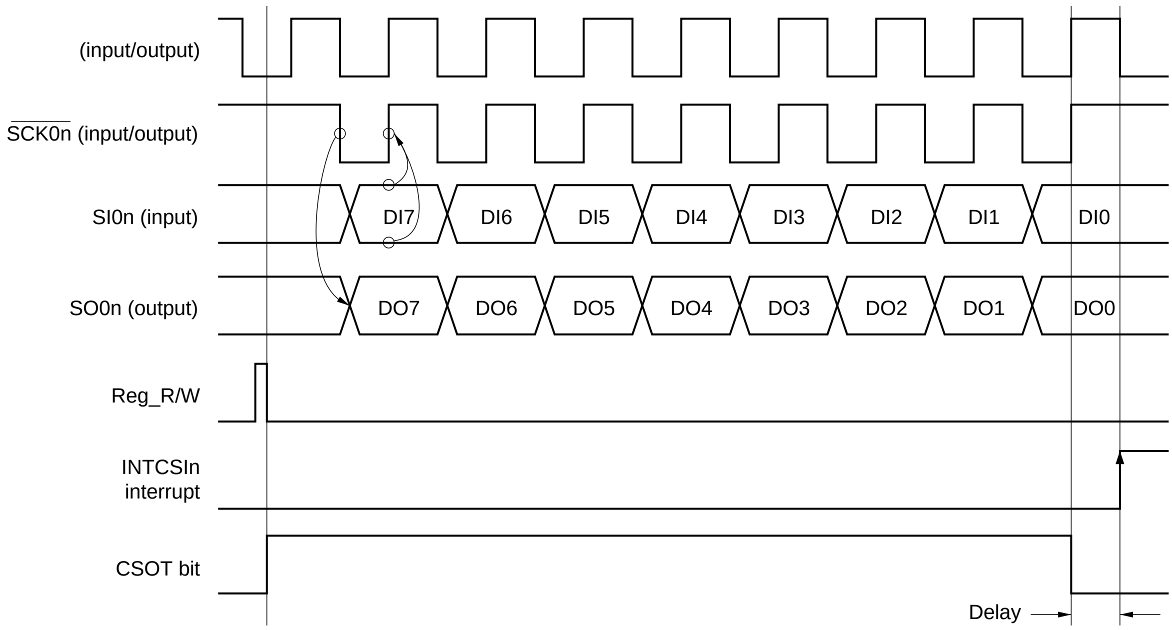
(c) Transmission/reception completion interrupt request signals (INTCSI0 to INTCSI2)

INTCSI0n is set (1) upon completion of data transmission/reception.

Caution: The delay mode (CSIT bit = 1) is valid only in the master mode (bits CKS2 to CKS0 of the CSICn register are not 111B). The delay mode cannot be set when the slave mode is set (bits CKS2 to CKS0 = 111B).

Figure 13-36: Timing Chart of Interrupt Request Signal Output in Delay Mode (1/2)

(a) When CKP bit = 0, DAP bit = 0

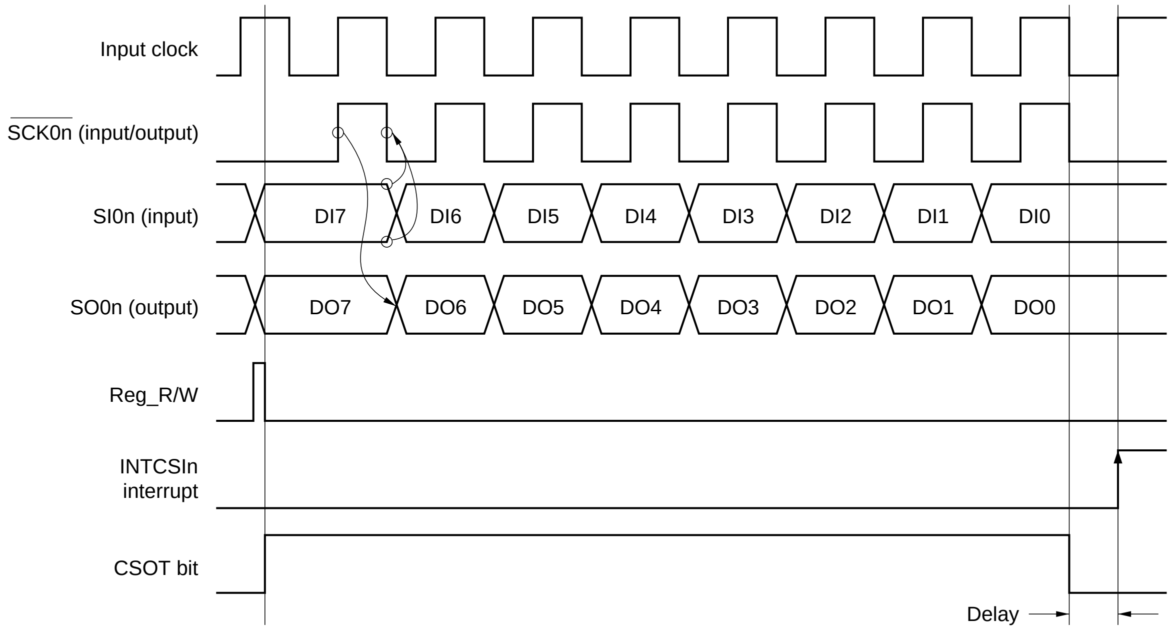


Remarks: 1. n = 0 to 2

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

Figure 13-36: Timing Chart of Interrupt Request Signal Output in Delay Mode (2/2)

(b) When CKP bit = 1, DAP bit = 1



Remarks: 1. n = 0 to 2

2. Reg_R/W:Internal signal. This signal indicates that receive data buffer register (SIRBn/SIRBLn) read or transmit data buffer register (SOTBn/SOTBLn) write was performed.

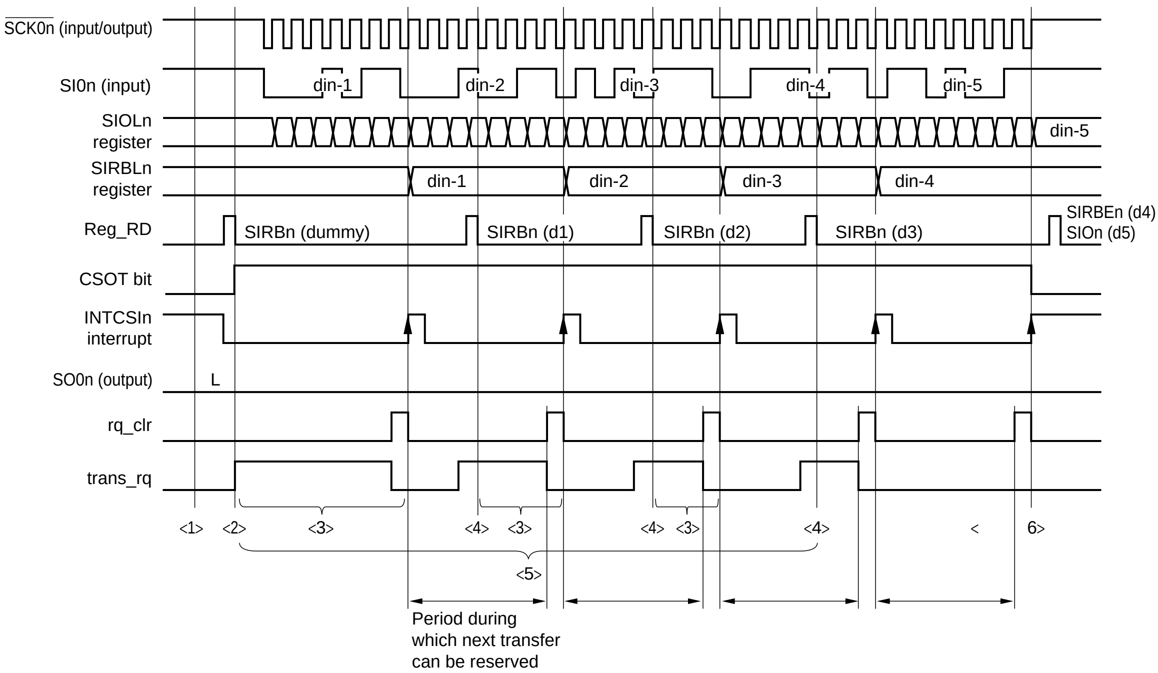
(2) Repeat transfer mode

(a) Usage (receive-only)

<1> Set the repeat transfer mode (AUTO bit of CSIMn register = 1) and the receive-only mode (TRMD bit of CSIMn register = 0).
<2> Read SIRBn register (start transfer with dummy read).
<3> Wait for transmission/reception completion interrupt request (INTCSI0n).
<4> When the transmission/reception completion interrupt request (INTCSI0n) has been set to (1), read the SIRBn register[Note] (reserve next transfer).
<5> Repeat steps <3> and <4> (n - 2) times (n: number of transfer data).
<6> Following output of the last transmission/reception completion interrupt request (INTCSI0n), read the SIRBn register and the SIOn register[Note].

Note: When transferring n number of data, receive data is loaded by reading the SIRBn register from the first data to the (n - 2)-th data. The (n-1)-th data is loaded by reading the SIRBEn register, and the n-th (last) data is loaded by reading the SIOn register.

Figure 13-37: Repeat Transfer (Receive-Only) Timing Chart



Remarks: 1. n = 0 to 2

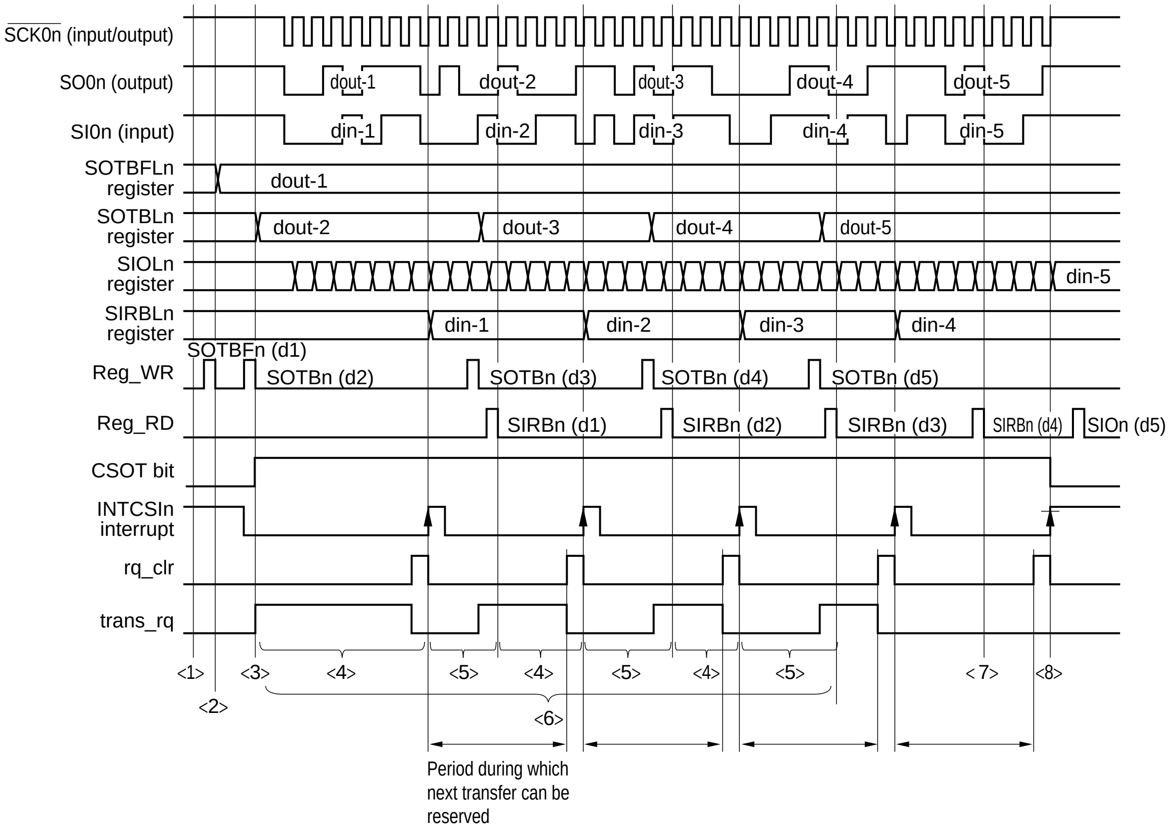
2. Reg_RD:Internal signal. This signal indicates that the receive data buffer register (SIRBn/SIRBLn) has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

In the case of the repeat transfer mode, two transfer requests are set at the start of the first transfer. Following the transmission/reception completion interrupt request (INTCSI0n), transfer is continued if the SIRBn register can be read within the next transfer reservation period. If the SIRBn register cannot be read, transfer ends and the SIRBn register does not receive the new value of the SIOn register.
The last data can be obtained by reading the SIOn register following completion of the transfer.

(b) Usage (transmission/reception)

<1> Set the repeat transfer mode (AUTO bit of CSIMn register = 1) and the transmission/reception mode (TRMD bit of CSIMn register = 1).
<2> Write the first data to the SOTBFn register.
<3> Write the 2nd data to the SOTBn register (start transfer).
<4> Wait for transmission/reception completion interrupt request (INTCSI0n).
<5> When the transmission/reception completion interrupt request (INTCSI0n) has been set to (1), write the next data to the SOTBn register (reserve next transfer), and read the SIRBn register to load the receive data.
<6> Repeat steps <4> and <5> as long as data to be sent remains.
<7> Wait for the INTCSI0n interrupt. When the interrupt request signal is set to (1), read the SIRBn register to load the (n - 1)-th receive data.
<8> Following the last transmission/reception completion interrupt request (INTCSI0n), read the SIOn register to load the n-th (last) receive data.

Figure 13-38: Repeat Transfer (Transmission/Reception) Timing Chart



Remarks: 1. n = 0 to 2

2. Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBLn) has been written.
Reg_RD:Internal signal. This signal indicates that the receive data buffer register (SIRBn/SIRBLn) has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

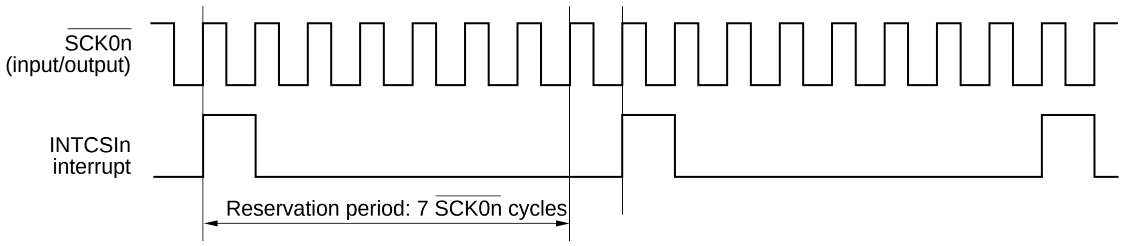
In the case of the repeat transfer mode, two transfer requests are set at the start of the first transfer. Following the transmission/reception completion interrupt request (INTCSI0n), transfer is continued if the SOTBn register can be written within the next transfer reservation period. If the SOTBn register cannot be written, transfer ends and the SIRBn register does not receive the new value of the SIOn register. The last receive data can be obtained by reading the SIOn register following completion of the transfer.

(c) Next transfer reservation period

In the repeat transfer mode, the next transfer must be prepared with the period shown in Figure 13-39.

Figure 13-39: Timing Chart of Next Transfer Reservation Period (1/2)

(a) When data length: 8 bits, operation mode: CKP bit = 0, DAP bit = 0



(b) When data length: 16 bits, operation mode: CKP bit = 0, DAP bit = 0

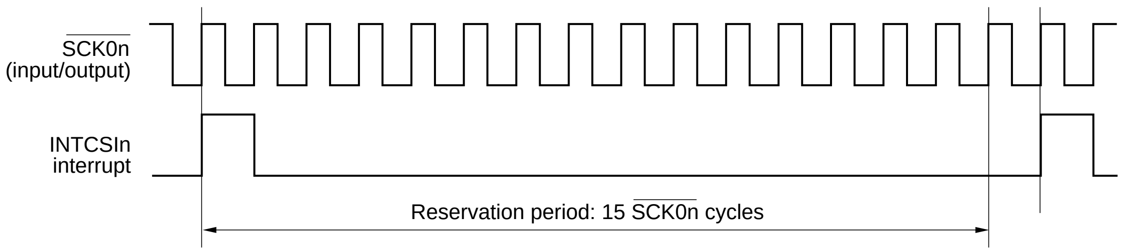
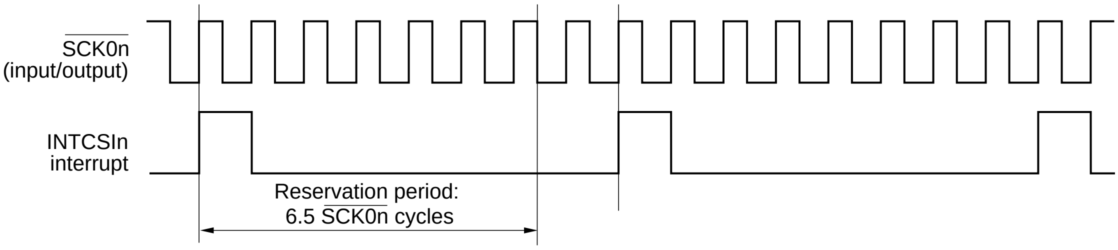
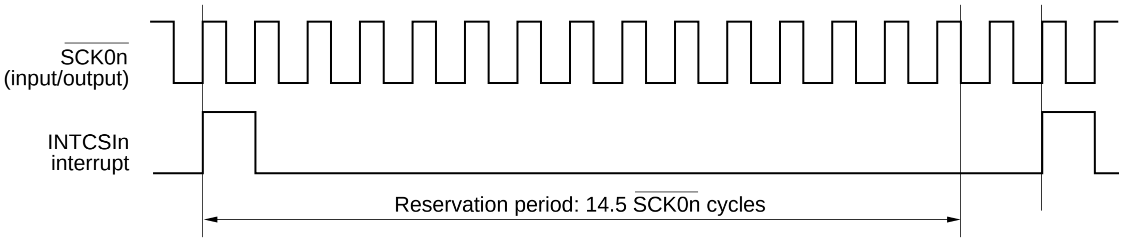


Figure 13-39: Timing Chart of Next Transfer Reservation Period (2/2)

(c) When data length: 8 bits, operation mode: CKP bit = 0, DAP bit = 1



(d) When data length: 16 bits, operation mode: CKP bit = 0, DAP bit = 1



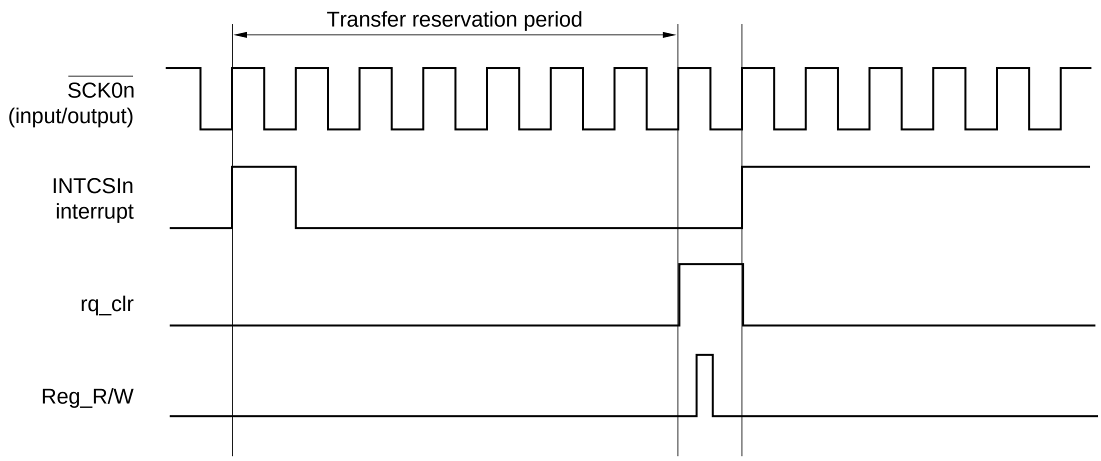
Remark: n = 0 to 2

(d) Cautions

To continue repeat transfers, it is necessary to either read the SIRBn register or write to the SOTBn register during the transfer reservation period.
If access is performed to the SIRBn register or the SOTBn register when the transfer reservation period is over, the following occurs.

- In case of contention between transfer request clear and register access
Since request cancellation has higher priority, the next transfer request is ignored. Therefore, transfer is interrupted, and normal data transfer cannot be performed.

Figure 13-40: Transfer Request Clear and Register Access Contention

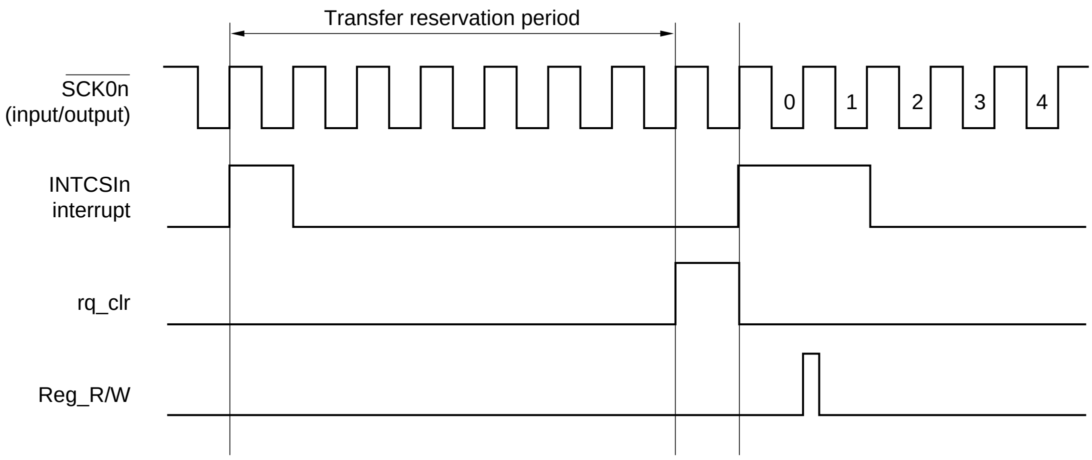


Remarks: 1. n = 0 to 2

2. rq_clr: Internal signal. Transfer request clear signal.
Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBLn) has been written.

- In case of contention between interrupt request and register access
Since continuous transfer has stopped once, executed as a new repeat transfer.
In the slave mode, a bit phase error transfer error results (refer to Figure 13-41).
In the transmission/reception mode, the value of the SOTBFn register is retransmitted, and illegal data is sent.

Figure 13-41: Interrupt Request and Register Access Contention



Remarks: 1. n = 0 to 2

2. rq_clr: Internal signal. Transfer request clear signal.
Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBLn) has been written.

13.3.5 Output pins

(1) $\overline{\text{SCK0n}}$ pin

When the CSI0n operation is disabled (CSIE bit of CSIMn register = 0), the $\overline{\text{SCK0n}}$ pin output status is as follows (n = 0 to 2).

CKP	CKS2	CKS1	CKS0	$\overline{\text{SCK0n}}$ Pin Output
0	Don't care	Don't care	Don't care	Fixed to high level
1	1	1	1	Fixed to high level
	Other than above			Fixed to low level

Remarks: 1. n = 0 to 2

2. When any of bits CKP and CKS2 to CKS0 of the CSICn register is overwritten, the $\overline{\text{SCK0n}}$ pin output changes.

(2) SO0n pin

When the CSI0n operation is disabled (CSIE bit of CSIMn register = 0), the SO0n pin output status is as follows (n = 0 to 2).

TRMD	DAP	AUTO	CCL	DIR	SO0n Pin Output
0	Don't care	Don't care	Don't care	Don't care	Fixed at low level
1	0	Don't care	Don't care	Don't care	SO latch value (low level)
	1	0	0	0	SOTB7 value
				1	SOTB0 value
			1	0	SOTB15 value
				1	SOTB0 value
		1	0	0	SOTBF7 value
				1	SOTBF0 value
			1	0	SOTBF15 value
				1	SOTBF0 value

Remarks: 1. When any of bits TRMD, CCL, DIR, AUTO, and CSICn of the CSIMn register or DAP bit of the CSICn register is overwritten, the SO0n pin output changes.

2. SOTBm: Bit m of SOTBn register (m = 0, 7, 15)

3. SOTBFm: Bit m of SOTBFn register (m = 0, 7, 15)

4. n = 0 to 2

13.3.6 Dedicated baud rate generators 0, 1 (BRG0, BRG1)

(1) Selecting the baud rate generator

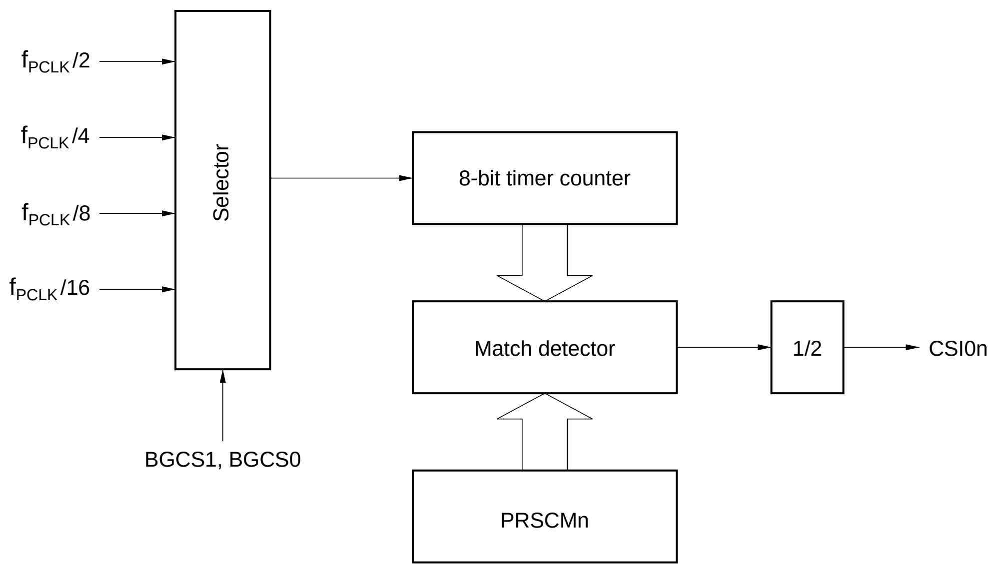
The CSI00 to CSI02 serial clocks can be selected between dedicated baud rate generator output or internal peripheral clock (f_{PCLK}).

The serial clock source is specified by bits CKS2 to CKS0 of registers CSIC0 and CSIC1 (refer to 12.3.3 (2) Clocked serial interface clock selection registers 0, 1 (CSIC0, CSIC1)).

If the dedicated baud rate generator output is specified, BRG0 or BRG1 respectively is selected as the clock source.

Since the same serial clock can be shared for transmission and reception, baud rate is the same for the transmission/reception.

Figure 13-42: Baud Rate Generators 0, 1 (BRG0, BRG1) Block Diagram



Remarks: 1. f_{PCLK}: internal peripheral clock

2. n = 0 to 2

(2) Configuration

BRGn is configured of an 8-bit timer counter that generates the baud rate signal, a prescaler mode register n (PRSMn) that controls baud rate signal generation, a prescaler compare register n (PRSCMn) that sets the value of the 8-bit timer counter, and a prescaler (n = 0 to 2).

(a) Input clock

The internal peripheral clock (f_{PCLK}) is input to BRGn.

(b) Prescaler mode registers 0, 1 (PRSM0, PRSM1)

The PRSMn register controls the generation of the CSI00 to CSI02 baud rate signals respectively. This register can be read/written in 8-bit or 1-bit units (n = 0 to 2).

Figure 13-43: Prescaler Mode Registers 0, 1 (PRSM0, PRSM1)

	7	6	5	4	3	2	1	0	Address	Initial value
PRSM0	0	0	0	CE	0	0	BGCS1	BGCS0	FFFF FDC0H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
PRSM1	0	0	0	CE	0	0	BGCS1	BGCS0	FFFF FDE0H	00H

Bit Position	Bit Name	Function
4	CE	Enables baud rate counter operation. 0: Stop baud rate counter operation and fix baud rate output signal to 0. 1: Enable baud rate counter operation and start baud rate output operation.
1, 0	BGCS1, BGCS0	Selects count clock for baud rate counter. (see table below) **Remarks: 1.** f_{PCLK}: internal peripheral clock.

BGCS1	BGCS0	Count Clock Selection
0	0	$f_{PCLK}/2$
0	1	$f_{PCLK}/4$
1	0	$f_{PCLK}/8$
1	1	$f_{PCLK}/16$

Cautions: 1. Do not change the value of the BGCS1, BGCS0 bits during transmission/ reception operation.

2. Set the PRSMn register prior to setting the CE bit to 1.

(c) Prescaler compare registers 0, 1 (PRSCM0, PRSCM1)

PRSCMn is an 8-bit compare register that sets the value of the 8-bit timer counter.
This register can be read/written in 8-bit or 1-bit units (n = 0 to 2).

Figure 13-44: Prescaler Compare Registers 0, 1 (PRSCM0, PRSCM1)

	7	6	5	4	3	2	1	0	Address	Initial value
PRSCM0	PRSCM7	PRSCM6	PRSCM5	PRSCM4	PRSCM3	PRSCM2	PRSCM1	PRSCM0	FFFF FDC1H	00H

	7	6	5	4	3	2	1	0	Address	Initial value
PRSCM1	PRSCM7	PRSCM6	PRSCM5	PRSCM4	PRSCM3	PRSCM2	PRSCM1	PRSCM0	FFFF FDE1H	00H

Bit Position	Bit Name	Function
7 to 0	PRSCM7 to PRSCM0	Compare value of the 8-bit timer counter.

Cautions: 1. The internal timer counter is cleared by writing to the PRSMn register. Therefore, do not write to the PRSCMn register during transmission.

2. Set the PRSCMn register prior to setting the CE bit of the PRSMn register to 1. If the contents of the PRSCMn register are overwritten when the value of the CE bit is 1, the cycle of the baud rate signal is not guaranteed.

(d) Baud rate signal cycle

The baud rate signal cycle is calculated as follows.

- **When setting value of PRSCMn register is 00H**

(Cycle of signal selected with bits BGCS1, BGCS0 of PRSMn register) / 256 × 2

- **In cases other than above**

(Cycle of signal selected with bits BGCS1, BGCS2 of PRSMn register) / (setting value of PRSCMn register) × 2.

(e) Baud rate setting example

Table 13-6: Baud Rate Generator Setting Data

<1> When f_{PCLK} = 16 MHz

BGCS1	BGCS0	PRSCM Register Value	Clock (Hz)
0	0	1	4000000
0	0	2	2000000
0	0	4	1000000
0	0	8	500000
0	0	16	250000
0	0	40	100000
0	0	80	50000
0	0	160	25000
0	1	200	10000
1	0	200	5000

<2> When f_{PCLK} = 20 MHz

BGCS1	BGCS0	PRSCM Register Value	Clock (Hz)
0	0	2	2500000
0	0	5	1000000
0	0	10	500000
0	0	20	250000
0	0	50	100000
0	0	100	50000
0	0	200	25000
0	1	250	10000
1	0	250	5000

Caution: Set the transfer clock so that it does not fall below the minimum value of 200 ns of the $\overline{\text{SCK0n}}$ cycle (t_{CYSK1}) prescribed in the electrical specifications.

Chapter 14 FCAN Interface Function

14.1 Features

- Active support of extended format (ISO 11898, former CAN specification version 2.0B active), supporting transmission and reception of standard and extended frame format messages
- 2 or 4**Note** CAN modules
- CAN bus speed up to 1 Mbit per second
- Direct message storage for minimum CPU burden
- Configurable number of message buffers per CAN module
- 32 message buffers in total
- Mask option for receive messages (BasicCAN channels)
- 4 masks per CAN module (each mask can be assigned to each message)
- Buffered reception (FIFO)
- Message buffers can be redefined in normal operation mode
- FCAN interface and CPU share common RAM area
- Interrupt on receive, transmit and error condition
- Time stamp and global time system function
- Two power-save modes
  - SLEEP mode: wake-up at CAN bus activity
  - STOP mode: no wake-up at CAN bus activity
- Diagnostic features
  - Readable error counters
  - CAN bus status information register
  - Receive-only mode (e.g. used for automatic bit rate detection)
  - Bus error cause information

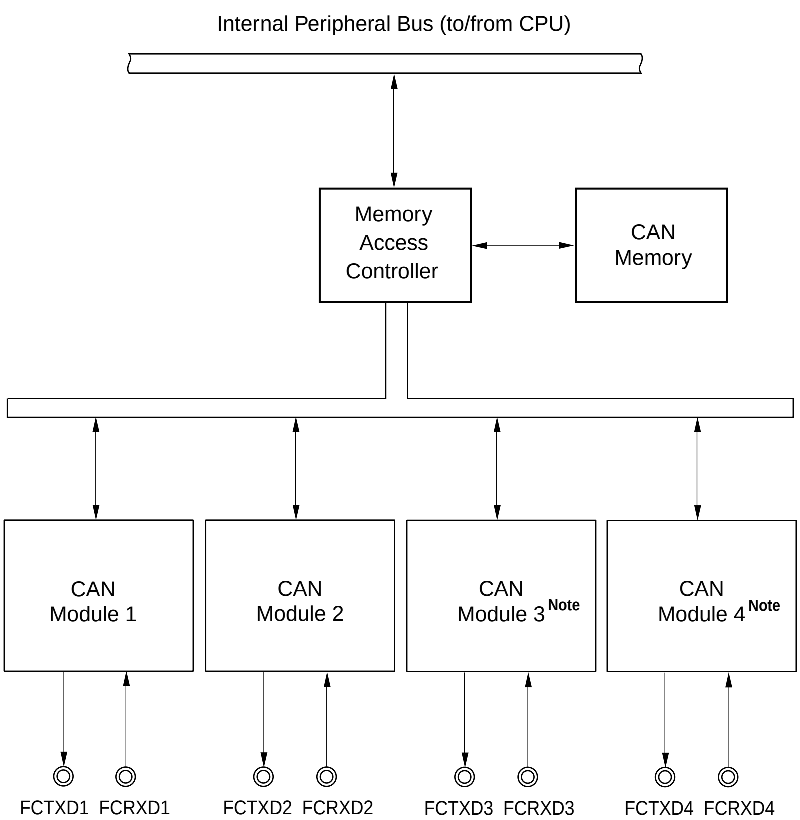
Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

14.2 Outline of the FCAN System

14.2.1 General

The FCAN (Full-CAN) system of the V850E/CA2 supports 2 or 4[Note] independent CAN modules (CAN module 1, CAN module 2, CAN module 3[Note], CAN module 4[Note]), which provide each an interface to a Controller Area Network (CAN).
The CAN modules are conform to ISO 11898, former CAN specification version 2.0B active.
An external bus transceiver has to be used to connect a CAN module to a CAN bus. That external bus transceiver converts the transmit data line and receive data line signals to the necessary electrical signal characteristic on the CAN bus itself.
All protocol activities in a CAN module are handled by hardware (transfer layer).
The CAN modules themselves provide no memory for the necessary data buffers, rather all CAN modules have access to the common CAN memory area via a memory access controller (MAC).
The MAC allows integration of machines other than CAN modules (e.g. CAN bridge). The CPU also accesses to the common CAN memory via the MAC. The MAC offers data scan capability beside controlling the arbitration of CAN modules or CPU accesses to the CAN memory.
By means of that scan capability inner priority inversions at message transmissions are automatically avoided and received messages are sorted into the corresponding receive message buffers according to an inner storage priority rule.

Figure 14-1: Functional Blocks of the FCAN Interface

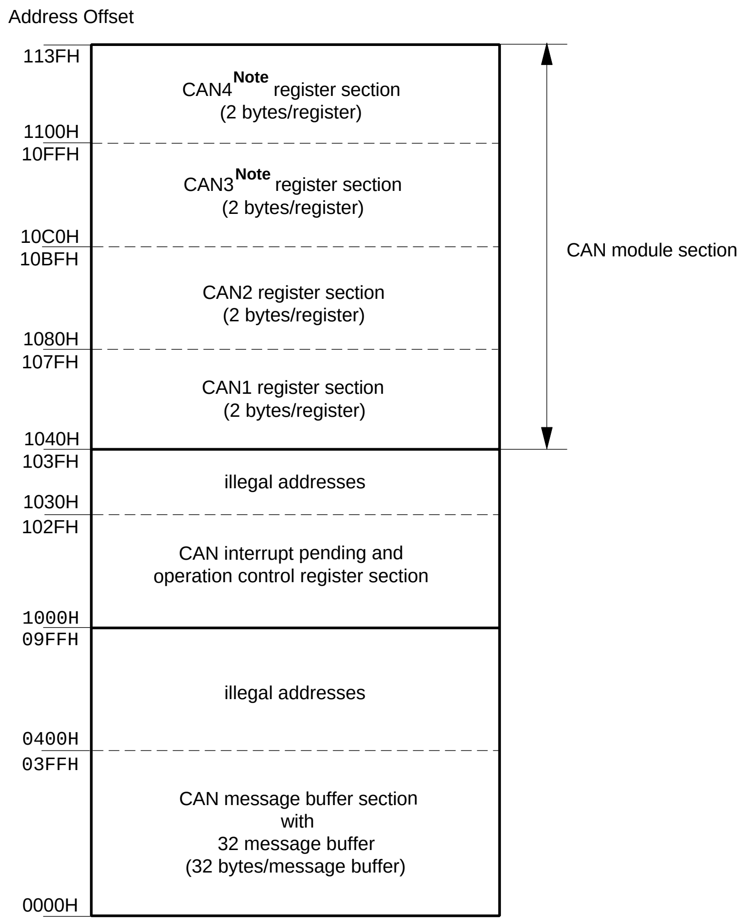


Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

14.2.2 CAN memory and register layout

All buffers and registers of the FCAN system are arranged within a memory layout of 4.5 KB.

Figure 14-2: Memory Area of the FCAN System



Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Remarks:

1. Effective address = PP_BASE + address offset

2. The memory area is located in the 16 KB programmable peripheral I/O area of the V850E/CA2. The base address (PP_BASE) of the programmable peripheral I/O area is set by the BPC register.

3. The memory area of the FCAN system is divided into certain functional sections. The start and end addresses of those sections are given as an address offset value.

Caution: Before accessing any register or buffer of the FCAN system the base address PP_BASE must be fixed by the BPC register.

The sections within the FCAN memory layout contain areas, which are defined as illegal addresses or CANx temporary buffer (x = 1 to 2 for the derivative µPD703128 (A), x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1)).

Remarks: 1. Areas defined as illegal addresses contain neither FCAN registers nor FCAN buffers. Those area must not be read nor written by user program.

2. CANx temporary buffers can be accessed by CPU (write and read accesses) when the GOM bit of the CGST register is cleared (0) (means FCAN system inactive).
Whenever the FCAN system is in global operating mode (GOM = 1) the temporary buffer must not be written by the CPU. The global interrupt GINT2 signals accidental write accesses by CPU while the FCAN system is active.

(1) CAN message buffer section

The message buffer section consists of 32 message buffers. Each message buffer allocates 32 bytes.
The message buffers are not statically distributed and linked to the CAN modules, rather the user must determine the link of a message buffer to a CAN module by software. As a consequence the message buffers can be allocated to a CAN module according to the need of the particular CAN network.

Table 14-1: Configuration of the CAN Message Buffer Section

Address Offset[Note]	Name
800H to 81FH	Message buffer 0
820H to 83FH	Message buffer 1
840H to 85FH	Message buffer 2
. . .	
3C0H to 3DFH	Message buffer 30
3E0H to 3FFH	Message buffer 31

Note: The address of a message buffer entry is calculated according to the following formula:
effective address = PP_BASE + address offset

Each message buffer has the same register layout (refer to Table 14-2, “CAN Message Buffer Registers Layout,” on page 431).

Table 14-2: CAN Message Buffer Registers Layout

Address Offset Note 1, 2	Symbol Note 1	Name	Ref. Page	Access Type			
				R/W	1 bit	8 bit	16 bits
(m × 20H) + 800H	M_EVTm0	Message event register 0 Note 3	484	R/W		×	
(m × 20H) + 801H	M_EVTm1	Message event register 1 Note 3		R/W		×	
(m × 20H) + 802H	M_EVTm2	Message event register 2 Note 3	–	–			
(m × 20H) + 803H	M_EVTm3	Message event register 3 Note 3	484	R/W		×	
(m × 20H) + 804H	M_DLCm	Message data length code register	480	R/W		×	
(m × 20H) + 805H	M_CTRLm	Message control register	481	R/W		×	
(m × 20H) + 806H	M_TIMEm	Message time stamp register	483	R/W			×
(m × 20H) + 808H	M_DATAm0	Message data byte 0	478	R/W		×	
(m × 20H) + 809H	M_DATAm1	Message data byte 1		R/W		×	
(m × 20H) + 80AH	M_DATAm2	Message data byte 2		R/W		×	
(m × 20H) + 80BH	M_DATAm3	Message data byte 3		R/W		×	
(m × 20H) + 80CH	M_DATAm4	Message data byte 4		R/W		×	
(m × 20H) + 80DH	M_DATAm5	Message data byte 5		R/W		×	
(m × 20H) + 80EH	M_DATAm6	Message data byte 6		R/W		×	
(m × 20H) + 80FH	M_DATAm7	Message data byte 7		R/W		×	
(m × 20H) + 810H	M_IDLm	Message identifier register (lower half-word)	472	R/W			×
(m × 20H) + 812H	M_IDHm	Message identifier register (upper half-word)		R/W			×
(m × 20H) + 814H	M_CONFm	Message configuration register	473	R/W		×	
(m × 20H) + 815H	M_STATm	Message status register	475	R		×	
(m × 20H) + 816H	SC_STATm	Message set/clear status register	477	W			×
(m × 20H) + 818H to (m × 20H) + 81FH	–	Reserved	–	–			

Notes: 1. m = number of CAN message buffer (m = 00 to 31)

2. The address of a message buffer entry is calculated according to the following formula: effective address = PP_BASE + address offset

3. The V850E/CA2 Jupiter device does not contain an event processor. Therefore the message event bytes are reserved.
However, these registers can be used for storing user data in case that the event processing is disabled explicitly by clearing the bit EVM in the register "CAN global status register" and by clearing the bit ERQ in the "Message status register".

(2) CAN Interrupt Pending Registers Section

The layout of the interrupt pending register section is shown in Table 14-3.

Table 14-3: Relative Addresses of CAN Interrupt Pending Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
1004H	CCINTP	CAN interrupt pending register	467	R		×	×	
1020H	CGINTP	CAN global interrupt pending register	468	R		×	×	
				W			×	bit-set function only
1022H	C1INTP	CAN1 interrupt pending register	470	R		×	×	
				W		×	×	bit-clear function only
1024H	C2INTP	CAN2 interrupt pending register	470	R		×	×	
				W		×	×	bit-clear function only
1026H	C3INTP	CAN3 interrupt pending register	470	R		×	×	
				W		×	×	bit-clear function only
1028H	C4INTP	CAN4 interrupt pending register	470	R		×	×	
				W		×	×	bit-clear function only

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

(3) CAN Common Registers Section

The layout of the common register section is shown in Table 14-4.

Table 14-4: Relative Addresses of CAN Common Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
1000H	CSTOP	CAN stop register	454	R/W		×	×	
1010H	CGST	CAN global status register	457	R	×	×	×	
				W			×	bit set/clear function
1012H	CGIE	CAN global interrupt enable register	460	R	×	×	×	
				W			×	bit set/clear function
1014H	CGCS	CAN main clock select register	455	R	×	×	×	
				W	×	×	×	only if GOM bit = 0
1016H	CGTEN	CAN timer event enable register	462	R/W	×	×	×	
1018H	CGTSC	CAN global time system counter	462	R	×	×	×	
				W			×	complete clear only
101AH	CGMSS	CAN message search start register	464	W			×	write only
	CGMSR	CAN message search result register	465	R	×	×	×	read only
101CH	CTBR	CAN test bus register	466	R/W		×	×	

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = PP_BASE + address offset

(4) CAN Module Registers Section

The appropriate register section of each CAN module is shown in Table 14-5 for CAN module 1, in Table 14-6, “Relative Addresses of CAN Module 2 Registers,” on page 435 for CAN module 2, in Table 14-7, “Relative Addresses of CAN Module 3Note1 Registers,” on page 436 for CAN module 3 and in Table 14-8, “Relative Addresses of CAN Module 4Note1 Registers,” on page 437 for CAN module 4.

Table 14-5: Relative Addresses of CAN Module 1 Registers

Address Offset[Note2]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
1040H	C1MASKL0	CAN1 mask 0 register L	485	R/W		×	×	lower half-word
1042H	C1MASKH0	CAN1 mask 0 register H		R/W		×	×	upper half-word
1044H	C1MASKL1	CAN1 mask 1 register L		R/W		×	×	lower half-word
1046H	C1MASKH1	CAN1 mask 1 register H		R/W		×	×	upper half-word
1048H	C1MASKL2	CAN1 mask 2 register L		R/W		×	×	lower half-word
104AH	C1MASKH2	CAN1 mask 2 register H		R/W		×	×	upper half-word
104CH	C1MASKL3	CAN1 mask 3 register L		R/W		×	×	lower half-word
104EH	C1MASKH3	CAN1 mask 3 register H		R/W		×	×	upper half-word
1050H	C1CTRL	CAN1 control register	487	R		×	×	
				W			×	bit set/clear function
1052H	C1DEF	CAN1 definition register	492	R		×	×	
				W			×	bit set/clear function
1054H	C1LAST	CAN1 information register	496	R		×	×	read only
1056H	C1ERC	CAN1 error counter register	497	R		×	×	read only
1058H	C1IE	CAN1 interrupt enable register	498	R		×	×	
				W			×	bit set/clear function
105AH	C1BA	CAN1 bus activity register	501	R		×	×	
				W			×	bit set/clear function
105CH	C1BRP	CAN1 bit rate prescaler register	503	R		×	×	
				W		×	×	in initialisation state only (ISTAT = 1)
	C1DINF	CAN1 bus diagnostic information register	508	R		×	×	in diagnostic mode only
105EH	C1SYNC	CAN1 synchronization control register	506	R		×	×	
				W		×	×	

Notes: 1. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

2. The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset.

Table 14-6: Relative Addresses of CAN Module 2 Registers

Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1 bit	8 bits	16 bits	
1080H	C2MASKL0	CAN2 mask 0 register L	485	R/W		×	×	lower half-word
1082H	C2MASKH0	CAN2 mask 0 register H		R/W		×	×	upper half-word
1084H	C2MASKL1	CAN2 mask 1 register L		R/W		×	×	lower half-word
1086H	C2MASKH1	CAN2 mask 1 register H		R/W		×	×	upper half-word
1088H	C2MASKL2	CAN2 mask 2 register L		R/W		×	×	lower half-word
108AH	C2MASKH2	CAN2 mask 2 register H		R/W		×	×	upper half-word
108CH	C2MASKL3	CAN2 mask 3 register L		R/W		×	×	lower half-word
108EH	C2MASKH3	CAN2 mask 3 register H		R/W		×	×	upper half-word
1090H	C2CTRL	CAN2 control register	487	R		×	×	
				W			×	bit set/clear function
1092H	C2DEF	CAN2 definition register	492	R		×	×	
				W			×	bit set/clear function
1094H	C2LAST	CAN2 information register	496	R		×	×	read only
1096H	C2ERC	CAN2 error counter register	497	R		×	×	read only
1098H	C2IE	CAN2 interrupt enable register	498	R		×	×	
				W			×	bit set/clear function
109AH	C2BA	CAN2 bus activity register	501	R		×	×	
				W			×	bit-set/clear function
109CH	C2BRP	CAN2 bit rate prescaler register	503	R		×	×	
				W		×	×	in initialisation state only (ISTAT bit = 1)
	C2DINF	CAN2 bus diagnostic information register	508	R		×	×	in diagnostic mode only
109EH	C2SYNC	CAN2 synchronization control register	506	R		×	×	
				W		×	×	

Note: The address of a CAN module 2 register is calculated according to the following formula: effective address = PP_BASE + address offset

Table 14-7: Relative Addresses of CAN Module 3[Note1] Registers

Address Offset[Note2]	Symbol	Name	Ref. Page	Access Type R/W	1 bit	8 bits	16 bits	Comment
10C0H	C3MASKL0	CAN3 mask 0 register L	485	R/W		×	×	lower half-word
10C2H	C3MASKH0	CAN3 mask 0 register H		R/W		×	×	upper half-word
10C4H	C3MASKL1	CAN3 mask 1 register L		R/W		×	×	lower half-word
10C6H	C3MASKH1	CAN3 mask 1 register H		R/W		×	×	upper half-word
10C8H	C3MASKL2	CAN3 mask 2 register L		R/W		×	×	lower half-word
10CAH	C3MASKH2	CAN3 mask 2 register H		R/W		×	×	upper half-word
10CCH	C3MASKL3	CAN3 mask 3 register L		R/W		×	×	lower half-word
10CEH	C3MASKH3	CAN3 mask 3 register H		R/W		×	×	upper half-word
10D0H	C3CTRL	CAN3 control register	487	R		×	×	
				W			×	bit set/clear function
10D2H	C3DEF	CAN3 definition register	492	R		×	×	
				W			×	bit set/clear function
10D4H	C3LAST	CAN3 information register	496	R		×	×	read only
10D6H	C3ERC	CAN3 error counter register	497	R		×	×	read only
10D8H	C3IE	CAN3 interrupt enable register	498	R		×	×	
				W			×	bit set/clear function
10DAH	C3BA	CAN3 bus activity register	501	R		×	×	
				W			×	bit set/clear function
10DCH	C3BRP	CAN3 bit rate prescaler register	503	R		×	×	
				W		×	×	in initialisation state only (ISTAT bit = 1)
	C3DINF	CAN3 bus diagnostic information register	508	R		×	×	in diagnostic mode only
10DEH	C3SYNC	CAN3 synchronization control register	506	R		×	×	
				W		×	×	

Notes: 1. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

2. The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset.

Table 14-8: Relative Addresses of CAN Module 4[Note1] Registers

Address Offset[Note2]	Symbol	Name	Ref. Page	Access Type R/W	1 bit	8 bits	16 bits	Comment
1100H	C4MASKL0	CAN4 mask 0 register L	485	R/W		×	×	lower half-word
1102H	C4MASKH0	CAN4 mask 0 register H		R/W		×	×	upper half-word
1104H	C4MASKL1	CAN4 mask 1 register L		R/W		×	×	lower half-word
1106H	C4MASKH1	CAN4 mask 1 register H		R/W		×	×	upper half-word
1108H	C4MASKL2	CAN4 mask 2 register L		R/W		×	×	lower half-word
110AH	C4MASKH2	CAN4 mask 2 register H		R/W		×	×	upper half-word
110CH	C4MASKL3	CAN4 mask 3 register L		R/W		×	×	lower half-word
110EH	C4MASKH3	CAN4 mask 3 register H		R/W		×	×	upper half-word
1110H	C4CTRL	CAN4 control register	487	R		×	×	
				W			×	bit set/clear function
1112H	C4DEF	CAN4 definition register	492	R		×	×	
				W			×	bit set/clear function
1114H	C4LAST	CAN4 information register	496	R		×	×	read only
1116H	C4ERC	CAN4 error counter register	497	R		×	×	read only
1118H	C4IE	CAN4 interrupt enable register	498	R		×	×	
				W			×	bit set/clear function
111AH	C4BA	CAN4 bus activity register	501	R		×	×	
				W			×	bit set/clear function
111CH	C4BRP	CAN4 bit rate prescaler register	503	R		×	×	
				W		×	×	in initialisation state only (ISTAT bit = 1)
	C4DINF	CAN4 bus diagnostic information register	508	R		×	×	in diagnostic mode only
111EH	C4SYNC	CAN4 synchronization control register	506	R		×	×	
				W		×	×	

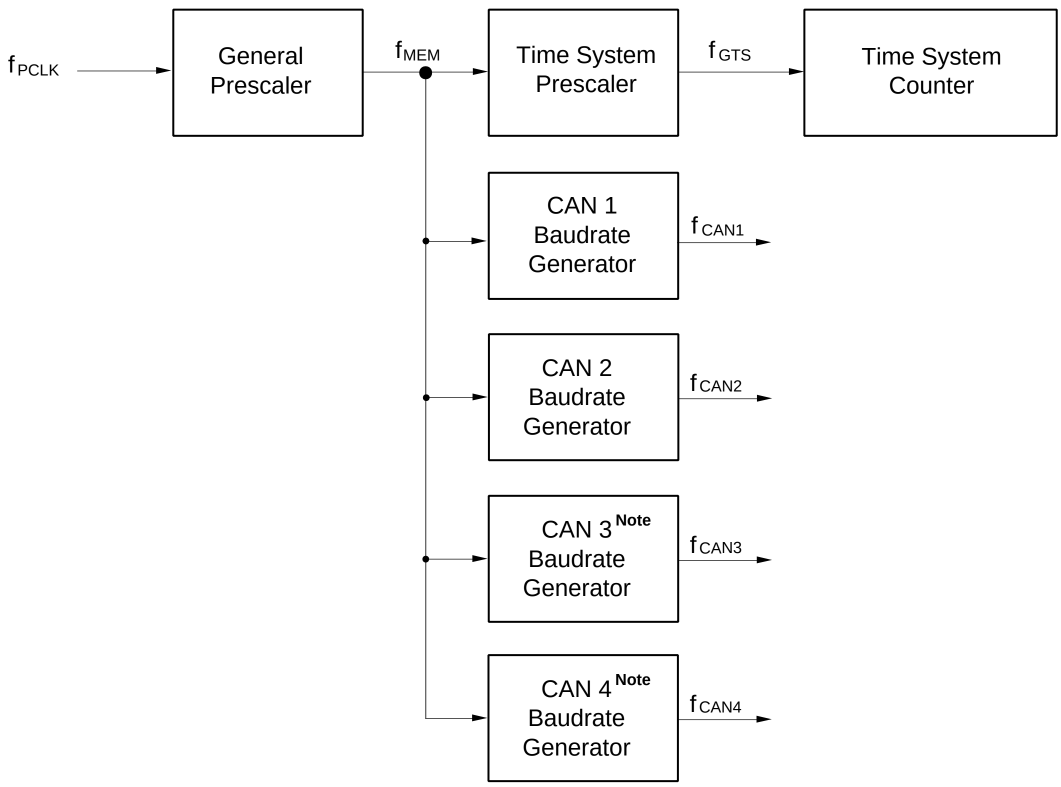
Notes: 1. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

2. The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset.

14.2.3 Clock structure

All functional blocks within the FCAN system are supplied by a unique clock (f_{MEM}) derived from the internal system clock (f_{PLCK}).

Figure 14-3: Clock Structure of the FCAN System



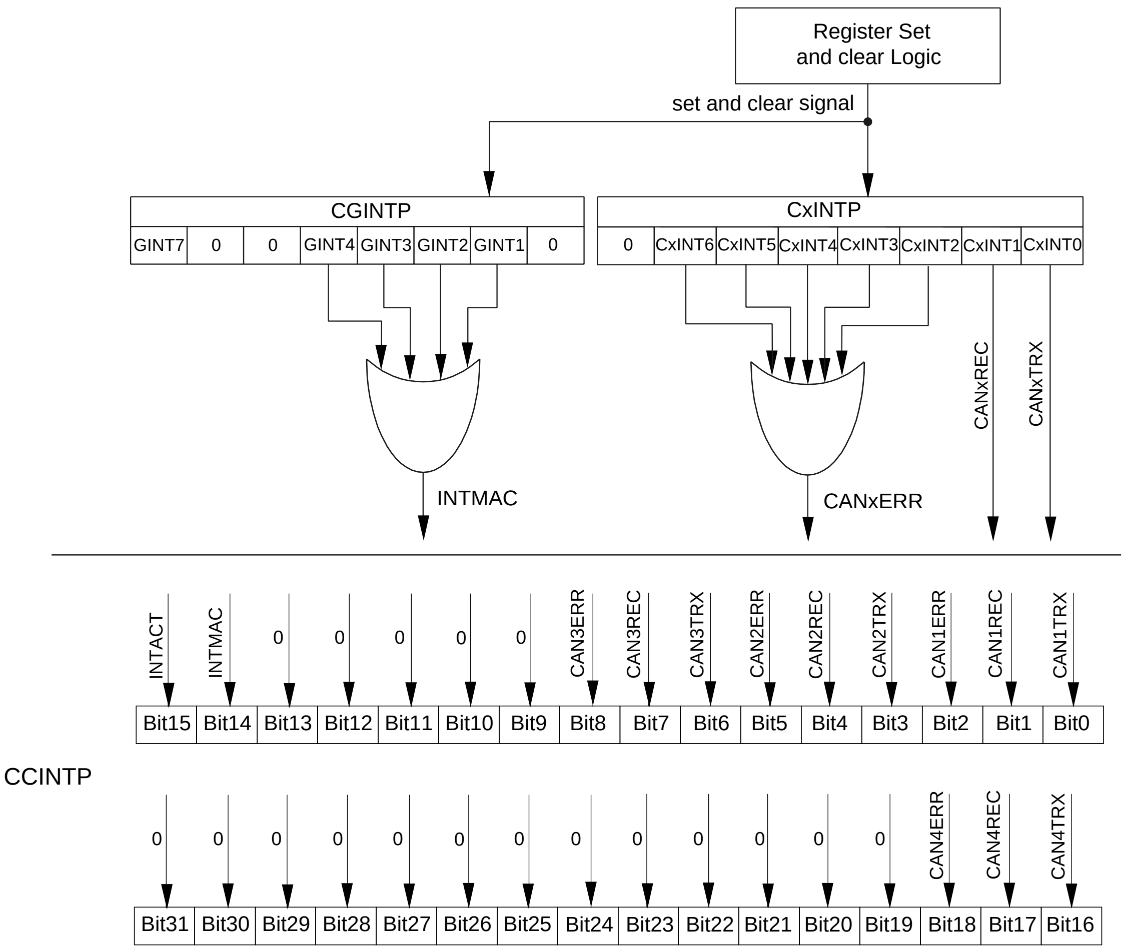
Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

A functional block for a global time system is integrated in the FCAN system. That functional block is supplied by the global time system clock (f_{GTS}), which is derived from f_{MEM}. The time system prescaler scales f_{GTS} and is controlled by the CGCS register.
The time base of the global time system is realised by the 16-bit free-running counter, the CAN global time system counter (CGTSC). Time stamp information is captured from the CGTSC counter.

14.2.4 Interrupt handling

The very high number of interrupt events generated by the FCAN system does not allow to assign an independent interrupt vector of the V850E/CA2 to each event. Therefore, the interrupt request signals are bundled into groups and the grouped interrupt request signal is then assigned to an independent interrupt vector.
The concept of interrupt request signal bundling leads to the fact that all interrupt request signals of the FCAN system are designed as interrupt pending signals. Interrupt pending signals are not automatically treated by an interrupt service routine like interrupt request signals with an unambiguous interrupt vector. Rather, on occurrence of the interrupt event the interrupt signal is generated and latched.
In the interrupt service routine the software must analyse, which particular interrupt event caused the interrupt request by scanning the interrupt pending flags of a bundled interrupt signal group. After the particular interrupt has been identified, the corresponding interrupt pending flag must be reset by software at least before leaving the interrupt service routine.

Figure 14-4: FCAN Interrupt Bundling of V850E/CA2



Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Remark: x = 1 to 2 for the derivative µPD703128 (A), x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1)

The interrupt pending registers of the FCAN system are:

- CGINTP: Global interrupt pending register
- C1INTP: CAN module 1 interrupt pending register
- C2INTP: CAN module 2 interrupt pending register
- C3INTP: CAN module 3 interrupt pending register
- C4INTP: CAN module 4 interrupt pending register

Additionally the entire interrupt pending flags are summarized in one register, the CAN interrupt pending register (CCINTP). However, the CCINTP register is a read-only register, and cannot be used for clearing the interrupt pending flags.
For details on the interrupt pending registers refer to the chapter 14.3.3 “CAN interrupt pending registers” on page 467.

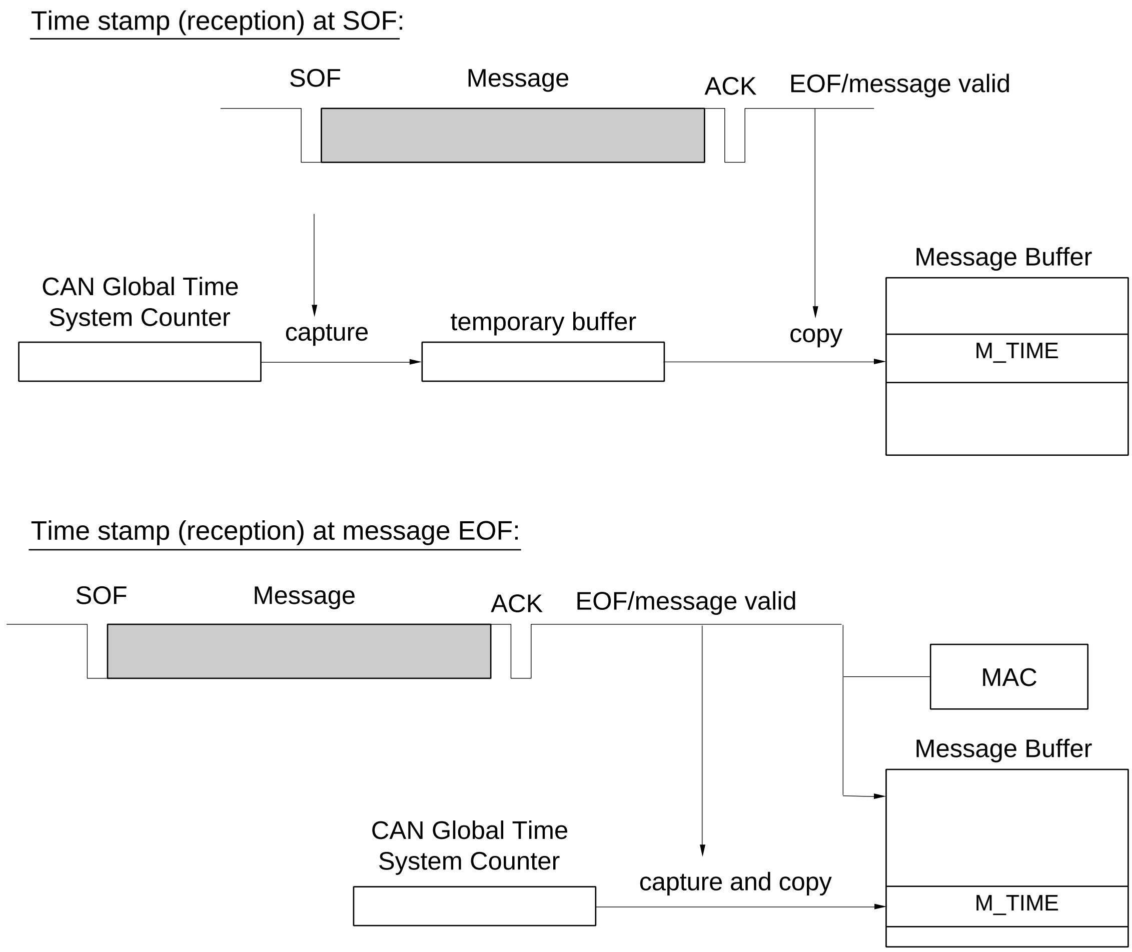
14.2.5 Time stamp

The FCAN system offers a time stamp capture capability at message reception and transmission. The time stamp capture function is used to realize a synchronized, global clock in a CAN network, also called global time system. However, the development and functionality of such a global clock system has to be implemented by the user.

For time stamp capturing at message reception two trigger events are selectable (see Figure 9-5). The counter value of the CAN global time system counter (CGTSC) is either captured upon the start-of-frame signal (SOF) of the receive message or it is captured at the time the message is detected as valid, i.e. if no error was detected until the last but one bit of the end-of-frame (EOF) was received. The selection of the two trigger options is controlled by the TMR bit in the CxCTRL register (x = 1 to 2 for the derivative µPD703128 (A), x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1)). The capture value itself is stored in the M_TIMEm register (m = 00 to 31) of the message buffer, for which the received message has been accepted.

Remark: The value of M_TIMEm register is undefined when an error occurs while receiving the message.

Figure 14-5: Time Stamp Capturing at Message Reception



For the time stamp capturing at message transmission the SOF signal of the transmit message is used as the event trigger (see Figure 14-6).
The captured value from the CGTSC counter is written into particular data bytes of the transmit message's data field. Table 14-9 shows the scheme about which data bytes of the data field are replaced with the time stamp capture value according to the setting of the M_DLCm register (m = 00 to 31).

Figure 14-6: Time Stamp Capturing at Message Transmission

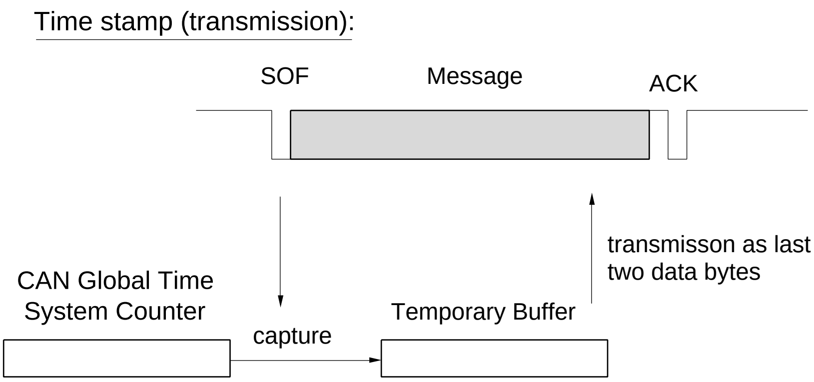


Table 14-9: Transmitted Data On the CAN Bus (ATS = 1)

M_DLCm	Bus Data 1	Bus Data 2	Bus Data 3	Bus Data 4	Bus Data 5	Bus Data 6	Bus Data 7	Bus Data 8
1	M_DATAm0	–	–	–	–	–	–	–
2	lower 8-bit of CGTSC-Note	upper 8-bit of CGTSC-Note	–	–	–	–	–	–
3	M_DATAm0	lower 8-bit of CGTSC-Note	upper 8-bit of CGTSC-Note	–	–	–	–	–
4	M_DATAm0	M_DATAm1	lower 8-bit of CGTSC-Note	upper 8-bit of CGTSC-Note	–	–	–	–
5	M_DATAm0	M_DATAm1	M_DATAm2	lower 8-bit of CGTSC-Note	upper 8-bit of CGTSC-Note	–	–	–
6	M_DATAm0	M_DATAm1	M_DATAm2	M_DATAm3	lower 8-bit of CGTSC-Note	upper 8-bit of CGTSC-Note	–	–
7	M_DATAm0	M_DATAm1	M_DATAm2	M_DATAm3	M_DATAm4	lower 8-bit of CGTSC-Note	upper 8-bit of CGTSC-Note	–
8	M_DATAm0	M_DATAm1	M_DATAm2	M_DATAm3	M_DATAm4	M_DATAm5	lower 8-bit of CGTSC-Note	upper 8-bit of CGTSC-Note

Note: CGTSC value captured at SOF.

Remark: m = 00 to 31

14.2.6 Message handling

In the FCAN system the assignment of message buffers to the CAN modules is not defined by hardware. Each message buffer in the message buffer section can be assigned to any CAN module by software. The message buffers have individual configuration registers to assign the CAN module and to specify the message buffer type.
Basically, a message buffer can be selected as a transmit message buffer or as a receive message buffer. For receive message buffers there are further differentiations according to the mask links.

(1) Message transmission

According to the CAN protocol the highest prior message must always gain the CAN bus access against lower prior messages sent by other nodes at the same time (due to arbitration mechanism of CAN protocol) and against messages waiting to be transmitted in the same node (i.e. inner priority inversion).
The FCAN system scans the message buffer section at the beginning of each message transmit to analyse that no other message with a higher priority is waiting to be transmitted on the same CAN bus. The FCAN system avoids inner priority inversion automatically.

Example:

5 transmit messages are waiting to be sent at the same time in the example shown in Table 14-10, “Example for Automatic Transmission Priority Detection,” on page 444. Although the priority of the transmit messages are not sorted according any scheme, the sequence of transmits on the CAN bus is:

<1> message buffer number 15 (ID = 023H)
<2> message buffer number 1 (ID = 120H)
<3> message buffer number 22 (ID = 123H)
<4> message buffer number 14 (ID = 223H)
<5> message buffer number 2 (ID = 229H)

Table 14-10: Example for Automatic Transmission Priority Detection

Message Buffer Address Offset[Note1]	Message Buffer Number	Message Buffer Link	Message Buffer Type[Note2]	Waiting for Transmission	Identifier
400H	31				
⋮	⋮				
300H	24				
2E0H	23				
2C0H	22	CAN 1	TRX	3	123H
2A0H	21				
280H	20				
260H	19				
240H	18				
220H	17				
200H	16				
1E0H	15	CAN 1	TRX	3	023H
1C0H	14	CAN 1	TRX	3	223H
1A0H	13				
180H	12				
160H	11				
140H	10				
120H	9				
100H	8				
8E0H	7				
8C0H	6				
8A0H	5				
880H	4				
860H	3				
840H	2	CAN 1	TRX	3	229H
820H	1	CAN 1	TRX	3	120H
800H	0				

Notes: 1. The address of a message buffer entry is calculated according to the following formula: effective address = PP_BASE + address offset

2. TRX = transmit message

Caution: In case more than 5 transmit messages are linked to a CAN module, the user must allocate the 5 higher prior transmit messages to message buffers with a lower address. There is no sorting needed among the 5 higher prior message buffer.

Table 14-11: Example for Transmit Buffer Allocation When More Than 5 Buffers Linked to a CAN Module

Message Buffer Address Offset Note 1	Message Buffer Number	Message Buffer Link	Message Buffer Type Note 2	Identifier
400H	31			
. . .	. . .			
300H	24			
2E0H	23			
2C0H	22	CAN1	TRX	005H
2A0H	21			
280H	20			
260H	19	CAN1	TRX	006H
240H	18			
220H	17			
200H	16			
1E0H	15	CAN1	TRX	007H
1C0H	14	CAN1	TRX	001H Note 3
1A0H	13			
180H	12			
160H	11			
140H	10	CAN1	TRX	003H Note 3
120H	9			
100H	8			
8E0H	7			
8C0H	6	CAN1	TRX	000H Note 3
8A0H	5			
880H	4			
860H	3			
840H	2	CAN1	TRX	004H Note 3
820H	1	CAN1	TRX	002H Note 3
800H	0			

Notes: 1. The address of a message buffer entry is calculated according to the following formula: effective address = PP_BASE + address offset

2. TRX = transmit message

3. 5 higher prior transmit messages assigned to message buffers with lower address values.

(2) Message reception

Due to the vast initialisation possibilities for each message buffer in the FCAN system, it is possible that a received message fits in several message buffers assigned to a CAN module.
A fixed rule according to the priority classes has been implemented to avoid arbitrary message storage and uncontrolled behaviour.
The storage priority for data frames and for remote frames is different (refer to Table 14-12 and Table 14-13).

Table 14-12: Storage Priority for Reception of Data Frames

Priority Class	Condition
1 (high)	received data frame fits in non-masked receive buffer
2	received data frame fits in receive buffer linked to mask 0
3	received data frame fits in receive buffer linked to mask 1
4	received data frame fits in receive buffer linked to mask 2
5 (low)	received data frame fits in receive buffer linked to mask 3

Table 14-13: Storage priority for Reception of Remote Frames

Priority Class	Condition
1 (high)	received remote frame fits in transmit buffer
2	received remote frame fits in non-masked receive buffer
3	received remote frame fits in receive buffer linked to mask 0
4	received remote frame fits in receive buffer linked to mask 1
5	received remote frame fits in receive buffer linked to mask 2
6 (low)	received remote frame fits in receive buffer linked to mask 3

Caution: A priority class with lower priority don’t provide a backup for classes with higher priority. That means that a message (i.e. data frame / remote frame) is explicitly stored in the priority class with higher priority and never stored in the lower prior class.

Example:

Two receive message buffers are linked to CAN module 1:

- Buffer 1: non-masked receive buffer with identifier ID_K
- Buffer 2: receive buffer with ID_K linked to mask 2.

Under that configuration a message with ID_K is never stored in the receive buffer linked to mask 2, but always into the non-masked receive buffer.

Furthermore, there is a fixed inner storage rule in case several buffers of the same priority class are linked to a CAN module. For the inner priority class storage rule the data new flag (DN) in the M_STATm register is the first storage criteria (m = 00 to 31).
Whenever the DN flag cannot provide an unambiguous criteria for storing the message (i.e. there are several message buffers of the same priority class with DN flag set or not set) the physical message buffer number is chosen as the second criteria.

Table 14-14: Inner Storage Priority Within a Priority Class

Priority	First Criteria	Priority	Second Criteria
1 (high)	DN flag not set	1 (high)	lowest physical message buffer number
		2 (low)	next physical message buffer number
2 (low)	DN flag set	1 (high)	lowest physical message buffer number
		2 (low)	next physical message buffer number

Example:

When the very first message is received, which fits into several message buffer of the same priority class, the DN flag in all buffers is not set, hence that message is stored in the buffer with the lowest physical buffer number. Subsequent messages are stored to the message buffers in ascending message buffer number order as long the DN flags remains as set into the buffer of the previous message storage.
As soon the CPU reads one of the message buffer with DN flag set and then clears the DN flag, the storing in ascending message buffer number order is interrupted.

Due to the storage priority for receive messages it is possible to design multiple buffer arrays for a CAN message – while not all message buffers assigned to the same identifier contain new data (DN flag set) the FCAN system will store the data in the next free message buffer (DN flag cleared).

14.2.7 Mask handling

The FCAN system supports two concepts of message reception, the BasicCAN concept and the Full-CAN concept.
In the Full-CAN concept a particular message buffer accepts only one single message, hence there is no further sorting and filtering required by software. As a consequence only one unambiguous identifier is assigned to a message buffer.

In the BasicCAN concept a receive message buffer operates as a channel, which can accept several messages. After reception software must sort, respectively filter, which particular message has been received.
By the usage of hardware masks the range of receive messages can be limited to reduce the CPU load caused by message sorting.

In the FCAN system each CAN module provides 4 different masks.
For a receive message buffer assigned to a CAN module one of the 4 masks can be selected when the BasicCAN concept is used.

When using a mask, a certain identifier value must be written into the identifier register M_IDm (equals 32 bit value build by M_IDHm and M_IDLm) of the receive message buffer at initialisation.
Then the linked mask CxMASKn composed from CxMASKHn and CxMASKLn determines which identifier bits of a received message must match exactly to accept the received message for the message buffer.
The mask facilitates that certain identifier bits of the received message will not be compared with the corresponding identifier bits of the message buffer, thus several messages might be accepted for the receive message buffer.

Remarks:
1. n = 0 to 3
2. m = 00 to 31
3. x = 1 to 2 for the derivative µPD703128 (A), x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1)

14.2.8 Remote frame handling

The FCAN macro offers enhanced features for generating remote frames and for the reaction of a CAN module upon remote frames.

(1) Generation of a remote frame

According to the CAN specification a remote frame has the same format as a data frame except the RTR bit of the control field, which has recessive level, and the data field, which is omitted completely.
By means of a remote frame, receiving nodes can request the transmitting node of a particular message for sending an update of that message to the CAN bus. Usually remote frames are generated from CAN nodes which do not provide the requested message by themselves.

In the FCAN system a remote frame is automatically sent, when setting the transmit request bit (TRQ) of the M_STATm register for a message buffer defined as receive message buffer (m = 00 to 31). Same as for generating a data frame from a transmit message buffer, the ready bit (RDY) of M_STATm register must be set (1).

Remote frames can also be generated by means of a transmit message buffer by setting the RTR bit of the M_CTRLm register, and using the same transmission procedure as for data frames. However, from application point of view that method is not recommended, because it consumes message buffer resources unnecessarily. A data frame in a CAN network can be provided, i.e. transmitted, by only one node. All other nodes in the network may receive that data frame. Using a transmit message buffer for a remote frame generation means that two message buffers for handling of one message within one node are required - one receive message buffer for the reception of a data frame, and the transmit message buffer explicitly for the remote frame generation.

(2) Reception of a remote frame

The FCAN allows the reception of remote frames in message buffers defined for reception or for transmission.

(a) Reception in a receive message buffer

If a remote frame is received in a message buffer m (m = 00 to 31) configured for reception, the following message buffer information will be updated:

M_DLCm	message data length code register
M_CTRLm	message control register
M_TIMEm	message time stamp register (16-bit)
M_DATAm0	message data byte 0
M_DATAm1	message data byte 1
M_DATAm2	message data byte 2
M_DATAm3	message data byte 3
M_DATAm4	message data byte 4
M_DATAm5	message data byte 5
M_DATAm6	message data byte 6
M_DATAm7	message data byte 7
M_IDLm	message identifier register (lower half word)
M_IDHm	message identifier register (upper half word)
M_STATm	message status register

Remarks: **1.** Receiving a remote frame in a receive message buffer does not activate any automatic remote frame handling activities from the FCAN system. The application software must handle the remote frame in the expected way.

2. RMDE0, RMDE1 bits as well as ATS bit of M_CTRLm register are set to 0.

(b) Reception in a transmit message buffer

When the FCAN system searches for the corresponding message buffer after reception of a remote frame and finds a message buffer with a matching identifier, which is defined for transmission, the content of the remote frame is not stored but programmable reactions are launched.
Accepting a remote frame for a transmit message buffer does not change the content of the transmit message buffer except the DN flag of the M_STATm register depending on the setting of the RMDE0, RMDR1 and RTR bits of the M_CTRLm register (refer to Table 14-15).
The remote frame reception in a transmit message buffer causes a reaction according to the setting of the RMDE0, RMDR1 bits and the RTR bit of the M_CTRLm register. The following reactions are programmable:

- Generation of an auto-answer (i.e. TRQ bit of the transmit message buffer is automatically set without any CPU interaction).
- Signalling the remote frame reception by updating the DN flag in the transmit message buffer.
- No reaction at all.

Table 14-15 shows the detailed handling (reaction) upon the reception of a remote frame for a transmit message buffer depending on the settings of RMDE0, RMDE1 and RTR flags.

Table 14-15: Remote Frame Handling upon Reception into a Transmit Message Buffer

M_CTRLm setting			Resulting Automatic Remote Frame Handling	
RMDE0	RMDE1	RTR	DN flag	other actions
0	0	x	no change	– („ignore remote frame“)
1	0	0	Clear when transmit message buffer sent successfully	send transmit message buffer (data frame) as an automatic answer.
		1	no change	_ Note
0	1	x	DN is set upon reception	–
1	1	0	Clear when transmit message buffer sent successfully	send transmit message buffer (data frame) as an automatic answer.
		1	DN is set upon reception	_ Note

Note: Auto-answer upon remote frame is suppressed, because the transmit message buffer is configured to send a remote frame (RTR = 1).

Remarks: **1.** In case a remote frame is automatically answered upon receiving a remote frame for a transmit message buffer, the reception of the remote frame is not notified by a receive interrupt. However, the successful transmission of the data frame (i.e. the automatic answer) is notified by the corresponding transmit interrupt.

2. m = 00 to 31

14.3 Control and Data Registers

14.3.1 Bit set/clear function

Direct writing of data (bit operations, read-modify write, direct writing of a target value) is not allowed to few specific registers, where bit setting and bit clearing might be performed by CPU and by the FCAN system. The following registers of the FCAN system are concerned.

- CAN global status register (CGST)
- CAN global interrupt enable register (CGIE)
- CAN global interrupt pending register (CGINTP)
- CAN x interrupt pending registers (CxINTP)
- CAN x control registers (CxCTRL)
- CAN x definition registers (CxDEF)
- CAN x interrupt enable registers (CxIE)
- CAN x bus activity registers (CxBA)

Remark: x = 1 to 2 for the derivative µPD703128 (A), x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1)

Registers like above, where bit access and direct write operations are prohibited, are organized in such a way that all bits allowed for manipulation are located in the lower byte (bits 7 to 0), while in the upper byte (bits 15 to 8) either no or read-only information is located.

The registers can be read in the usual way to get all 16 data bits in their actual setting (ref. to appropriated register description).

For setting or clearing any of the lower 8 bits the following mechanism is implemented:
When writing 16-bit data to the register address, each of the lower 8 data bits indicates whether the corresponding register bit should be cleared (data bit set) or remain unchanged (data bit not set). Each of the upper 8 data bits indicates whether the corresponding register bit should be set (data bit set) or remain unchanged (data bit cleared).
The organization of 16-bit data write for such registers is shown in Figure 14-7, “16-Bit Data Write Operation for Specific Registers,” on page 453.

Figure 14-7: 16-Bit Data Write Operation for Specific Registers

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SE_7	SE_6	SE_5	SE_4	SE_3	SE_2	SE_1	SE_0	CL_7	CL_6	CL_5	CL_4	CL_3	CL_2	CL_1	CL_0

<table>
<tr><th>Bit Name</th><th>Function</th></tr>
<tr><td>SE_n</td><td>Sets the register bit n.
0: No change of register bit n
1: Register bit n is set (1)</td></tr>
<tr><td>CL_n</td><td>Clears the register bit n.
0: No change of register bit n
1: Register bit n is cleared (0)</td></tr>
<tr><td>SE_n,
CL_n</td><td>Sets/clears the Register bit n.
<table>
<tr><th>SE_n</th><th>CL_n</th><th>Status of Register Bit n</th></tr>
<tr><td>0</td><td>1</td><td>Register bit n is cleared (0)</td></tr>
<tr><td>1</td><td>0</td><td>Register bit n is set (1)</td></tr>
<tr><td colspan="2">Others</td><td>No change in register bit n value.</td></tr>
</table></td></tr>
</table>

Remarks: 1. If only bits are to be cleared, the 16-bit write access can be replaced by an 8-bit write access to the register address. If only bits are to be set, the 16-bit write access can be replaced by an 8-bit write access to the register address+1. Nevertheless, for better visibility of the program code it is recommended to perform only 16-bit write accesses.

2. n = 0 to 7

14.3.2 Common registers

(1) CAN stop register (CSTOP)

The CSTOP register controls the clock supply of the FCAN system.
This register can be read/written in 8-bit and16-bit units.

Figure 14-8: CAN Stop Register (CSTOP)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CSTOP	CSTP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1000H	0000H

Bit Position	Bit Name	Function
15	CSTP	Controls the clock supply for the complete FCAN system. The CSTP flag can be used to reduce the power consumption when the FCAN system is set to SLEEP mode and STOP mode to a minimum. 0: FCAN system is supplied with clock f_{MEM}. 1: Clock supply of the FCAN system is stopped. **Remark:** When switching off the clock supply of the FCAN system during SLEEP mode, wake-up by CAN bus activity is possible. But, instead of CxINT4 interrupt (i.e. wake-up from SLEEP mode interrupt), the GINT3 interrupt must be used. **Cautions: 1. In case CSTP is set (1), access to the register and buffer of the FCAN system is impossible, except access to the CSTOP register.** **2. Do not set CSTP = 1 while the FCAN system is under normal operation, especially while a CAN module handles messages on the CAN bus. A sudden stop of the FCAN system might cause malfunctions of the entire CAN network.**

Note: The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset

(2) CAN main clock select register (CGCS)

The CGCS register controls the internal memory access clock (f_{MEM}), which is used as main clock for each CAN module, as well as the global time system clock (f_{GTS}), used for the time stamp function and event generation. (For details refer to chapter 14.2.3 "Clock structure" on page 438.
This register can be read/written in 1-bit, 8-bit and16-bit units.

Figure 14-9: CAN Main Clock Select Register (CGSC) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGCS	CGTS7	CGTS6	CGTS5	CGTS4	CGTS3	CGTS2	CGTS1	CGTS0	GTSC0	GTSC0	0	MCS	MCP3	MCP2	MCP1	MCP0	1014H	7F05H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15 to 8</td><td>CGTS7 to CGTS0</td><td>Specifies the 8-bit prescaler compare value for the global time system clock (f_{GTS}) (ref. to Fig. 9-11).
<table>
<tr><th>CGTS7 to CGTS0 (k)</th><th>Prescaler (k + 1)</th><th>Global Time System Clock $f_{GTS} = f_{GTS1} / (k + 1)$</th></tr>
<tr><td>0</td><td>1</td><td>$f_{GTS} = f_{GTS1}$</td></tr>
<tr><td>1</td><td>2</td><td>$f_{GTS} = f_{GTS1} / 2$</td></tr>
<tr><td>2</td><td>3</td><td>$f_{GTS} = f_{GTS1} / 3$</td></tr>
<tr><td>.
.
.</td><td>.
.
.</td><td>.
.
.</td></tr>
<tr><td>255</td><td>256</td><td>$f_{GTS} = f_{GTS1} / 256$</td></tr>
</table>
Remark: The global time system clock is the source clock for the 16-bit timer used for the time stamp functionality. This clock is common for all CAN modules.</td></tr>
<tr><td>7, 6</td><td>GTCS1, GTCS0</td><td>Selects the global time system basic clock (f_{GTS1}) from the memory clock (f_{MEM}) (ref. to Fig. 9-11).
<table>
<tr><th>GTCS1</th><th>GTCS0</th><th>Global Time System Basic Clock (f_{GTS1})</th></tr>
<tr><td>0</td><td>0</td><td>$f_{GTS1} = f_{MEM} / 2$</td></tr>
<tr><td>0</td><td>1</td><td>$f_{GTS1} = f_{MEM} / 4$</td></tr>
<tr><td>1</td><td>0</td><td>$f_{GTS1} = f_{MEM} / 8$</td></tr>
<tr><td>1</td><td>1</td><td>$f_{GTS1} = f_{MEM} / 16$</td></tr>
</table></td></tr>
<tr><td>4</td><td>MCS</td><td>Selects input clock for the memory access clock prescaler (f_{MEM1}) (ref. to Fig. 9-10).
0: f_{MEM1} = internal system clock (f_{CPU})
1: f_{MEM1} = external clock input (f_{EXT})[Note]
Note: V850E/CA2 doesn't offer an external clock f_{MEM} supply pin. Therefore, the MCS must not be set at any time.</td></tr>
</table>

Note: The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-9: CAN Main Clock Select Register (CGSC) (2/2)

Bit Position	Bit Name	Function
3 to 0	MCP3 to MCP0	Specifies the prescaler for the memory access clock (f_{MEM}) (ref. to Fig. 9-10).

MCP3	MCP2	MCP1	MCP0	Prescaler (m+1)	Memory Clock $f_{MEM} = f_{MEM1} / (m+1)$
0	0	0	0	1	$f_{MEM} = f_{MEM1}$
0	0	0	1	2	$f_{MEM} = f_{MEM1} / 2$
0	0	1	0	3	$f_{MEM} = f_{MEM1} / 3$
. . .				. . .	
1	1	1	1	16	$f_{MEM} = f_{MEM1} / 16$

Figure 14-10: Configuration of FCAN System Main Clock

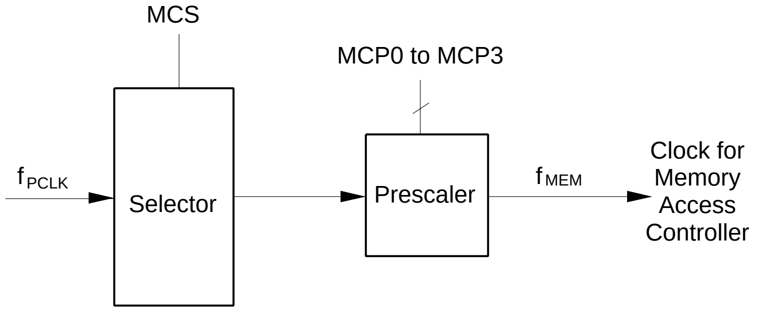
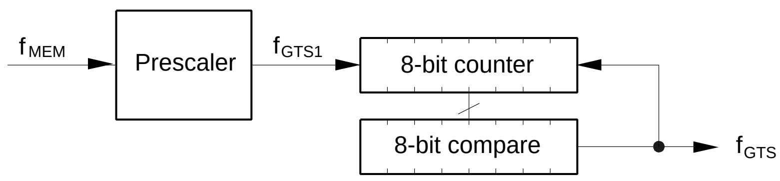


Figure 14-11: Configuration of FCAN Global Time System Clock



(3) CAN global status register (CGST)

The CGST register indicates and controls the operation modes of the FCAN system. This register can be read in 1-bit, 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies. (Refer to chapter 14.3.1 "Bit set/clear function" on page 452)

Figure 14-12: CAN Global Status Register (CGST) (1/3)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGST	0	0	0	0	0	0	0	1	MERR	0	0	0	EFSD	TSM	EVM	GOM	1010H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CGST	0	0	0	0	ST_EFSD	ST_TSM	ST_EVM	ST_GOM	CL_MERR	0	0	0	CL_EFSD	CL_TSM	CL_EVM	CL_GOM	1010H

Read (1/2)

Bit Position	Bit Name	Function
7	MERR	Indicates the error status of the memory access controller (MAC). 0: No error occurrence 1: At least one error occurred since the flag was cleared last A MAC error occurs under the following conditions: - An attempt to clear the GOM flag was performed although not all CAN modules are set to initialization state. - Access to an illegal address, or access is prohibited by MAC (see GOM flag description below)
3	EFSD	Enable forced shut down. 0: Forced shut down is disabled. 1: Forced shut down is enabled. **Remark:** In case of an emergency it might be necessary to reset all CAN modules immediately. In this case the EFSD flag has to be set before clearing the GOM flag.
2	TSM	Indicates the operating mode of the CAN global time system counter (CGTSC). 0: CAN global time system counter is stopped. 1: CAN global time system counter is operating.
1	EVM	Indicates the event operating mode. 0: CAN bridge is disabled. 1: CAN bridge is enabled. **Remark:** Due to the reason that no CAN bridge is implemented in the V850E/CA2 device, this bit must not be set at any time.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-12: CAN Global Status Register (CGST) (2/3)

Read (2/2)

Bit Position	Bit Name	Function
0	GOM	Indicates the global operating mode. 0: Access to CAN module registers is prohibited, except mask registers and temporary buffers.[Note 1] 1: Operation of all CAN modules are enabled. Temporary buffers can be read only.[Note 1] **Caution: To ensure that resetting the CAN modules do not cause any unexpected behaviour on the CAN bus, the GOM flag can only be cleared, if all CAN modules are set into initialisation state (exception: forced-shut-down, see EFSD flag). If the software clears the flag while at least one CAN module is still not in initialisation state (ISTAT flag of CxCTRL register (x = 1 to 4) is set (1)), the GOM flag remains set.**

Figure 14-12: CAN Global Status Register (CGST) (3/3)

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 3</td><td>ST_EFSD,
CL_EFSD</td><td>Sets/clears the EFSD bit.
<table>
<tr><th>ST_EFSD</th><th>CL_EFSD</th><th>Status of EFSD Bit</th></tr>
<tr><td>0</td><td>1</td><td>EFSD bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>EFSD bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in EFSD bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_TSM,
CL_TSM</td><td>Sets/clears the TSM bit.
<table>
<tr><th>ST_TSM</th><th>CL_TSM</th><th>Status of TSM Bit</th></tr>
<tr><td>0</td><td>1</td><td>TSM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TSM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TSM bit value.</td></tr>
</table></td></tr>
<tr><td>9, 1</td><td>ST_EVM,
CL_EVM</td><td>Sets/clears the EVM bit.
<table>
<tr><th>ST_EVM</th><th>CL_EVM</th><th>Status of EVM Bit</th></tr>
<tr><td>0</td><td>1</td><td>EVM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>EVM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in EVM bit value.</td></tr>
</table></td></tr>
<tr><td>8, 0</td><td>ST_GOM,
CL_GOM</td><td>Sets/clears the GOM bit.
<table>
<tr><th>ST_GOM</th><th>CL_GOM</th><th>Status of GOM Bit</th></tr>
<tr><td>0</td><td>1</td><td>GOM bit is cleared (0).Note 2</td></tr>
<tr><td>1</td><td>0</td><td>GOM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in GOM bit value.</td></tr>
</table></td></tr>
<tr><td>7</td><td>CL_MERR</td><td>Clears the MERR bit.
0: No change of MERR bit.
1: MERR bit is cleared (0).</td></tr>
</table>

Notes: 1. Access to the message buffer area is not affected.

2. Refer to description of GOM flag above.

(4) CAN global interrupt enable register (CGIE)

The CGIE register enables the global interrupts of the FCAN system.
This register can be read in 1-bit, 8-bit and16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies. (Refer to 14.3.1 "Bit set/clear function" on page 452)

Figure 14-13: CAN Global Interrupt Enable Register (CGIE) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGIE	0	0	0	0	1	0	0	01	G_IE7	0	0	0	0	G_IE2	G_IE1	0	1012H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CGIE	ST_ G_IE7	0	0	0	0	ST_ G_IE2	ST_ G_IE1	0	CL_ G_IE7	0	0	0	0	CL_ G_IE2	CL_ G_IE1	0	1012H

Read

Bit Position	Bit Name	Function
7	G_IE7	Enables interrupt by CAN bridge. 0: Interrupt disabled 1: Interrupt enabled **Remark:** Due to the reason that no CAN bridge is implemented in the V850E/CA2 device, this bit must not be set at any time.
2	G_IE2	Enables illegal address interrupt. 0: Interrupt disabled 1: Interrupt enabled **Remarks: 1.** Interrupt signals any access to CAN module register while GOM bit of the CGST register is reset (0). **2.** Interrupt signals a write access to the temporary buffer while GOM bit of the CGST register is set (1).
1	G_IE1	Enables "access to unavailable memory addresses" interrupt. 0: Interrupt disabled 1: Interrupt enabled **Remarks: 1.** Interrupt signals an access to any CAN memory area not explicitly specified. **2.** Interrupt signals an illegal FCAN system shut down, i.e. GOM bit is going to be cleared while at least one of the CAN modules is not in initialisation state or "forced shut down" is not selected.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-13: CAN Global Interrupt Enable Register (CGIE) (2/2)

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 7</td><td>ST_G_IE7,
CL_G_IE7</td><td>Sets/clears the G_IE7 bit.
<table>
<tr><th>ST_G_IE7</th><th>CL_G_IE7</th><th>Status of G_IE7 Bit</th></tr>
<tr><td>0</td><td>1</td><td>G_IE7 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>G_IE7 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in G_IE7 bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_G_IE2,
CL_G_IE2</td><td>Sets/clears the G_IE2 bit.
<table>
<tr><th>ST_G_IE2</th><th>CL_G_IE2</th><th>Status of G_IE2 Bit</th></tr>
<tr><td>0</td><td>1</td><td>G_IE2 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>G_IE2 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in G_IE2 bit value.</td></tr>
</table></td></tr>
<tr><td>9, 1</td><td>ST_G_IE1,
CL_G_IE1</td><td>Sets/clears the G_IE1 bit.
<table>
<tr><th>ST_G_IE1</th><th>CL_G_IE1</th><th>Status of G_IE1 Bit</th></tr>
<tr><td>0</td><td>1</td><td>G_IE1 bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>G_IE1 bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in G_IE1 bit value.</td></tr>
</table></td></tr>
</table>

(5) CAN timer event enable register (CGTEN

The CGTEN register enables/disables the 4 timer events.
This register can read and written in 8-bit and 16-bit units.

Figure 14-14: CAN Timer Event Enable Register (CGTEN)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note1]	Initial value
CGTEN	0	0	0	0	0	0	0	0	0	0	0	0	CTEN3	CTEN2	CTEN1	CTEN0	1016H	0000H

Read

Bit Position	Bit Name	Function
3	CTEN3	Enables CAN timer event 3. 0: Timer event disabled 1: Timer event enabled
2	CTEN2	Enables CAN timer event 2. 0: Timer event disabled 1: Timer event enabled
1	CTEN1	Enables CAN timer event 1. 0: Timer event disabled 1: Timer event enabled
0	CTEN0	Enables CAN timer event 0. 0: Timer event disabled 1: Timer event enabled

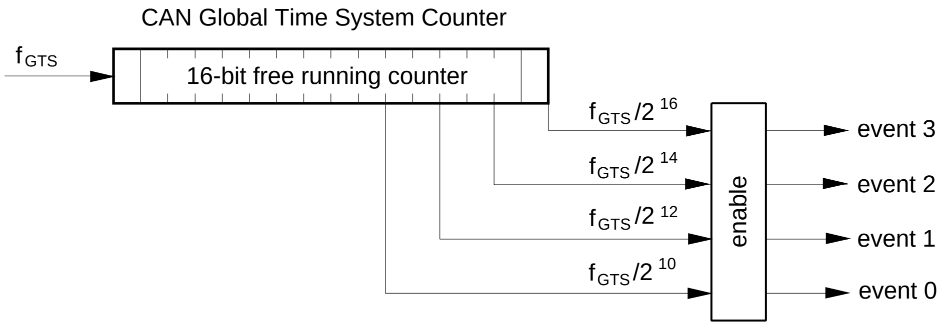
Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset.

2. Since there's no CAN bridge implemented in the V850E/CA2 device, the CGTEN register must not be written at any time. It's recommended to keep the reset value always.

The timer events are as follows:

- timer event 0: $f_{GTS} / 2^{10}$
- timer event 1: $f_{GTS} / 2^{12}$
- timer event 2: $f_{GTS} / 2^{14}$
- timer event 3: $f_{GTS} / 2^{16}$

Figure 14-15: CAN Global Time System Counter and event generation



(6) CAN global time system counter (CGTSC)

The CGTSC register holds the value of the free-running 16-bit CAN global time system counter. (For details refer to chapters 14.2.3 “Clock structure” on page 438 and 14.2.5 “Time stamp” on page 441)

This register can be read and written[Note 1] in 16-bit units only.

Figure 14-16: CAN Global Time System Counter (CGTSC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note2]	Initial value
CGTSC	TSC15	TSC14	TSC13	TSC12	TSC11	TSC10	TSC9	TSC8	TSC7	TSC6	TSC5	TSC4	TSC3	TSC2	TSC1	TSC0	1018H	0000H

Notes: 1. When writing is performed to CGTSC register, the counter is cleared to 0.

2. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: The CGTSC register can be read at any time.

(7) CAN message search start register (CGMSS)

The CGMSS register controls the start of a message search. It can be used for a fast message retrieval within the message buffers matching a search criteria (e.g. messages with DN flag set). This register is write-only and must be written in 16-bit units.

Figure 14-17: CAN Message Search Start Register (CGMSS)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note1]	Initial value
CGMSS	CIDE	CERQ	CTRQ	CMSK	CDN	SMN2	SMN1	SMN0	0	0	STRT5	STRT4	STRT3	STRT2	STRT1	STRT0	101AH	–

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15</td><td>CIDE</td><td>Search criteria for message identifier type (IDE).
0: Do not check status of the message identifier type.
1: Message identifier type must be standard identifier (IDE = 0).</td></tr>
<tr><td>14</td><td>CERQ</td><td>Search criteria for event processing request flag (ERQ) of the M_STATm registers.
0: Do not check status of the ERQ flag.
1: ERQ flag must be set.</td></tr>
<tr><td>13</td><td>CTRQ</td><td>Search criteria for transmit request flag (TRQ) and message ready flag (RDY) of the M_STATm registers.
0: Do not check status of TRQ flag and RDY flag.
1: TRQ flag and RDY flag must be set.</td></tr>
<tr><td>12</td><td>CMSK</td><td>Search criteria for the mask link bits MT2 to MT0 of the M_CONFm registers.
0: Do not check mask link bits.
1: Check only message buffers not linked with a mask.</td></tr>
<tr><td>11</td><td>CDN</td><td>Search criteria for data new flag (DN) of the M_STATm registers.
0: Do not check status of the DN flag.
1: DN flag must be set.</td></tr>
<tr><td>9, 8</td><td>SMN1,
SMN0</td><td>Specifies the CAN module number to search for.
<table>
<tr><th>SMN2</th><th>SMN1</th><th>SMN0</th><th>CAN Module Number</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Search for message buffers not linked to any CAN module.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Search for message buffers linked to CAN module 1</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Search for message buffers linked to CAN module 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Search for message buffers linked to CAN module 3[Note 2]</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Search for message buffers linked to CAN module 4[Note 2]</td></tr>
</table>
Remark: The SMNO2 to SMNO0 bits define which messages are checked by the message search. Only messages assigned to the CAN module defined by SMNO2 to SMNO0 are checked, all other messages are ignored.</td></tr>
<tr><td>5 to 0</td><td>STRT5 to
STRT0</td><td>Specifies the number of message buffer the search starts for. (0 to 31)
Remarks: 1. Any search will start from the message number defined by STRT5 to STRT0 and end at the highest available message buffer. If a search results in multiple matches, the lowest buffer number is returned.
2. To get the next match without modifying the search criteria the STRT5 to STRT0 bits must be set to the succeeding number of the found one in (MFND5 to MFND0) of the CGMSR register.</td></tr>
</table>

Notes: 1. The address of an interrupt pending register is calculated according to the following formula:
effective address = PP_BASE + address offset

2. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Remark: m = 00 to 31

(8) CAN message search result register (CGMSR)

The CGMSR register returns the result of a message search, started by writing the CGMSS register.

This register is read-only and can be read in 16-bit units.

Figure 14-18: CAN Message Search Result Register (CGMSR)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note1]	Initial value
CGMSR	0	0	0	0	0	0	MM	AM	0[Note3]	0[Note3]	MFND5	MFND4	MFND3	MFND2	MFND1	MFND0	101AH	0000H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>9, 8</td><td>MM,
AM</td><td>Indicates the match result of the preceding message search.
<table><tr><th>MM</th><th>AM</th><th>Number of Hits</th></tr><tr><td>×</td><td>0</td><td>No match</td></tr><tr><td>0</td><td>1</td><td>1 message meets the search criteria.</td></tr><tr><td>1</td><td>1</td><td>Several message meet the search criteria.[Note 2]</td></tr></table></td></tr>
<tr><td>5 to 0</td><td>MFND5 to
MFND0</td><td>Indicates the number of the message buffer, which was found by the message search. (0 to 31)[Note 2]
Remarks: 1. Any search will start from the message number defined by STRT5 to STRT0 and end at the highest available message buffer. If a search results in multiple matches, the lowest buffer number is returned.
2. To get the next match without modifying the search criteria the STRT5 to STRT0 bits must be set to the succeeding number of the found one in (MFND5 to MFND0) of the CGMSR register.</td></tr>
</table>

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. If a message search finds several message buffers meeting the search option, the MM flag is set. In that case the MFND5 to MFND0 bits return number of the message buffer with the lowest number.

3. Value of Bits 6 and 7 is undefined after search function.

(9) CAN test bus register (CTBR)

For test purposes an internal test bus is available. The CTBR register controls this test bus capability.
This register can be read and written in 8-bit and 16-bit units.

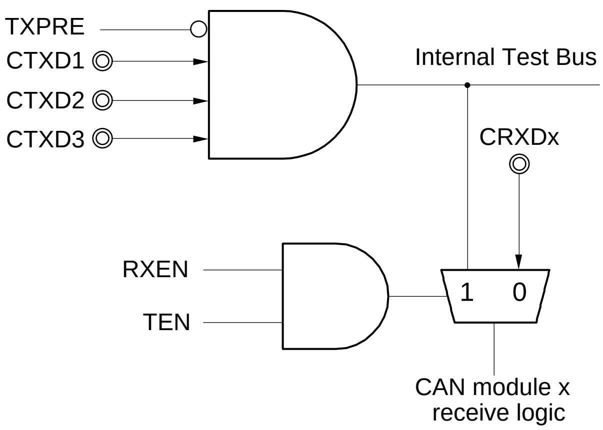
Figure 14-19: CAN Test Bus Register (CTBR)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CTBR	0	0	0	0	0	0	0	0	0	0	0	0	0	RXEN	TXPRE	TEN	101CH	0000H

Bit Position	Bit Name	Function
2	RXEN	Enables the receive line. 0: CAN module receive lines are input from the corresponding CANxRX pins. 1: CAN module receive lines are input from to the internal test bus.
1	TXPRE	Presets the transmit lines. 0: No preset on the transmit lines. 1: Error injection into the internal test bus by forcing all transmit pins to a dominant level.
	TEN	Enables internal test bus. 0: Internal test bus is disabled. 1: Internal test bus is enabled.

The figure below shows the structure of the internal CAN test bus.

Figure 14-20: Internal CAN Test Bus Structure



Remarks: **1.** Both, TEN bit and RXEN bit must be set (1) to use the internal CAN bus.

2. Using the internal CAN bus connects all CAN modules (CAN module 1 to CAN module 4[Note]) to one internal CAN bus. The internal CAN bus is used to operate the FCAN system without any external hardware (e.g. CAN transceiver, bus harness, etc.).

3. x = 1 to 4

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Caution: The internal test bus must only be used when none of the CAN modules are connected to a CAN bus.

14.3.3 CAN interrupt pending registers

(1) CAN interrupt pending register (CCINTP)

The CCINTP register summarizes all grouped interrupt pending signals. Each of them is assigned to an unambiguous interrupt vector of the V850E/CA2.

This register is read-only and can be read in 8-bit and16-bit units.

Figure 14-21: CAN Interrupt Pending Registers (CCINTPL, CCINTPH)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset Note 1	Initial value
CCINTPH	0	0	0	0	0	0	0	0	0	0	0	0	0	CAN4ERR	CAN4REC	CAN4TRX	1006H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset Note 1	Initial value
CCINTPL	INTACT	INTMAC	0	0	0	0	0	CAN3ERR	CAN3REC	CAN3TRX	CAN2ERR	CAN2REC	CAN2TRX	CAN1ERR	CAN1REC	CAN1TRX	1004H	0000H

Bit Position	Bit Name Note 2, 3	Function
2 (CCINTPH) 8, 5, 2 (CCINTPL)	CANxERR	Indicates an error interrupt of CAN module x (OR function of CxINT6 to CxINT2 bits of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
1 (CCINTPH) 7, 4, 1 (CCINTPL)	CANxREC	Indicates a receive completion interrupt of CAN module x (CxINT1 bit of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
0 (CCINTPH) 6, 3, 0 (CCINTPL)	CANxTRX	Indicates a transmit completion interrupt of CAN module x (CxINT0 bit of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
15 (CCINTPL)	INTACT	Indicates an interrupt of the CAN bridge (GINT7 bit of CGINTP register)Note 4. 0: No Interrupt pending 1: Interrupt pending
14 (CCINTPL)	INTMAC	Indicates a MAC interrupt (OR function of GINT3 to GINT1 bits of CGINTP register). 0: No Interrupt pending 1: Interrupt pending

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. x = 1 to 2 for the derivative µPD703128 (A).

3. x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1).

4. Due to the fact that there’s no bridge functionality implemented in the V850E/CA2 device, this bit isn’t relevant for the application.

Remark: The CCINTP register is a read-only register, which summarizes the CAN interrupt pending signals. Therefore it cannot be used to clear the interrupt pending signals after servicing. The interrupt pending signals must be cleared in the dedicated interrupt pending registers CGINTP, C1INTP, C2INTP and C3INTP.

(2) CAN global interrupt pending register (CGINTP)

The CGINTP register indicates the global interrupt pending signals. The interrupt pending flags can be cleared by writing to the register according to the special bit-clear method. (Refer to chapter 14.3.1 "Bit set/clear function" on page 452)
This register can be read in 8-bit and 16-bit units. It can be written in 16-bit units only.

Figure 14-22: CAN Global Interrupt Pending Register (CGINTP) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
CGINTP	0	0	0	0	0	0	0	0	GINT7	0	0	0	GINT3	GINT2	GINT1	0	1020H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CGINTP	0	0	0	0	0	0	0	0	CL_ GINT7	0	0	0	CL_ GINT3	CL_ GINT2	CL_ GINT1	0	1020H

Read

Bit Position	Bit Name	Function
7	GINT7	Indicates an interrupt of the CAN bridge ELISA (GINT7 bit of CGINTP register). 0: No Interrupt pending 1: Interrupt pending
3	GINT3	Indicates a wake-up interrupt from CAN sleep mode while clock supply to the FCAN system was stopped (ref. to CSTOP register). 0: No Interrupt pending 1: Interrupt pending
2	GINT2	Indicates an illegal address access interrupt. 0: No Interrupt pending 1: Interrupt pending **Remarks: 1.** Interrupt signals an illegal address access (refer to Figure 9-2). **2.** Interrupt signals a write access to temporary buffer while GOM bit of the CGST register is set (1).
1	GINT1	Indicates an invalid write access interrupt. 0: No Interrupt pending 1: Interrupt pending **Remarks: 1.** Interrupt signals a write access to a CAN module register while GOM bit of the CGST register is cleared (0). **2.** Interrupt signals an illegal FCAN system shut down, i.e. GOM bit is going to be cleared while at least one of the CAN modules is not in initialisation state.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-22: CAN Global Interrupt Pending Register (CGINTP) (2/2)

Write

Bit Position	Bit Name	Function
7	CL_GINT7	Clears the interrupt pending bit GINT7. 0: No change of GINT7 bit. 1: GINT7 bit is cleared (0).
3	CL_GINT3	Clears the interrupt pending bit GINT3. 0: No change of GINT3 bit. 1: GINT3 bit is cleared (0).
2	CL_GINT2	Clears the interrupt pending bit GINT2. 0: No change of GINT2 bit. 1: GINT2 bit is cleared (0).
1	CL_GINT1	Clears the interrupt pending bit GINT1. 0: No change of GINT1 bit. 1: GINT1 bit is cleared (0).

Remarks: **1.** The interrupts GINT1, GINT2 and GINT7 are only generated when the corresponding interrupt enable bit in the CGIE register is set.

2. In the CGIE register is no interrupt enable bit implemented for GINT3. Thus this interrupt cannot be disabled.

3. The interrupt pending bits must be cleared by software in the interrupt service routine.

Caution: **In case the interrupt pending bit is not cleared by software in the interrupt service routine, no subsequent interrupt is generated anymore.**

(3) CAN 1 to 4 interrupt pending registers (C1INTP to C4INTP)

The C1INTP to C4INTP registers indicate the corresponding CAN module interrupt pending signals. The interrupt pending flags can be cleared by writing to the registers according to the special bit-clear method. (Refer to chapter 14.3.1 “Bit set/clear function” on page 452)
This register can be read and written in 8-bit and16-bit units.

Figure 14-23: CAN 1 to 4 Interrupt Pending Registers (C1INTP to C4INTP) (1/2)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset Note 1	Initial value
C1INTP	0	0	0	0	0	0	0	0	0	C1INT6	C1INT5	C1INT4	C1INT3	C1INT2	C1INT1	C1INT0	1022H	0000H
C2INTP	0	0	0	0	0	0	0	0	0	C2INT6	C2INT5	C2INT4	C2INT3	C2INT2	C2INT1	C2INT0	1024H	0000H
C3INTP	0	0	0	0	0	0	0	0	0	C3INT6	C3INT5	C3INT4	C3INT3	C3INT2	C3INT1	C3INT0	1026H	0000H
C4INTP	0	0	0	0	0	0	0	0	0	C3INT6	C3INT5	C3INT4	C3INT3	C3INT2	C3INT1	C3INT0	1028H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1INTP	0	0	0	0	0	0	0	0	0	CL_ C1INT6	CL_ C1INT5	CL_ C1INT4	CL_ C1INT3	CL_ C1INT2	CL_ C1INT1	CL_ C1INT0	1022H
C2INTP	0	0	0	0	0	0	0	0	0	CL_ C2INT6	CL_ C2INT5	CL_ C2INT4	CL_ C2INT3	CL_ C2INT2	CL_ C2INT1	CL_ C2INT0	1024H
C3INTP	0	0	0	0	0	0	0	0	0	CL_ C3INT6	CL_ C3INT5	CL_ C3INT4	CL_ C3INT3	CL_ C3INT2	CL_ C3INT1	CL_ C3INT0	1026H
C4INTP	0	0	0	0	0	0	0	0	0	CL_ C3INT6	CL_ C3INT5	CL_ C3INT4	CL_ C3INT3	CL_ C3INT2	CL_ C3INT1	CL_ C3INT0	1028H

Read (1/2)

Bit Position	Bit Name **Note 2, 3**	Function
6	CxINT6	Indicates a CAN module x error. 0: No Interrupt pending 1: Interrupt pending
5	CxINT5	Indicates a CAN bus error of CAN module x. 0: No Interrupt pending 1: Interrupt pending
4	CxINT4	Indicates a wake-up from sleep mode of CAN module x. 0: No Interrupt pending 1: Interrupt pending
3	CxINT3	Indicates a error passive status on reception of CAN module x. 0: No Interrupt pending 1: Interrupt pending

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. x = 1 to 2 for the derivative µPD703128 (A).

3. x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1).

Figure 14-23: CAN 1 to 4 Interrupt Pending Registers (C1INTP to C4INTP) (2/2)

Read (2/2)

Bit Position	Bit Name **Note**	Function
2	CxINT2	Indicates a error passive or bus off status on transmission of CAN module x. 0: No Interrupt pending 1: Interrupt pending
1	CxINT1	Indicates a reception completion interrupt of CAN module x. 0: No Interrupt pending 1: Interrupt pending
0	CxINT0	Indicates a transmission completion interrupt of CAN module x. 0: No Interrupt pending 1: Interrupt pending

Write

Bit Position	Bit Name **Note 1, 2**	Function
6	CL_CxINT6	Clears the interrupt pending bit CxINT6. 0: No change of CxINT6 bit. 1: CxINT6 bit is cleared (0).
5	CL_CxINT5	Clears the interrupt pending bit CxINT5. 0: No change of CxINT5 bit. 1: CxINT5 bit is cleared (0).
4	CL_CxINT4	Clears the interrupt pending bit CxINT4. 0: No change of CxINT4 bit. 1: CxINT4 bit is cleared (0).
3	CL_CxINT3	Clears the interrupt pending bit CxINT3. 0: No change of CxINT3 bit. 1: CxINT3 bit is cleared (0).
2	CL_CxINT2	Clears the interrupt pending bit CxINT2. 0: No change of CxINT2 bit. 1: CxINT2 bit is cleared (0).
1	CL_CxINT1	Clears the interrupt pending bit CxINT1. 0: No change of CxINT1 bit. 1: CxINT1 bit is cleared (0).
0	CL_CxINT0	Clears the interrupt pending bit CxINT0. 0: No change of CxINT0 bit. 1: CxINT0 bit is cleared (0).

Notes: 1. x = 1 to 2 for the derivative µPD703128 (A).

2. x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1).

Remarks: 1. The interrupts CxINT1 to CxINT6 are only generated when the corresponding interrupt enable bit in the CGIE register is set.

2. The interrupt pending bits must be cleared by software in the interrupt service routine.

Caution: In case the interrupt pending bit is not cleared by software in the interrupt service routine, no subsequent interrupt is generated anymore.

14.3.4 CAN message buffer registers

(1) Message identifier registers L00 to L31 and H00 to H31 (M_IDL00 to M_IDL31, M_IDH00 to M_IDH31)

The M_IDLm, M_IDHm registers specify the identifier and format of the corresponding message m (m = 00 to 31).
These registers can be read/written 16-bit units.

Figure 14-24: Message Identifier Registers L00 to L31 and H00 to H31 (M_IDL00 to M_IDL31, M_IDH00 to M_IDH31)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
M_IDHm	IDE	0	0	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	ID17	ID16	812H +(m × 20H)	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
M_IDLm	ID15	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	810H +(m × 20H)	undef.

Bit Position	Bit Name	Function
15 (M_IDHm)	IDE	Specifies the format of message identifier. 0: Standard format mode (11-bit) 1: Extended format mode (29-bit)
12 to 0 (M_IDHm)	ID28 to ID16	When IDE = 0 (standard format): ID28 to ID18 specify the 11-bit identifier, where ID28 is the most significant bit. ID17, ID16 contain received data bits.[Note 2, 3] When IDE = 1 (extended format): ID28 to ID16 specify the 13 most significant bits of the 29-bit identifier, where ID28 is the most significant bit.
15 to 0 (M_IDLm)	ID15 to ID0	When IDE = 0 (standard format): ID15 to ID0 contain received data bits.[Note 2, 3] When IDE = 1 (extended format): ID15 to ID0 specify the 16 least significant bits of the 29-bit identifier.

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. In standard format mode (IDE = 0) these bits (ID17 to ID0) are only used for receive message buffers linked to a mask.
- Bits ID17 to ID10 storing the first data byte (D0) is stored, where ID17 is the MSB.
- Bits ID9 to ID2 storing the second data byte (D1), where ID9 is the MSB
- Bits ID1, ID0 contain the two most significant bits 7 and 6 of the third byte (D2)

3. When received message in standard format mode (IDE = 0) has less than 18 data bits, the values of the not received bits are undefined.

Remark: m = 00 to 31

(2) Message configuration registers 00 to 31 (M_CONF00 to M_CONF31)

The M_CONFm registers specify the message type, mask link and CAN module assignment of the corresponding message m (m = 00 to 31).

These registers can be read/written 8-bit units.

Figure 14-25: Message Configuration Registers 00 to 31 (M_CONF00 to M_CONF31) (1/2)

	7	6	5	4	3	2	1	0	Address Offset Note 1	Initial value
M_CONFm	0	0	MT2	MT1	MT0	MA2	MA1	MA0	814H + (m × 20H)	undef.

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>5 to 3</td><td>MT2 to MT0</td><td>Specifies the message type and mask link.
<table>
<tr><th>MT2</th><th>MT1</th><th>MT0</th><th>Message type and mask link</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Transmit message</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>Receive message, no mask linked</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>Receive message, mask 0 linked Note 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>Receive message, mask 1 linked Note 2</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>Receive message, mask 2 linked Note 2</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Receive message, mask 3 linked Note 1</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>Reserved Note 3</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>Receive message in diagnostic mode (type 7) Note 4</td></tr>
</table></td></tr>
</table>

Figure 14-25: Message Configuration Registers 00 to 31 (M_CONF00 to M_CONF31) (2/2)

Bit Position	Bit Name	Function			
1, 0	MA1, MA0	Assigns the message buffer to a CAN module.			
		MA2	MA1	MA0	CAN module assignment
		0	0	0	Message buffer is not assigned to a CAN module [Note 5]
		0	0	1	Message buffer is assigned to CAN module 1.
		0	1	0	Message buffer is assigned to CAN module 2
		0	1	1	Message buffer is assigned to CAN module 3[Note 6]
		1	0	0	Message buffer is assigned to CAN module 4[Note 6]
		1	0	1	Reserved
		1	1	0	Reserved
		1	1	1	Reserved

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. Mask number of the linked CAN module specified by MA1, MA0 bits.

3. CAN module does not handle a message buffer of this type.

4. A message buffer of this type is only handled if the linked CAN module is set to diagnostic mode. In this case all messages received on the CAN bus will be stored in this message buffer, regardless whether they could have been stored in other message buffers as well. Even the type of the identifier (standard or extended) and the type of the frame (remote or data frame) are not respected. In normal operation mode the message buffer is not handled.

5. If the message buffer is not assigned to a CAN module, it can be used as temporary buffer of the application or by the CAN bridge ELISA.

6. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Remark: m = 00 to 31

(3) Message status registers 00 to 31 (M_STAT00 to M_STAT31)

The M_STATm registers indicate transmit and receive status of the corresponding message m (m = 00 to 31). Bits can be set/cleared only by means of the SC_STATm register.
These registers can be read-only in 8-bit units.

Figure 14-26: Message Status Registers 00 to 31 (M_STAT00 to M_STAT31)

	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
M_STATm	0	0	0	0	ERQ	DN	TRQ	RDY	815H + (m × 20H)	undef.

Bit Position	Bit Name	Function
3	ERQ	Indicates a request for event processing by a CAN bridge[Note2] for this message. 0: No pending event processing. 1: Event processing is pending.
2	DN	Indicates new data received for this message. 0: No new message was received. 1: At least one new message was received. **Remarks: 1.** If the DN flag is set for a transmit message buffer, it indicates a remote frame reception. In case auto answering (RMDE0 bit of the M_CTRLm register) is active, the DN flag is cleared automatically after the answering data frame is sent. **2.** If the OVM bit of CxCTRL register is cleared (0), a message buffer assigned to the CAN module might be overwritten by new messages, although the DN flag is already set (x = 1 to 4[Note3]). Checking the MOVR bit of the M_CTRLm register additionally, indicates whether the message buffer has been overwritten. **3.** After copying a received message from the message buffer to the application memory, the DN flag has to be cleared (0) by software.
1	TRQ	Indicates a transmit request of this message. 0: No pending transmit request. 1: Transmit request is pending. **Remark:** If the TRQ flag is set for a receive message, a remote frame is sent. (refer to Table 9-16)
0	RDY	Enables and indicates application processing of this message. 0: Message is processed by the application, and not ready to be handled by the assigned CAN module. 1: Message is ready to be handled by the assigned CAN module. **Remark:** Transmit as well as receive messages are only handled by the assigned CAN module if the RDY flag is set. (refer to Table 9-16)

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. V850E/CA2 Jupiter has no CAN bridge implemented.

3. CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Remark: m = 00 to 31

Processing of a transmit or receive message by TRQ and RDY flags is summarized in 14-16.

Table 14-16: CAN Message Processing by TRQ and RDY Bits

Message Type	TRQ	RDY	Message Processing
Any	×	0	Message buffer is disabled for any processing by the assigned CAN module.
Receive message	0	1	Message buffer is ready for reception.
	1	1	Request for sending a remote frame.
Transmit message	0	1	No processing of the transmit message.
	1	1	Request for message transmission.

(4) Message set/clear status registers 00 to 31 (SC_STAT0 to SC_STAT31)

The SC_STATm registers set/clear the flags of the corresponding M_STATm registers (m = 00 to 31). By means of this register transmission can be requested and reception can be confirmed.

These registers can be written-only in 16-bit units.

Figure 14-27: Message Set/Clear Status Registers 00 to 31 (SC_STAT00 to SC_STAT31)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
SC_STATm	0	0	0	0	ST_ ERQ	ST_ DN	ST_ TRQ	ST_ RDY	0	0	0	0	CL_ ERQ	CL_ DN	CL_ TRQ	CL_ RDY	816H +(m × 20H)	–

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11, 3</td><td>ST_ERQ,
CL_ERQ</td><td>Sets/clears the ERQ bit of the M_STATm register.
<table>
<tr><th>ST_ERQ</th><th>CL_ERQ</th><th>Status of ERQ bit</th></tr>
<tr><td>0</td><td>1</td><td>ERQ bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>ERQ bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in ERQ bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_DN,
CL_DN</td><td>Sets/clears the DN bit of the M_STATm register.
<table>
<tr><th>ST_DN</th><th>CL_DN</th><th>Status of DN bit</th></tr>
<tr><td>0</td><td>1</td><td>DN bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DN bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DN bit value.</td></tr>
</table></td></tr>
<tr><td>9, 1</td><td>ST_TRQ,
CL_TRQ</td><td>Sets/clears the TRQ bit of the M_STATm register.
<table>
<tr><th>ST_TRQ</th><th>CL_TRQ</th><th>Status of TRQ bit</th></tr>
<tr><td>0</td><td>1</td><td>TRQ bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TRQ bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TRQ bit value.</td></tr>
</table></td></tr>
<tr><td>8, 0</td><td>ST_RDY,
CL_RDY</td><td>Sets/clears the RDY bit of the M_STATm register.
<table>
<tr><th>ST_RDY</th><th>CL_RDY</th><th>Status of RDY bit</th></tr>
<tr><td>0</td><td>1</td><td>RDY bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>RDY bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in RDY bit value.</td></tr>
</table></td></tr>
</table>

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: m = 00 to 31

(5) Message data registers m0 to m7 (M_DATAm0 to M_DATAm7) (m = 00 to 31)

The M_DATAm0 to M_DATAm7 registers are used to hold the receive or transmit data of the corresponding message m (m = 00 to 31).
These registers can be read/written in 8-bit units.

Figure 14-28: Message Data Registers m0 to m7 (M_DATAm0 to M_DATAm7) (m = 00 to 31) (1/2)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_DATA m0	D0_7	D0_6	D0_5	D0_4	D0_3	D0_2	D0_1	D0_0	808H + (m × 20H)	undef.
M_DATA m1	D1_7	D1_6	D1_5	D1_4	D1_3	D1_2	D1_1	D1_0	809H + (m × 20H)	undef.
M_DATA m2	D2_7	D2_6	D2_5	D2_4	D2_3	D2_2	D2_1	D2_0	80AH + (m × 20H)	undef.
M_DATA m3	D3_7	D3_6	D3_5	D3_4	D3_3	D3_2	D3_1	D3_0	80BH + (m × 20H)	undef.
M_DATA m4	D4_7	D4_6	D4_5	D4_4	D4_3	D4_2	D4_1	D4_0	80CH + (m × 20H)	undef.
M_DATA m5	D5_7	D5_6	D5_5	D5_4	D5_3	D5_2	D5_1	D5_0	80DH + (m × 20H)	undef.
M_DATA m6	D6_7	D6_6	D6_5	D6_4	D6_3	D6_2	D6_1	D6_0	80EH + (m × 20H)	undef.
M_DATA m7	D7_7	D7_6	D7_5	D7_4	D7_3	D7_2	D7_1	D7_0	80FH + (m × 20H)	undef.

Figure 14-28: Message Data Registers m0 to m7 (M_DATAm0 to M_DATAm7) (m = 00 to 31) (2/2)

Bit Position	Bit Name	Function
7 to 0 (M_DATAm0)	D0_7 to D0_0	Contents of the message data byte 0. (first message data byte)
7 to 0 (M_DATAm1)	D1_7 to D1_0	Contents of the message data byte 1.
7 to 0 (M_DATAm2)	D2_7 to D2_0	Contents of the message data byte 2.
7 to 0 (M_DATAm3)	D3_7 to D3_0	Contents of the message data byte 3.
7 to 0 (M_DATAm4)	D4_7 to D4_0	Contents of the message data byte 4.
7 to 0 (M_DATAm5)	D5_7 to D5_0	Contents of the message data byte 5.
7 to 0 (M_DATAm6)	D6_7 to D6_0	Contents of the message data byte 6.
7 to 0 (M_DATAm7)	D7_7 to D7_0	Contents of the message data byte 7.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: m = 00 to 31

Cautions:

1. **When transmitting data, only the number of bytes defined by the data length code (DLC) in the M_DLCm register are transmitted on the CAN bus. The transmission always starts with M_DATAm0.**
2. **If the ATS flag of the corresponding CxCTRL register is set (1) and the data length code (DLC) in the M_DLCm register is greater or equal 2, the last two bytes which are normally taken from the data part of the message buffer are ignored, and instead of these bytes a time stamp value is sent. (x = 1 to 2 for the derivative µPD703128 (A). x = 1 to 4 for the derivatives µPD703129 (A) and µPD703129 (A1)) (refer to chapter 14.2.5 "Time stamp" on page 441)**
3. **When a new message is received, all data bytes are updated, even if the data length code (DLC) in the M_DLCm register is less than 8. The values of the data bytes that have not been received may be change undefined.**

(6) Message data length code registers 00 to 31 (M_DLC0 to M_DLC31)

The M_DLCm registers specify the data length code (DLC) of the corresponding message m (m = 00 to 31). The DLC determines how many data bytes have to be transmitted, or received respectively, for the corresponding data frame.
These registers can be read/written in 8-bit units.

Figure 14-29: Message Data Length Code Registers 00 to 31 (M_DLC00 to M_DLC31)

	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
M_DLCm	RFU[Note 2]	RFU[Note 2]	RFU[Note 2]	RFU[Note 2]	DLC3	DLC2	DLC1	DLC0	804H + (m × 20H)	undef.

Bit Position	Bit Name	Function				
3 to 0	DLC3 to DLC0	Specifies the data length code of the transmit/receive message.				
		DLC3	DLC2	DLC1	DLC0	Data Length Code (DLC)
		0	0	0	0	No data bytes (0)
		0	0	0	1	1 data byte
		0	0	1	0	2 data bytes
		0	0	1	1	3 data bytes
		0	1	0	0	4 data bytes
		0	1	0	1	5 data bytes
		0	1	1	0	6 data bytes
		0	1	1	1	7 data bytes
		1	0	0	0	8 data bytes
		Others than above				Setting not recommended[Note 3]

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. RFU = Reserved for future use. Ensure to set these bits to 0 when writing to the M_DLCm register.

3. If a DLC is specified to a value greater 8 for a transmit message, 8-byte transfer is performed regardless of the DLC value.

Remark: m = 00 to 31

Cautions: 1. If a remote frame is received on a transmit buffer the DLC value leaves unchanged.

2. If a remote frame is received on a receive buffer the DLC value is updated by the DLC value of the remote frame.

(7) Message control registers 00 to 31 (M_CTRL0 to M_CTRL31)

The M_CTRLm registers control the behaviour on reception or transmission of the corresponding message buffer m (m = 00 to 31).
These registers can be read/written in 8-bit units.

Figure 14-30: Message Control Registers 00 to 31 (M_CTRL00 to M_CTRL31) (1/2)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_CTRL m	RMDE1	RMED0	ATS	IE	MOVR	R1	R0	RTR	805H + (m $\times$ 20H)	undef.

Bit Position	Bit Name	Function
7	RMED1	Specifies the remote frame handling mode 1. 0: DN flag is not changed, when receiving a remote frame. 1: DN flag is set (1), when receiving a remote frame. **Remark:** The remote frame handling mode 1 is only valid for transmit messages and indicates how the DN flag is updated if a remote frame is received on that message buffer. (For details refer to chapter 14.2.8 “Remote frame handling” on page 449)
6	RMED0	Specifies the remote frame handling mode 0. 0: Auto answering of remote frame is not active. 1: Auto answering of remote frame is active. **Remark:** The remote frame handling mode 0 is only valid for transmit messages and indicates how to respond if a remote frame is received on that message buffer. (For details refer to chapter 14.2.8 “Remote frame handling” on page 449)
5	ATS	Controls appending of the time stamp. 0: No time stamp appending. 1: Append time stamp (Only valid for transmit messages) **Remark:** This bit is only handled for transmit messages. If ATS is set (1) and the data length code (DLC) is greater or equal 2, the last two data bytes are replaced by the 16-bit time stamp. The appended time stamp is the capture value of the CAN global time system counter (CGTSC) on the SOF for this message. The last two data bytes defined in the data area are ignored. (For further details refer to chapter 14.2.5 “Time stamp” on page 441)

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remark: m = 00 to 31

Figure 14-30: Message Control Registers 00 to 31 (M_CTRL00 to M_CTRL31) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>4</td><td>IE</td><td>Enables message buffer m related interrupts.
0: Interrupts related to message buffer m disabled.
1: Interrupts related to message buffer m enabled.

Remark: If the message related interrupt is enabled, an interrupt is generated for any of the following conditions:
<table>
<tr><th>Condition</th><th>Related Interrupt</th></tr>
<tr><td>Data frame or remote frame is transmitted from transmit message buffer.</td><td>CANxTRX</td></tr>
<tr><td>Data frame or remote frame is received on receive message buffer.</td><td rowspan="2">CANxREC</td></tr>
<tr><td>Remote frame is received on transmit message without auto answering set (RMDE0 = 0).</td></tr>
</table>
An interrupt is not generated, even if enabled, for any of the following conditions:
<table>
<tr><th>Condition</th></tr>
<tr><td>Remote frame is received on a transmit message with auto answering mode (RMDE0 = 1).</td></tr>
<tr><td>Remote frame is transmitted from receive message buffer.</td></tr>
</table></td></tr>
<tr><td>3</td><td>MOVR</td><td>Indicates a message buffer overwrite.
0: No overwriting occurred.
1: Message buffer contents have been overwritten at least once since the DN flag of the M_STATm register has been cleared (0).
Remark: If the OVM bit of the CxCTRL register is cleared (0) a message buffer linked to this CAN module might be overwritten by new messages although the DN flag is already set. Checking the MOVR bit additionally, indicates whether the message buffer has been overwritten.</td></tr>
<tr><td>2</td><td>R1</td><td>Reserved bit (value of CAN bus bit r0 for receive message buffer)</td></tr>
<tr><td>1</td><td>R0</td><td>Reserved bit (value of CAN bus bit r1 for receive message buffer)</td></tr>
<tr><td>0</td><td>RTR</td><td>Specifies remote or data frame type of the message buffer.
0: Message received or to be sent is a data frame
1: Message received or to be sent is a remote frame.
Remark: When the RTR bit is set (1) for a transmit message, a remote frame is transmitted for the given identifier instead of a data frame. The RTR bit can be read for a receive message to determine whether a data frame or a remote frame was received.</td></tr>
</table>

Remarks: 1. m = 00 to 31

2. x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1).

3. x = 1 to 2 for the derivative µPD703128(A).

(8) Message time stamp registers 00 to 31 (M_TIME00 to M_TIME31)

The M_TIMEm registers store the captured time stamp value on reception of the corresponding message m (m = 00 to 31).

These registers can be read/written in 16-bit units.

Figure 14-31: Message Time Stamp Registers 00 to 31 (M_TIME00 to M_TIME31)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_TIMEm	TS15	TS14	TS13	TS12	TS11	TS19	TS9	TS8	TS7	TS6	TS5	TS4	TS3	TS2	TS1	TS0	806H + (m × 20H)	undef.

Bit Position	Bit Name	Function
15 to 0	TS15 to TS0	16-bit time stamp value captured on message reception. **Remark:** The trigger for the time stamp capture event is selected by the TMR flag of the CxCTRL register. (For details refer to chapter **11.2.5 Time stamp**)

Note: The address of a message time stamp register is calculated according to the following formula: effective address = PP_BASE + address offset

Remarks: **1.** m = 00 to 31

2. x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1).

3. x = 1 to 2 for the derivative µPD703128(A).

(9) Message event registers m0, m1, and m3 (M_EVTm0, M_EVTm1, M_EVTm3) (m = 00 to 31)

The message event registers M_EVTm0, MEVTm1, and M_EVTm3 imply the event pointers for event processing with a CAN bridge (m = 0 to 31).
These register can be read/written in 8-bit units.

Figure 14-32: Message Event Registers m0, m1, and m3 (M_EVTm0, M_EVTm1, M_EVTm2, M_EVTm3) (m = 00 to 31)

	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
M_EVTm0	PTR07	PTR06	PTR05	PTR04	PTR03	PTR02	PTR01	PTR00	800H + (m × 20H)	undef.
M_EVTm1	PTR17	PTR16	PTR15	PTR14	PTR13	PTR12	PTR11	PTR10	801H + (m × 20H)	undef.
M_EVTm2	PTR27	PTR26	PTR25	PTR24	PTR23	PTR22	PTR21	PTR20	802H + (m × 20H)	undef.
M_EVTm3	PTR37	PTR36	PTR35	PTR34	PTR33	PTR32	PTR31	PTR30	803H + (m × 20H)	undef.

Bit Position	Bit Name	Function
7 to 0 (M_EVTm0)	PTR07 to PTR00	8-bit event pointer 0 for processing with a CAN bridge.
7 to 0 (M_EVTm1)	PTR17 to PTR10	8-bit event pointer 1 for processing with a CAN bridge.
7 to 0 (M_EVTm2)	PTR27 to PTR20	8-bit event pointer 2 for processing with a CAN bridge.
7 to 0 (M_EVTm3)	PTR37 to PTR30	8-bit event pointer 3 for processing with a CAN bridge.

Remark: m = 00 to 31

Note: V850E/CA2 Jupiter has no CAN bridge implemented. Therefore the “Message Event Bytes” have no function. To avoid unexpected settings of the ERQ flag, it is recommended to initialize all “Message Event Bytes” with the value 0x00 at the first initialization and let that initialization unchanged always.

14.3.5 CAN Module Registers

(1) CAN 1 to 4[Note] mask 0 to 3 registers L, H (CxMASKL0 to CxMASKL3, CxMASKH0 to CxMASKH3) (x = 1 to 4[Note])

The CxMASKL0 to CxMASKL3, and CxMASKH0 to CxMASKH3 registers specify the four acceptance masks for each CAN module x (x = 1 to 4[Note]).
(For more details refer to chapter 14.2.7 "Mask handling" on page 448).
These registers can be read/written in 8-bit and 16-bit units.

Figure 14-33: CAN 1 to 4 Mask 0 to 3 Registers L, H (CxMASKL0 to CxMASKL3, CxMASKH0 to CxMASKH3) (x = 1 to 4)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset	Initial value
CxMASKHn	CMIDE	0	0	CMID28	CMID27	CMID26	CMID25	CMID24	CMID23	CMID22	CMID21	CMID20	CMID19	CMID18	CMID17	CMID16	see Table 14-17 on page 486	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset	Initial value
CxMASKLn	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0	see Table 14-17 on page 486	undef.

Bit Position	Bit Name	Function
15 (CxMASKHn)	CMIDE	Sets the CAN module mask option for the identifier type of the receive message. 0: Check identifier type of a received message. 1: Do not check identifier type. **Remark:** When CMIDE is cleared (0), the specified identifier type (standard or extended) of the message buffer linked to this CAN mask register must match the identifier type of the received message, in order to accept it for that message buffer.
12 to 0 (CxMASKHn) 15 to 0 (CxMASKLn)	CMID28 to CMID0	Sets the CAN module mask option for the corresponding identifier bit (ID28 to ID0) of the receive message. 0: Check identifier bit of a received message. 1: Do not check identifier bit. **Remarks: 1.** When CMIDn is cleared (0), the specified identifier bit of the message buffer linked to this CAN mask register must match the identifier bit of the received message, in order to accept it for that message buffer. **2.** When a receive message buffer is linked to a mask, always 29 bits of the specified identifier in the M_IDHm, M_IDLm registers of the message buffer are compared with the identifier of the received message, even if a standard format (11 identifier bits) is set. In case standard format identifier is selected (IDE = 0) the lower 18 bits in the M_IDm register contain a copy of data field bits, so that an address extensions by means of data field bits is possible. **3.** When a mask is exclusively intended for a standard format identifier the irrelevant mask bits CMID17 to CMID0 have to be set (1).

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only

Remarks: 1. n = 0 to 3 (mask number)

2. x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1).

3. x = 1 to 2 for the derivative µPD703128(A).

Table 14-17: Address Offsets of the CAN 1 to 4 Mask Registers

Symbol[Note1,2]	Address Offset[Note 3]			
	x = 1	x = 2	x = 3[Note 1, 2]	x = 4[Note 1, 2]
CxMASKL0	1040H	1080H	10C0H	1100H
CxMASKH0	1042H	1082H	10C2H	1102H
CxMASKL1	1044H	1084H	10C4H	1104H
CxMASKH1	1046H	1086H	10C6H	1106H
CxMASKL2	1048H	1088H	10C8H	1108H
CxMASKH2	104AH	108AH	10CAH	110AH
CxMASKL3	104CH	108CH	10CCH	110CH
CxMASKH3	104EH	108EH	10CEH	110EH

Notes: 1. CAN module number: x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1).

2. CAN module number: x = 1 to 2 for the derivative µPD703128(A).

3. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

(2) CAN 1 to 4 control registers (C1CTRL to C4CTRL)

The CxCTRL registers control the operating modes and indicate the operating status of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

These registers can be read in 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies (refer to chapter 14.3.1 "Bit set/clear function" on page 452).

Figure 14-34: CAN 1 to 4 Control Registers (C1CTRL to C4CTRL) (1/5)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	TPE	DLEVR	DLEVT	OVM	TMR	STOP	SLEEP	INIT	1050H	0101H
C2CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	TPE	DLEVR	DLEVT	OVM	TMR	STOP	SLEEP	INIT	1090H	0101H
C3CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	TPE	DLEVR	DLEVT	OVM	TMR	STOP	SLEEP	INIT	10D0H	0101H
C4CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	TPE	DLEVR	DLEVT	OVM	TMR	STOP	SLEEP	INIT	1110H	0101H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1CTRL	ST_TPE	ST_DLEVR	ST_DLEVT	ST_OVM	ST_TMR	ST_STOP	ST_SLEEP	ST_INIT	CL_TPE	CL_DLEVR	CL_DLEVT	CL_OVM	CL_TMR	CL_STOP	CL_SLEEP	CL_INIT	1050H
C2CTRL	ST_TPE	ST_DLEVR	ST_DLEVT	ST_OVM	ST_TMR	ST_STOP	ST_SLEEP	ST_INIT	CL_TPE	CL_DLEVR	CL_DLEVT	CL_OVM	CL_TMR	CL_STOP	CL_SLEEP	CL_INIT	1090H
C3CTRL	ST_TPE	ST_DLEVR	ST_DLEVT	ST_OVM	ST_TMR	ST_STOP	ST_SLEEP	ST_INIT	CL_TPE	CL_DLEVR	CL_DLEVT	CL_OVM	CL_TMR	CL_STOP	CL_SLEEP	CL_INIT	10D0H
C3CTRL	ST_TPE	ST_DLEVR	ST_DLEVT	ST_OVM	ST_TMR	ST_STOP	ST_SLEEP	ST_INIT	CL_TPE	CL_DLEVR	CL_DLEVT	CL_OVM	CL_TMR	CL_STOP	CL_SLEEP	CL_INIT	1110H

Figure 14-34: CAN 1 to 4 Control Registers (C1CTRL to C4CTRL) (2/5)

Read (1/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 14</td><td>TECS1,
TECS0</td><td>Indicates the transmission error counter status.
<table>
<tr><th>TECS1</th><th>TECS0</th><th>Transmission Error Counter Status</th></tr>
<tr><td>0</td><td>0</td><td>Transmission error counter below warning level (< 96)</td></tr>
<tr><td>0</td><td>1</td><td>Transmission error counter in warning level range (96 to 127)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved (not possible)</td></tr>
<tr><td>1</td><td>1</td><td>Transmission error counter above warning level (≥ 128)</td></tr>
</table></td></tr>
<tr><td>13, 12</td><td>RECS1,
RECS0</td><td>Indicates the reception error counter status.
<table>
<tr><th>RECS1</th><th>RECS0</th><th>Reception Error Counter Status</th></tr>
<tr><td>0</td><td>0</td><td>Reception error counter below warning level (< 96)</td></tr>
<tr><td>0</td><td>1</td><td>Reception error counter in warning level range (96 to 127)</td></tr>
<tr><td>1</td><td>0</td><td>Reserved (not possible)</td></tr>
<tr><td>1</td><td>1</td><td>Reception error counter in the error passive range (≥ 128)</td></tr>
</table></td></tr>
<tr><td>11</td><td>BOFF</td><td>Indicates a bus-off status of the CAN module.
0: CAN module is not in bus-off state (transmission error counter < 256)
1: CAN module is in bus-off state (transmission error counter = 256)</td></tr>
<tr><td>10</td><td>TSTAT</td><td>Indicates the transmission status.
0: No transmission activity on the CAN bus.
1: Transmission activity on the CAN bus.</td></tr>
<tr><td>9</td><td>RSTAT</td><td>Indicates the reception status.
0: No reception activity on the CAN bus.
1: Reception activity on the CAN bus.</td></tr>
<tr><td>8</td><td>ISTAT</td><td>Indicates the initialisation mode.
0: CAN module is in normal operation mode.
1: CAN module is stopped and set into initialisation mode (Reset value).
Remarks: 1. The ISTAT bit is set when the setting of the INIT bit is acknowledged by the CAN protocol layer. It is cleared automatically when the INIT bit is cleared.
2. In initialisation mode the level of the corresponding CAN transmit output is recessive (logical high).
3. Data manipulation of the CxSYNC and CxBRP registers is only possible during INIT state.
4. In INIT state the transmission and reception error counters are cleared and any error status is reset.</td></tr>
<tr><td>7</td><td>TPE</td><td>Indicates the transmit pin status.
0: CAN transmit pin is disabled (tri-state).
1: CAN transmit pin is enabled.</td></tr>
<tr><td>6</td><td>DLEVR</td><td>Specifies the dominant level of the CAN receive input pin.
0: Low level at the receive input is interpreted as a dominant bit (0).
1: High level at the receive input is interpreted as a dominant bit (0).
Remark: From software point of view a dominant bit is always a “0” value.</td></tr>
<tr><td>5</td><td>DLEVT</td><td>Specifies the dominant level of the CAN transmit output pin.
0: A dominant bit (0) results in a low level output.
1: A dominant bit (0) results in a high level output.
Remark: From software point of view a dominant bit is always a “0” value.</td></tr>
</table>

Figure 14-34: CAN 1 to 4 Control Registers (C1CTRL to C4CTRL) (3/5)

Read (2/2)

Bit Position	Bit Name	Function
4	OVM	Specifies the CAN message buffer overwrite mode. 0: A new CAN message overwrites a message buffer with DN flag set (1). 1: A new CAN message is discarded, if it would be stored in a message buffer with DN bit set (1). **Remark:** The OVM bit determines how to handle a receive message in case this message would overwrite the corresponding receive message buffer.
3	TMR	Specifies the time stamp mode for reception. 0: CGTSC counter is captured into the corresponding M_TIMEm register at SOF signal of the receive message. 1: CGTSC counter is captured into the corresponding M_TIMEm register, when the valid receive message is copied into the message buffer. **Remark:** For details refer to chapter 14.2.5 "Time stamp" on page 441
2	STOP	Selects the CAN stop mode. 0: CAN module is not stop mode. 1: CAN module stop mode selected. **Remarks: 1.** The CAN stop mode can be entered only if the CAN module is already in sleep mode (SLEEP = 1). **2.** In CAN stop mode the CAN module is disabled (protocol layer activities stopped, and set in suspend mode), and wake up of the CAN module is only possible by CPU (CPU clears STOP bit).
1	SLEEP	Selects the CAN sleep mode. 0: Normal operation mode. 1: CAN module sleep mode selected. **Remarks: 1.** Entering the CAN sleep mode from normal operating mode is just possible when the CAN bus is idle. **2.** In CAN sleep mode the CAN module does not process any transmit request submitted by the CPU. **3.** In case there is activity on the CAN bus and in parallel the SLEEP bit is set (1), the CAN module remains in normal operating mode and the SLEEP bit is cleared (0) automatically. **4.** The CAN sleep mode is released and normal operating mode is entered under the following conditions: (a) CPU clears the SLEEP bit (i.e. internal wake up by CPU) (b) first dominant bit on the idle CAN bus (i.e. external wake up by CAN bus activity) **5.** After releasing the CAN sleep mode the WAKE bit of the CxDEF register is set (1), and an error interrupt is generated upon external wake up by CAN bus activity.
0	INIT	Requests entering the initialisation mode. 0: Normal operation mode 1: Initialisation mode request **Remark:** The INIT flag is used to set the CAN module in initialisation mode. The CAN module acknowledges the transition into initialisation state by setting the ISTAT flag (1). This may take some time, especially when the protocol layer is handling a transmission or reception.

Figure 14-34: CAN 1 to 4 Control Registers (C1CTRL to C4CTRL) (4/5)

Write (1/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 7</td><td>ST_TPE,
CL_TPE</td><td>Sets/clears the TPE bit.
<table>
<tr><th>ST_TPE</th><th>CL_TPE</th><th>Status of TPE bit</th></tr>
<tr><td>0</td><td>1</td><td>TPE bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TPE bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TPE bit value.</td></tr>
</table></td></tr>
<tr><td>14, 6</td><td>ST_DLEVR,
CL_DLEVR</td><td>Sets/clears the DLEVR bit.
<table>
<tr><th>ST_DLEVR</th><th>CL_DLEVR</th><th>Status of DLEVR bit</th></tr>
<tr><td>0</td><td>1</td><td>DLEVR bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DLEVR bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DLEVR bit value.</td></tr>
</table></td></tr>
<tr><td>13, 5</td><td>ST_DLEVT,
CL_DLEVT</td><td>Sets/clears the DLEVT bit.
<table>
<tr><th>ST_DLEVT</th><th>CL_DLEVT</th><th>Status of DLEVT bit</th></tr>
<tr><td>0</td><td>1</td><td>DLEVT bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DLEVT bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DLEVT bit value.</td></tr>
</table></td></tr>
<tr><td>12, 4</td><td>ST_OVM,
CL_OVM</td><td>Sets/clears the OVM bit.
<table>
<tr><th>ST_OVM</th><th>CL_OVM</th><th>Status of OVM bit</th></tr>
<tr><td>0</td><td>1</td><td>OVM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>OVM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in OVM bit value.</td></tr>
</table></td></tr>
<tr><td>11, 3</td><td>ST_TMR,
CL_TMR</td><td>Sets/clears the TMR bit.
<table>
<tr><th>ST_TMR</th><th>CL_TMR</th><th>Status of TMR bit</th></tr>
<tr><td>0</td><td>1</td><td>TMR bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>TMR bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in TMR bit value.</td></tr>
</table></td></tr>
<tr><td>10, 2</td><td>ST_STOP,
CL_STOP</td><td>Sets/clears the STOP bit.
<table>
<tr><th>ST_STOP</th><th>CL_STOP</th><th>Status of STOP bit</th></tr>
<tr><td>0</td><td>1</td><td>STOP bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>STOP bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in STOP bit value.</td></tr>
</table></td></tr>
</table>

Figure 14-34: CAN 1 to 4 Control Registers (C1CTRL to C4CTRL) (5/5)

Write (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>9, 1</td><td>ST_SLEEP,
CL_SLEEP</td><td>Sets/clears the SLEEP bit.
<table>
<tr><th>ST_SLEEP</th><th>CL_SLEEP</th><th>Status of SLEEP bit</th></tr>
<tr><td>0</td><td>1</td><td>SLEEP bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>SLEEP bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in SLEEP bit value.</td></tr>
</table></td></tr>
<tr><td>8, 0</td><td>ST_INIT,
CL_INIT</td><td>Sets/clears the INIT bit.
<table>
<tr><th>ST_INIT</th><th>CL_INIT</th><th>Status of INIT bit</th></tr>
<tr><td>0</td><td>1</td><td>INIT bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>INIT bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in INIT bit value.</td></tr>
</table></td></tr>
</table>

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset.

(3) CAN 1 to 4 definition registers (C1DEF to C4DEF)

The CxDEF registers define normal and diagnostic operation and indicate CAN bus error and states of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

These registers can be read in 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies (refer to chapter 14.3.1 "Bit set/ clear function" on page 452).

Figure 14-35: CAN 1 to 4 Definition Registers (C1DEF to C4DEF) (1/4)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	1052H	0000H
C2DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	1092H	0000H
C3DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	10D2H	0000H
C4DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	1112H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1DEF	ST_DGM	ST_MOM	ST_SSHT	ST_PBB	ST_BERR	ST_VALID	ST_WAKE	ST_OVR	CL_DGM	CL_MOM	CL_SSHT	CL_PBB	CL_BERR	CL_VALID	CL_WAKE	CL_OVR	1052H
C2DEF	ST_DGM	ST_MOM	ST_SSHT	ST_PBB	ST_BERR	ST_VALID	ST_WAKE	ST_OVR	CL_DGM	CL_MOM	CL_SSHT	CL_PBB	CL_BERR	CL_VALID	CL_WAKE	CL_OVR	1092H
C3DEF	ST_DGM	ST_MOM	ST_SSHT	ST_PBB	ST_BERR	ST_VALID	ST_WAKE	ST_OVR	CL_DGM	CL_MOM	CL_SSHT	CL_PBB	CL_BERR	CL_VALID	CL_WAKE	CL_OVR	10D2H
C4DEF	ST_DGM	ST_MOM	ST_SSHT	ST_PBB	ST_BERR	ST_VALID	ST_WAKE	ST_OVR	CL_DGM	CL_MOM	CL_SSHT	CL_PBB	CL_BERR	CL_VALID	CL_WAKE	CL_OVR	1112H

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-35: CAN 1 to 4 Definition Registers (C1DEF to C4DEF) (2/4)

Read (1/2)

Bit Position	Bit Name	Function
7	DGM	Specifies the storage of receive message in diagnostic mode. 0: receive only and store valid message in message buffer type 7. 1: receive only and store valid message as in normal operation mode **Remarks: 1.** The settings of the DGM bit are only effective in diagnostic mode (MOM = 1). In normal operation mode (MOM = 0) the DGM bit settings have no meaning. **2.** When the “diagnostic mode” is active, a valid reception is indicated by setting the VALID flag and depending on the setting on the setting of the DGM flag, storing valid data either in a message buffer of buffer type 7 or in the same way as in normal operation mode. The CAN protocol layer does not send an acknowledge, error frame or transmit message, and also the error counter does not count.
6	MOM	Defines the module operating mode. 0: Normal operating mode 1: Diagnostic mode **Remarks: 1.** The diagnostic mode provides the following functional behavior: (a) Transmission of data frames and remote frames is not possible. (b) No acknowledge is generated upon reception of a valid message. (c) On reception of a valid message the VALID flag is set (1). (d) Receive and transmit error counters remain unchanged on errors. **2.** The diagnostic mode can be used for baud rate detection and diagnostic purposes. **Caution: When the diagnostic mode (MOM = 1) is defined for a CAN module, the CxBRP register is only accessible in the initialisation state (ISTAT = 1). While ISTAT is cleared (0) write access to the CxBRP is prohibited and reading the address of the CxBRP register returns the status of the CxDINF register.**
5	SSHT	Defines the single-shot mode for a CAN module. 0: Normal operating mode 1: Single-shot mode **Remarks: 1.** In single shot mode the CAN module tries to transmit a message only once, and the TRQ flag of the corresponding message is cleared regardless whether the transmission was successful (no error occurred), or not. **2.** In case of an error frame caused during a transmission in single-shot mode, the CAN module does not launch a re-transmission. However, error management according to the CAN Protocol is executed (i.e. generation of error interrupt, incrementing of error counters). **3.** The CPU can switch between the normal operating mode and the single-shot mode while the CAN module is active without causing any error on the CAN bus. **Caution: According to the CAN protocol upon a loss of arbitration a transmitter attempts to re-transmit the message, though loss of arbitration is not defined as an error. When single shot mode is set (SSHT = 1), a loss of arbitration is signaled by setting the BERR flag (1). Since the BERR flag signals a bus error in normal operation, the user must check it in conjunction with the values of the error counter, in order to judge whether it was caused by an error or a loss of arbitration.**

Figure 14-35: CAN 1 to 4 Definition Registers (C1DEF to C4DEF) (3/4)

Read (2/2)

Bit Position	Bit Name	Function
4	PBB	Defines the priority by message buffer numbers. 0: Transmission priority is given by message identifier. 1: Transmission priority is given by the number of the message buffer. **Remark:** Normally the message identifier defines the transmission priority. If the PBB flag is set, the location of a message defines the priority – the lower the message buffer number the higher the transmission priority.
3	BERR	Indicates a CAN bus error. 0: No CAN bus error occurred since the bit was cleared last. 1: At least one CAN bus error occurred since the flag has been cleared last. **Remark:** For single shot mode (SSHT bit = 1) this flag indicates a loss of the arbitration.
2	VALID	Indicates valid protocol activity. 0: No valid message was detected by the CAN protocol layer. 1: At least one valid message was received on the CAN bus since the flag has been cleared last.
1	WAKE	Indicates the wake-up condition from CAN sleep mode. 0: No wake-up, or sleep mode has been terminated by CPU (normal operation). 1: CAN sleep mode has been terminated by detection of CAN bus activity.
0	OVR	Indicates an overrun error. 0: No overrun (normal operation) 1: An overrun occurred during access to the CAN memory. **Remark:** The OVR flag is set, if the CAN message handler is not able to scan all the message areas defined for the CAN module due to timing problems. The error interrupt CxINT6 is generated at the same time. Possible cause for an overrun situation: The CAN memory access clock f_{MEM} selection is too slow for the selected CAN baud rate.

Figure 14-35: CAN 1 to 4 Definition Registers (C1DEF to C4DEF) (4/4)

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>15, 7</td><td>ST_DGM,
CL_DGM</td><td>Sets/clears the DGM bit.
<table>
<tr><th>ST_DGM</th><th>CL_DGM</th><th>Status of DGM bit</th></tr>
<tr><td>0</td><td>1</td><td>DGM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>DGM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in DGM bit value.</td></tr>
</table></td></tr>
<tr><td>14, 6</td><td>ST_MOM,
CL_MOM</td><td>Sets/clears the MOM bit.
<table>
<tr><th>ST_MOM</th><th>CL_MOM</th><th>Status of MOM bit</th></tr>
<tr><td>0</td><td>1</td><td>MOM bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>MOM bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in MOM bit value.</td></tr>
</table></td></tr>
<tr><td>13, 5</td><td>ST_SSHT,
CL_SSHT</td><td>Sets/clears the SSHT bit.
<table>
<tr><th>ST_SSHT</th><th>CL_SSHT</th><th>Status of SSHT bit</th></tr>
<tr><td>0</td><td>1</td><td>SSHT bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>SSHT bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in SSHT bit value.</td></tr>
</table></td></tr>
<tr><td>12, 4</td><td>ST_PPB,
CL_PPB</td><td>Sets/clears the PPB bit.
<table>
<tr><th>ST_PPB</th><th>CL_PPB</th><th>Status of PPB bit</th></tr>
<tr><td>0</td><td>1</td><td>PPB bit is cleared (0).</td></tr>
<tr><td>1</td><td>0</td><td>PPB bit is set (1).</td></tr>
<tr><td colspan="2">Others</td><td>No change in PPB bit value.</td></tr>
</table></td></tr>
<tr><td>3</td><td>CL_BERR</td><td>Clears the BERR bit.
0: No change of BERR bit.
1: BERR bit is cleared (0).</td></tr>
<tr><td>2</td><td>CL_VALID</td><td>Clears the VALID bit.
0: No change of VALID bit.
1: VALID bit is cleared (0).</td></tr>
<tr><td>1</td><td>CL_WAKE</td><td>Clears the WAKE bit.
0: No change of WAKE bit.
1: WAKE bit is cleared (0).</td></tr>
<tr><td>0</td><td>CL_OVR</td><td>Clears the OVR bit.
0: No change of OVR bit.
1: OVR bit is cleared (0).</td></tr>
</table>

(4) CAN 1 to 4 information registers (C1LAST to C4LAST)

The CxLAST registers return the number of the last received message and last CAN protocol error of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

These registers can be read-only in 8-bit and 16-bit units.

Figure 14-36: CAN 1 to 4 Information Registers (C1LAST to C4LAST)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset Note 1	Initial value
C1LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	1054H	00FFH
C2LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	1094H	00FFH
C3LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	10D4H	00FFH
C4LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	1114H	00FFH

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>11 to 8</td><td>LERR3 to LERR0</td><td>Holds the code of the last CAN protocol error.
<table>
<tr><th>LERR3</th><th>LERR2</th><th>LERR1</th><th>LERR0</th><th>Code of Last CAN Protocol Error</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>No error (reset state only)</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>CAN bus bit error</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>CAN bus stuff error</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>CAN bus CRC error</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>CAN bus form error</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>CAN bus acknowledgement error</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>CAN bus arbitration lost Note 2</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>CAN module overrun error</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>CAN wake-up from CAN bus</td></tr>
<tr><td colspan="4">Others than above</td><td>Reserved</td></tr>
</table>
Remark: The LERR3 to LERR0 bits cannot be cleared. Thus these bits remain unchanged until the next error occurs.</td></tr>
<tr><td>7 to 0</td><td>LREC7 to LREC0</td><td>Holds the message buffer number of the last received message.
<table>
<tr><th>LREC7 to LREC0</th><th>Receive Message Buffer Number</th></tr>
<tr><td>0 to 31</td><td>Message buffer number of the last received message</td></tr>
<tr><td>32 to 255</td><td>Reserved (not possible)</td></tr>
</table></td></tr>
</table>

Notes: 1. The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

2. This error code only occurs in single-shot mode (SSHT bit of the CxDEF register = 1).

(5) CAN 1 to 4 error counter registers (C1ERC to C4ERC)

The CxERC registers reflect the status of the transmit and the receive error counters of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

These registers can be read-only in 8-bit and 16-bit units.

Figure 14-37: CAN 1 to 4 Error Counter Registers (C1ERC to C4ERC)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	1056H	0000H
C2ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	1096H	0000H
C3ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	10D6H	0000H
C4ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	1116H	0000H

Bit Position	Bit Name	Function
15 to 8	REC7 to REC0	The receive error counter (REC) holds the status of the error counter of reception errors as defined in the CAN protocol.
7 to 0	TEC7 to TEC0	The transmit error counter (TEC) holds the status of the error counter of transmission errors as defined in the CAN protocol.

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

(6) CAN 1 to 4 interrupt enable registers (C1IE to C4IE)

The CxIE registers enable the transmit, receive and error interrupts of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

These registers can be read in 8-bit and 16-bit units. It can be written in 16-bit units only. For setting and clearing certain bits a special set/clear method applies (refer to chapter 14.3.1 "Bit set/ clear function" on page 452).

Figure 14-38: CAN 1 to 4 Interrupt Enable Registers (C1IE to C4IE) (1/3)

Read	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1IE	0	0	0	0	0	0	0	0	0	E_INT6	E_INT5	E_INT4	E_INT3	E_INT2	E_INT1	E_INT0	1058H	0000H
C2IE	0	0	0	0	0	0	0	0	0	E_INT6	E_INT5	E_INT4	E_INT3	E_INT2	E_INT1	E_INT0	1098H	0000H
C3IE	0	0	0	0	0	0	0	0	0	E_INT6	E_INT5	E_INT4	E_INT3	E_INT2	E_INT1	E_INT0	10D8H	0000H
C4IE	0	0	0	0	0	0	0	0	0	E_INT6	E_INT5	E_INT4	E_INT3	E_INT2	E_INT1	E_INT0	1118H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
C1IE	0	ST_E_INT6	ST_E_INT5	ST_E_INT4	ST_E_INT3	ST_E_INT2	ST_E_INT1	ST_E_INT0	0	CL_E_INT6	CL_E_INT5	CL_E_INT4	CL_E_INT3	CL_E_INT2	CL_E_INT1	CL_E_INT0	1058H
C2IE	0	ST_E_INT6	ST_E_INT5	ST_E_INT4	ST_E_INT3	ST_E_INT2	ST_E_INT1	ST_E_INT0	0	CL_E_INT6	CL_E_INT5	CL_E_INT4	CL_E_INT3	CL_E_INT2	CL_E_INT1	CL_E_INT0	1098H
C3IE	0	ST_E_INT6	ST_E_INT5	ST_E_INT4	ST_E_INT3	ST_E_INT2	ST_E_INT1	ST_E_INT0	0	CL_E_INT6	CL_E_INT5	CL_E_INT4	CL_E_INT3	CL_E_INT2	CL_E_INT1	CL_E_INT0	10D8H
C4IE	0	ST_E_INT6	ST_E_INT5	ST_E_INT4	ST_E_INT3	ST_E_INT2	ST_E_INT1	ST_E_INT0	0	CL_E_INT6	CL_E_INT5	CL_E_INT4	CL_E_INT3	CL_E_INT2	CL_E_INT1	CL_E_INT0	1118H

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-38: CAN 1 to 4 Interrupt Enable Registers (C1IE to C4IE) (2/3)

Read

Bit Position	Bit Name	Function
6	E_INT6	Enables CAN module error interrupt (INT6). 0: Interrupt disabled 1: Interrupt enabled
5	E_INT5	Enables CAN bus error interrupt (INT5). 0: Interrupt disabled 1: Interrupt enabled
4	E_INT4	Enables wake-up from CAN sleep mode interrupt (INT4). 0: Interrupt disabled 1: Interrupt enabled
3	E_INT3	Enables interrupt for error passive on reception(INT3). 0: Interrupt disabled 1: Interrupt enabled
2	E_INT2	Enables interrupt for error passive or bus-off on transmission (INT2). 0: Interrupt disabled 1: Interrupt enabled
1	E_INT1	Enables reception successful completion interrupt (INT1). 0: Interrupt disabled 1: Interrupt enabled
0	E_INT0	Enables transmission completion interrupt (INT0). 0: Interrupt disabled 1: Interrupt enabled

Figure 14-38: CAN 1 to 4 Interrupt Enable Registers (C1IE to C4IE) (3/3)

Write

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>14, 6</td><td>ST_E_INT6
CL_E_INT6</td><td>Sets/clears the E_INT6 bit.<table><tr><th>ST_E_INT6</th><th>CL_E_INT6</th><th>Status of E_INT6 bit</th></tr><tr><td>0</td><td>1</td><td>E_INT6 bit is cleared (0).</td></tr><tr><td>1</td><td>0</td><td>E_INT6 bit is set (1).</td></tr><tr><td colspan="2">Others</td><td>No change in E_INT6 bit value.</td></tr></table></td></tr>
<tr><td>13, 5</td><td>ST_E_INT5
CL_E_INT5</td><td>Sets/clears the E_INT5 bit.<table><tr><th>ST_E_INT5</th><th>CL_E_INT5</th><th>Status of E_INT5 bit</th></tr><tr><td>0</td><td>1</td><td>E_INT5 bit is cleared (0).</td></tr><tr><td>1</td><td>0</td><td>E_INT5 bit is set (1).</td></tr><tr><td colspan="2">Others</td><td>No change in E_INT5 bit value.</td></tr></table></td></tr>
<tr><td>12, 4</td><td>ST_E_INT4
CL_E_INT4</td><td>Sets/clears the E_INT4 bit.<table><tr><th>ST_E_INT4</th><th>CL_E_INT4</th><th>Status of E_INT4 bit</th></tr><tr><td>0</td><td>1</td><td>E_INT4 bit is cleared (0).</td></tr><tr><td>1</td><td>0</td><td>E_INT4 bit is set (1).</td></tr><tr><td colspan="2">Others</td><td>No change in E_INT4 bit value.</td></tr></table></td></tr>
<tr><td>11, 3</td><td>ST_E_INT3
CL_E_INT3</td><td>Sets/clears the E_INT3 bit.<table><tr><th>ST_E_INT3</th><th>CL_E_INT3</th><th>Status of E_INT3 bit</th></tr><tr><td>0</td><td>1</td><td>E_INT3 bit is cleared (0).</td></tr><tr><td>1</td><td>0</td><td>E_INT3 bit is set (1).</td></tr><tr><td colspan="2">Others</td><td>No change in E_INT3 bit value.</td></tr></table></td></tr>
<tr><td>10, 2</td><td>ST_E_INT2
CL_E_INT2</td><td>Sets/clears the E_INT2 bit.<table><tr><th>ST_E_INT2</th><th>CL_E_INT2</th><th>Status of E_INT2 bit</th></tr><tr><td>0</td><td>1</td><td>E_INT2 bit is cleared (0).</td></tr><tr><td>1</td><td>0</td><td>E_INT2 bit is set (1).</td></tr><tr><td colspan="2">Others</td><td>No change in E_INT2 bit value.</td></tr></table></td></tr>
<tr><td>9, 1</td><td>ST_E_INT1
CL_E_INT1</td><td>Sets/clears the E_INT1 bit.<table><tr><th>ST_E_INT1</th><th>CL_E_INT1</th><th>Status of E_INT1 bit</th></tr><tr><td>0</td><td>1</td><td>E_INT1 bit is cleared (0).</td></tr><tr><td>1</td><td>0</td><td>E_INT1 bit is set (1).</td></tr><tr><td colspan="2">Others</td><td>No change in E_INT1 bit value.</td></tr></table></td></tr>
<tr><td>8, 0</td><td>ST_E_INT0,
CL_E_INT0</td><td>Sets/clears the E_INT0 bit.<table><tr><th>ST_E_INT0</th><th>CL_E_INT0</th><th>Status of E_INT0 bit</th></tr><tr><td>0</td><td>1</td><td>E_INT0 bit is cleared (0).</td></tr><tr><td>1</td><td>0</td><td>E_INT0 bit is set (1).</td></tr><tr><td colspan="2">Others</td><td>No change in E_INT0 bit value.</td></tr></table></td></tr>
</table>

(7) CAN 1o 4 bus activity registers (C1BA to C4BA)

The CxBA registers indicate the status of the CAN bus activities of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

These registers can be read-only in 8-bit and 16-bit units.

Figure 14-39: CAN 1 to 4 Bus Activity Registers (C1BA to C4BA) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	105AH	00FFH
C2BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	109AH	00FFH
C3BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	10DAH	00FFH
C4BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	111AH	00FFH

Bit Position	Bit Name	Function
12 to 8	CACT4 to CACT0	Indicates the CAN module activity. (see table below) **Remark:** The CACT4 to CACT0 bits reflect the status of the CAN protocol layer.

CACT4	CACT3	CACT2	CACT1	CACT0	CAN Module Activity
0	0	0	0	0	Reset state
0	0	0	0	1	Waiting for bus idle
0	0	0	1	0	Bus idle
0	0	0	1	1	Start of frame (SOF)
0	0	1	0	0	Standard format ID section
0	0	1	0	1	Data length code section
0	0	1	1	0	Data field section
0	0	1	1	1	CRC field section
0	1	0	0	0	CRC delimiter
0	1	0	0	1	Acknowledge slot
0	1	0	1	0	Acknowledge delimiter
0	1	0	1	1	End of frame section (EOF)
0	1	1	0	0	Intermission state
0	1	1	0	1	Suspend transmission
0	1	1	1	0	Error frame
0	1	1	1	1	Waiting for error delimiter
1	0	0	0	0	Error delimiter
1	0	0	0	1	Error bus-off
1	0	0	1	0	Extended format ID section
1	0	0	1	1	Suspend mode
Others than above					Reserved

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-39: CAN 1 to 4 Bus Activity Registers (C1BA to C4BA) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>7 to 0</td><td>TMNO7 to TMNO0</td><td>Indicates the message buffer, which is either waiting to be transmitted or in transmission progress.
<table>
<tr><th>TMNO7 to TMNO0</th><th>Number of Transmit Message Buffer</th></tr>
<tr><td>0 to 31</td><td>Current transmit message buffer (waiting for transmission, or in transmission progress)</td></tr>
<tr><td>32 to 254</td><td>Reserved (not possible)</td></tr>
<tr><td>255</td><td>No message waiting for transmission, or in transmission progress.</td></tr>
</table>
</td></tr>
</table>

(8) CAN 1 to 4 bit rate prescaler registers (C1BRP to C4BRP)

The CxBRP registers specify the bit rate prescaler and CAN bus speed of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

The register layout depends on the TLM bit (bit 15), and distinguishes between 6-bit prescaler (TLM bit = 0) and 8-bit prescaler (TLM bit = 1).

These registers can be read/written in 8-bit and 16-bit units. However, write access is only permitted in initialisation mode (ISTAT bit of the CxCTRL register = 1)

Figure 14-40: CAN 1 to 4 Bit Rate Prescaler Registers (C1BRP to C4BRP) (1/2)

TLM = 0	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	105CH	0000H
C2BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	109CH	0000H
C3BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	10DCH	0000H
C4BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	111CH	0000H

TLM = 1	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	105CH	
C2BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	109CH	
C3BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	10DCH	
C4BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	111CH	

Bit Position	Bit Name	Function
15	TLM	Specifies the transfer layer mode. 0: Transfer layer uses 6-bit prescaler setting. 1: Transfer layer uses 8-bit prescaler setting.
8 (TLM = 1) 6 (TLM = 0)	BTYPE	Specifies the CAN bus type. 0: CAN bus type is low speed bus (≤ 125 kbps) 1: CAN bus type is high speed bus (> 125 kbps)

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Remarks: **1.** Writing to this register is only possible if CAN module is set into initialisation mode.

2. CPU can read CxBRP register at any time.

Caution: **In diagnostic mode the CxBRP register is hidden, and the CxDINF register appears instead of it at the same address.**

Figure 14-40: CAN 1 to 4 Bit Rate Prescaler Registers (C1BRP to C4BRP) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr>
<td>7 to 0
(TLM = 1)

5 to 0
(TLM = 0)</td>
<td>BRP7 to
BRP0
(TLM = 1)

BRP5 to
BRP0
(TLM = 0)</td>
<td>Specifies the bit rate prescaler for the CAN protocol layer.

TLM = 0:
<table>
<tr><th>BRP5</th><th>BRP4</th><th>BRP3</th><th>BRP2</th><th>BRP1</th><th>BRP0</th><th>Bit Rate Prescaler
$f_{BTL} = f_{MEM} / 2\times(k+1)$</th><th>k</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>$f_{BTL} = f_{MEM} / 2$</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>$f_{BTL} = f_{MEM} / 4$</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>$f_{BTL} = f_{MEM} / 6$</td><td>2</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>$f_{BTL} = f_{MEM} / 8$</td><td>3</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>$f_{BTL} = f_{MEM} / 10$</td><td>4</td></tr>
<tr><td colspan="6">.
.
.</td><td>.
.
.</td><td>.
.
.</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>$f_{BTL} = f_{MEM} / 126$</td><td>62</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>$f_{BTL} = f_{MEM} / 128$</td><td>63</td></tr>
</table>
TLM = 1:
<table>
<tr><th>BRP7</th><th>BRP6</th><th>BRP5</th><th>BRP4</th><th>BRP3</th><th>BRP2</th><th>BRP1</th><th>BRP0</th><th>Bit Rate Prescaler
$f_{BTL} = f_{MEM} / (k+1)$</th><th>k</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>$f_{BTL} = f_{MEM}$</td><td>0</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>$f_{BTL} = f_{MEM} / 2$</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>$f_{BTL} = f_{MEM} / 3$</td><td>2</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>$f_{BTL} = f_{MEM} / 4$</td><td>3</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>$f_{BTL} = f_{MEM} / 5$</td><td>4</td></tr>
<tr><td colspan="8">.
.
.</td><td>.
.
.</td><td>.
.
.</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>$f_{BTL} = f_{MEM} / 255$</td><td>254</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>$f_{BTL} = f_{MEM} / 256$</td><td>255</td></tr>
</table>
</td>
</tr>
</table>

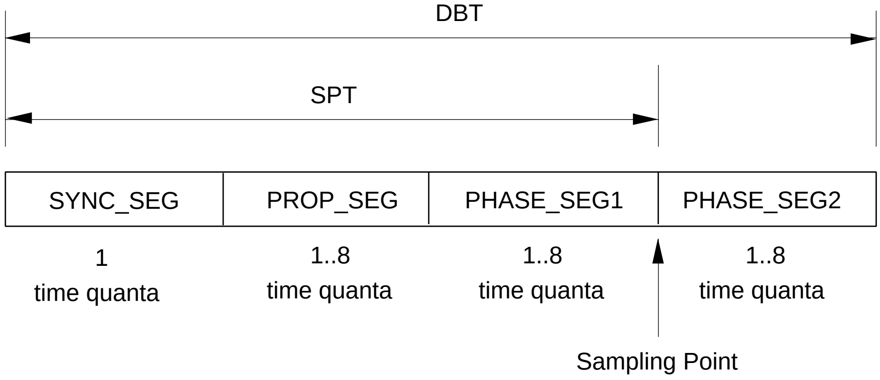
Remark: The BRP defines the period of the base clock f_{BTL} for the protocol layer of a CAN module. It determines the number of FCAN system clocks f_{MEM} per time quantum TQ. A time quantum TQ is the basic unit of a bit in a CAN frame:

$$TQ = \frac{1}{f_{BTL}}$$

(9) CAN 1 to 4 synchronization control registers (C1SYNC to C4SYNC)

A bit in a CAN frame is built by a programmable number of time quanta (TQ), as shown in the Figure 14-41 below.

Figure 14-41: CAN Bus Bit Timing



For the CAN modules in the FCAN system the bit length of segments SYNC_SEG, PROP_SEG, PHASE_SEG1 and PHASE_SEG2 must not be defined explicitly. All necessary CAN bit timings are programmed by defining

- the total number of time quanta TQ per CAN bit (i.e. DBT).
- the location of the sample point (i.e. SPT) as a number of TQ.

The CAN protocol segmentation is done by the CAN module automatically.

Due to re-synchronisation mechanisms the CAN module may lengthen PHASE_SEG1 or shorten PHASE_SEG2 by one or more TQ. The total number of TQ for which the CAN module is permitted to lengthen or shorten the phase segments is called synchronisation jump width (SJW). The SJW value must be less or equal the difference of DBT and SPT, which corresponds to PHASE_SEG2, and can be specified in the range of 1 TQ to 4 TQ.

For additional information on the CAN bus bit timing please refer to ISO 11898.

The relation between CAN memory clock and CAN bus baud rate is:

$$f_{CANBUS} = \frac{1}{DBT \times TQ} = \frac{f_{BTL}}{DBT} = \frac{f_{MEM}}{DBT \times BRP}$$

Valid values for DBT and BRP are:

TLM bit	DBT [TQ]	BRP[Note]	
0	8, 9, 10, ... ,25	2, 4, 6, ... ,128	2 × (k + 1)
1	8, 9, 10, ... ,25	1, 2, 3, ... ,256	k + 1

Note: BRP is the resulting bit rate prescaler value specified in the CxBRP register, where the variable k corresponds to the contents of bits BRP5 to BRP0 when TLM bit = 0, and bits BRP7 to BRP0 bits when TLM bit = 1, respectively (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

The CxSYNC registers specify the data bit time (DBT), sampling point position (SPT) and synchronisation jump width (SJW) of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).
These registers can be read/written in 8-bit and 16-bit units. However, write access is only permitted in initialisation mode (ISTAT bit of the CxCTRL register = 1)

Figure 14-42: CAN 1 to 4 Synchronization Control Registers (C1SYNC to C4SYNC) (1/2))

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1SYNC	0	0	0	SAMP	SJWR1	SJWR0	SPTR4	SPTR3	SPTR2	SPTR1	SPTR0	DBTR4	DBTR3	DBTR2	DBTR1	DBTR0	105EH	0218H
C2SYNC	0	0	0	SAMP	SJWR1	SJWR0	SPTR4	SPTR3	SPTR2	SPTR1	SPTR0	DBTR4	DBTR3	DBTR2	DBTR1	DBTR0	109EH	0218H
C3SYNC	0	0	0	SAMP	SJWR1	SJWR0	SPTR4	SPTR3	SPTR2	SPTR1	SPTR0	DBTR4	DBTR3	DBTR2	DBTR1	DBTR0	10DEH	0218H
C4SYNC	0	0	0	SAMP	SJWR1	SJWR0	SPTR4	SPTR3	SPTR2	SPTR1	SPTR0	DBTR4	DBTR3	DBTR2	DBTR1	DBTR0	111EH	0218H

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>12</td><td>SAMP</td><td>Specifies the bit sampling.
0: Sample receive data one time at sampling point.
1: Sample receive data three times and take majority decision at sampling point.</td></tr>
<tr><td>11, 10</td><td>SJWR1, SJWR0</td><td>Specifies the synchronization jump width.
<table>
<tr><th>SJWR1</th><th>SJWR0</th><th>Synchronization Jump Width</th></tr>
<tr><td>0</td><td>0</td><td>1 TQ</td></tr>
<tr><td>0</td><td>1</td><td>2 TQ</td></tr>
<tr><td>1</td><td>0</td><td>3 TQ</td></tr>
<tr><td>1</td><td>1</td><td>4 TQ</td></tr>
</table></td></tr>
<tr><td>9 to 5</td><td>SPTR4 to SPTR0</td><td>Specifies the sampling point position.
<table>
<tr><th>SPTR4</th><th>SPTR3</th><th>SPTR2</th><th>SPTR1</th><th>SPTR0</th><th>Sampling Point Position
SPTR = (p + 1) TQ</th><th>p</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="2">Setting prohibited</td><td rowspan="2"></td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>0</td><td>3 TQ</td><td>2</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>1</td><td>4 TQ</td><td>3</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>0</td><td>5 TQ</td><td>4</td></tr>
<tr><td colspan="5">.
.
.</td><td>.
.
.</td><td>.
.
.</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>17 TQ</td><td>16</td></tr>
<tr><td colspan="5">Other than above</td><td>Setting prohibited</td><td></td></tr>
</table></td></tr>
</table>

Note: The register address is calculated according to the following formula:
effective address = PP_BASE + address offset

Figure 14-42: CAN 1 to 4 Synchronization Control Registers (C1SYNC to C4SYNC) (2/2)

<table>
<tr><th>Bit Position</th><th>Bit Name</th><th>Function</th></tr>
<tr><td>4 to 0</td><td>DBTR4 to DBTR0</td><td>Specifies the number of TQ per bit.
<table>
<tr><th>DBTR4</th><th>DBTR3</th><th>DBTR2</th><th>DBTR1</th><th>DBTR0</th><th>Data Bit Time
DBTR = (q + 1) TQ</th><th>q</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td rowspan="3">Setting prohibited</td><td rowspan="3">–</td></tr>
<tr><td colspan="5">.
.
.</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>1</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>1</td><td>8 TQ</td><td>7</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>9 TQ</td><td>8</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>1</td><td>10 TQ</td><td>9</td></tr>
<tr><td colspan="5">.
.</td><td>.
.
.</td><td>.
.</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>0</td><td>25 TQ</td><td>24</td></tr>
<tr><td colspan="5">Other than above</td><td>Setting prohibited</td><td></td></tr>
</table>
</td></tr>
</table>

Remarks: 1. CPU can read the CxSYNC register at any time (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

2. Writing to the register is only possible when the CAN module is set to INIT mode.

3. For setting the DBTR and SPTR bits some rules must be observed, otherwise the CAN module will malfunction (for details refer to chapter 14.4 "Operating Considerations" on page 509).

(10) CAN 1 to 4 bus diagnostic information registers (C1DINF to C4DINF)

The CxDINF registers reflect the last transmission on CAN bus of the corresponding CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

These registers can be read-only in 1-bit, 8-bit and 16-bit units. It is only accessible when diagnostic mode is set (CxDEF register’s MOM bit = 1).

Figure 14-43: CAN 1 to 4 Bus Diagnostic Information Registers (C1DINF to C4DINF)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note]	Initial value
C1DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	105CH	0000H
C2DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	109CH	0000H
C3DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	10DCH	0000H
C4DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	111CH	0000H

Bit Position	Bit Name	Function
15 to 8	DINF15 to DINF8	Number of bits detected on the CAN bus since the last occurrence of a SOF signal.
7 to 0	DINF7 to DINF0	Reflects the value of the last 8 bits transmitted on the CAN bus, where DINF0 holds the very last bit.

Remarks: **1.** The CAN bus diagnostic information shows all CAN bus bits including stuff bits, delimiters, etc.

2. The storage of the last 8 bits is automatically stopped either when detecting an error on the CAN bus or when detecting a valid message (acknowledge delimiter). It is automatically reset whenever a SOF is detected on the CAN bus.

Caution: In normal operating mode the CxDINF register is hidden, and the corresponding CxBRP register appears instead of it at the same address.

14.4 Operating Considerations

14.4.1 Rules to be observed for correct baud rate settings

Observing the following rules for the baud rate setting assures correct operation of a CAN module and compliance to the CAN protocol specification.

(1) Rule for sampling point (SPT) setting:

The sample point position needs to be programmed between 3 TQ and 17 TQ, which corresponds to the SPTR4 to SPTR0 bits of the CxSYNC register (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)):

$$3\,\mathrm{TQ} \le SPT = (p+1)\,\mathrm{TQ} \le 17\,\mathrm{TQ}$$

$$2 \le p \le 16$$

$$p = \text{decimal value of bits SPTR4 to SPTR0}$$

(2) Rule for data bit time (DBT) setting:

The number of TQ per CAN frame bit is restricted to a range from 8 TQ to 25 TQ, which corresponds to the DBTR4 to DBTR0 bits of the CxSYNC register:

$$\mathrm{TQ} \le DBTR = (q+1)\,\mathrm{TQ} \le 25\,\mathrm{T}$$

$$7 \le q \le 24$$

$$q = \text{decimal value of bits DBTR4 to DBTR0}$$

(3) Rule for synchronization jump width (SJW) setting:

The number of TQ allowed for soft-synchronization must not exceed the number of TQ for PHASE_SEG2. The length of PHASE_SEG2 is given by the difference of data bit time (DBTR) and the sampling point position (SPTR). Converted to register values the following condition applies:

$$SJW = (s+1)\,TQ \le DBT - SPT$$

$$s \le q - p - 1$$

$$s = \text{decimal value of bits SJW1, SJW0}$$

Remark: The time quantum (TQ) is determined by the base clock f_{BTL} for the CAN protocol layer, which is defined in the CxBRP register:

$$TQ = \frac{1}{f_{BTL}}$$

Caution: The rules above represent CAN protocol limits. Violating these rules may cause erroneous operation.

14.4.2 Example for baudrate setting of CAN module

To illustrate how to calculate the correct setting of the registers CxBRP and CxSYNC the following example is given to 4:

Requirements from CAN bus:

- FCAN system global frequency f_{MEM} = 16 MHz
- CAN bus baud rate $f_{CANBUS} = (83^1/_3)$ kHz
- Sampling point 75% or above
- Synchronization jump width 3 TQ

First the frequency ratio between the global frequency and the CAN bus baud rate is calculated:

$$\frac{f_{MEM}}{f_{CANBUS}} = \frac{16\text{ MHz}}{(83^1/_3)\text{ KHz}} = 192 = 3 * 2^6$$

The register descriptions show that the prescaler must be an even number between 2 and 128, the data bit time must be a value in the range 8 to 25.
As the synchronization jump width (SJW) is defined as 3 TQ, the maximum setting for the sampling point (SPT) can be only 3 TQ less than the setting for the data bit time (DBT) and also less than 17 TQ:

$$SPT \leq min\ \{DBT - 1,\ 17\}$$

Based on the restrictions and assumptions above, the four settings are basically possible:

Prescaler (BRP)	Data Bit Time (DBTR)	Max. Sampling Point (SPTR)	Calculated Sampling Point
24	8 TQ	5 TQ	5/8 = 62.5%
16	12 TQ	9 TQ	9/12 = 75%
12	16 TQ	13 TQ	13/16 = 81.25%
8	24 TQ	17 TQ	17/24 = 71%

Regarding the maximum sampling point setting and the resulting sampling point, two settings meet all the requirements above. Therefore the correct settings are:

(1) TLM=0:

BRP5 to BRP0	=	000101B	(5)	(prescaler BRP = 12 TQ)
DBTR4 to DBTR0	=	01111B	(15)	(data bit time DBT = 16 TQ)
SPTR4 to SPTR0	=	01100B	(12)	(sampling point SPT = 13 TQ)

or

BRP5 to BRP0	=	000111B	(7)	(prescaler BRP = 16 TQ)
DBTR4 to DBTR0	=	01011B	(11)	(data bit time DBT = 12 TQ)
SPTR4 to SPTR0	=	01000B	(8)	(sampling point SPT = 9 TQ)

(2) TLM=1:

BRP7 to BRP0	=	00001011B	(11)	(prescaler BRP = 12)
DBTR4 to DBTR0	=	01111B	(15)	(data bit time DBT = 16 TQ)
SPTR4 to SPTR0	=	01100B	(12)	(sampling point SPT = 13 TQ)

or

BRP7 to BRP0	=	00001111B	(15)	(prescaler BRP = 16)
DBTR4 to DBTR0	=	01011B	(11)	(data bit time DBT = 12 TQ))
SPTR4 t oSPTR0	=	01000B	(8)	(sampling point SPT = 8 TQ)

14.4.3 Ensuring data consistency

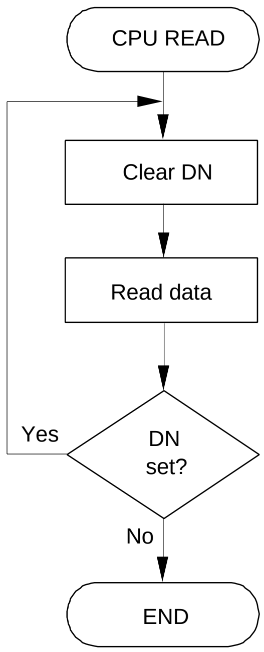
If the CPU reads data from the CAN message buffers, the consistency of data read has to be ensured. Therefore two mechanisms are provided:

- Sequential data read
- Burst mode data read

(1) Sequential data read

If the data is read by the CPU by sequential accesses to the CAN message buffers, the following sequence has to be observed:

Figure 14-44: Sequential CAN Data Read by CPU



As the DN flag is only set by the CAN module and cleared by the CPU only, it is ensured that the CPU can recognize that new data is stored in the message buffer during the read operation.

Remark: If the CPU reads the data by only one read access, the data consistency is always ensured. Therefore the check of the DN flag after reading the data is not necessary.

(2) Burst Mode Data Read

For faster access to a complete message the burst read mode is applicable.

In burst read mode the complete message is copied from the internal message buffer to a temporary read buffer located outside the CAN memory section. This allows read access without any wait, if the CAN memory is accessed by the CAN modules while the CPU tries to read data.

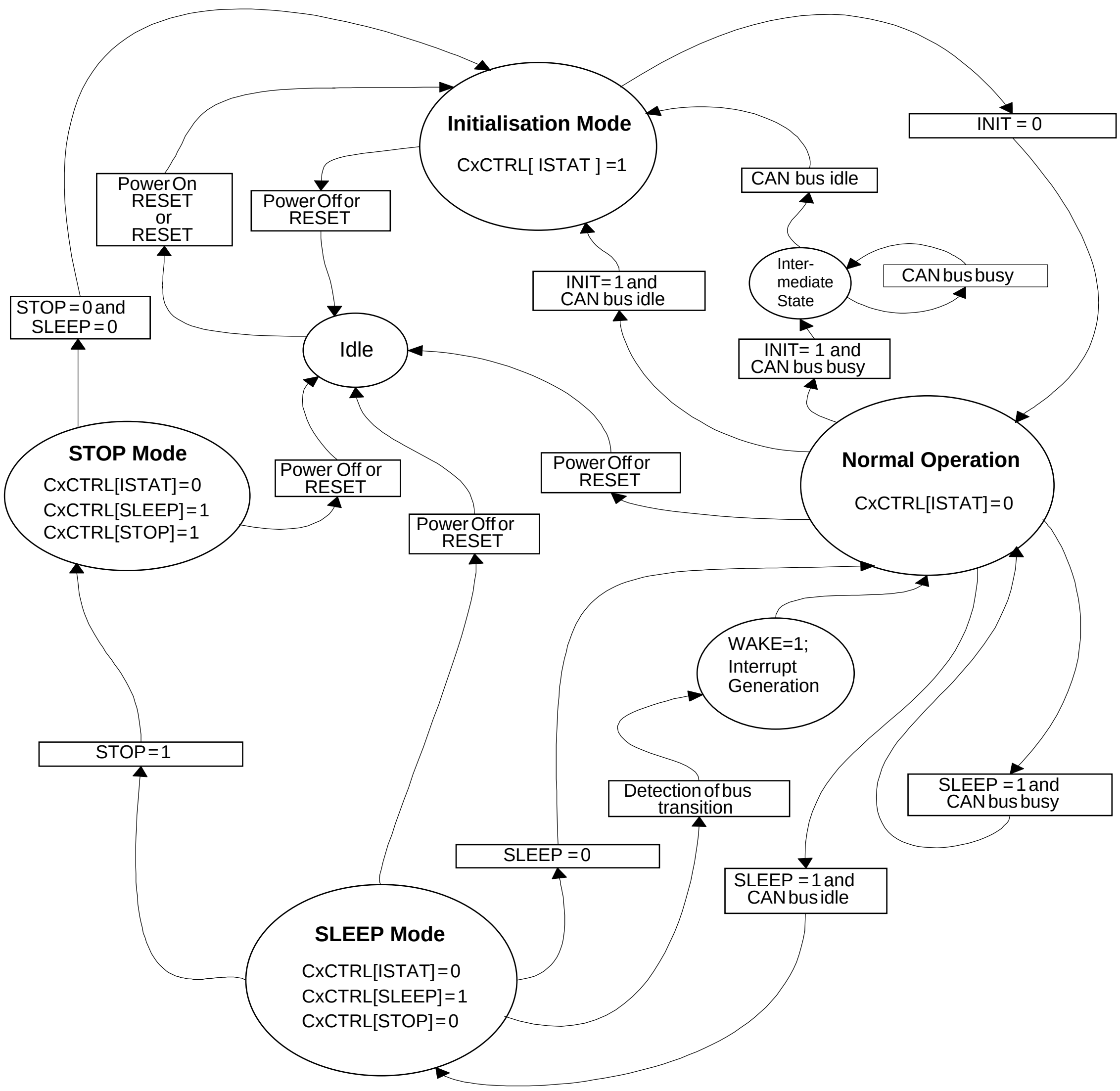
The copy of the message is automatically started whenever the data length code from the M_DLCm register is read by the CPU, and the data is copied from the message buffer into the temporary buffer. As long as the CPU reads 16-bit data from consecutive addresses (that means 16 -bit burst read sequence M_DLCm/M_CTRLm → M_TIMEm → M_DATAm0/m1 → M_DATAm2/m3 → M_DATAm4/m5 → M_DATAm6/m7 → M_IDLm → M_IDHm) the data is read from the temporary buffer.

Caution: The burst read requires consecutive 16-bit read accesses to the memory area. Any 8-bit access (byte read operation), even if not violating the linear address rule, causes that the read is performed from the register instead of the temporary buffer.

14.4.4 Operating states of the CAN modules

The different operating states and the state transitions of the CAN modules are shown in the state transition diagram in Figure 14-45.

Figure 14-45: State Transition Diagram for CAN Modules



Remark: x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A).

14.4.5 Initialisation routines

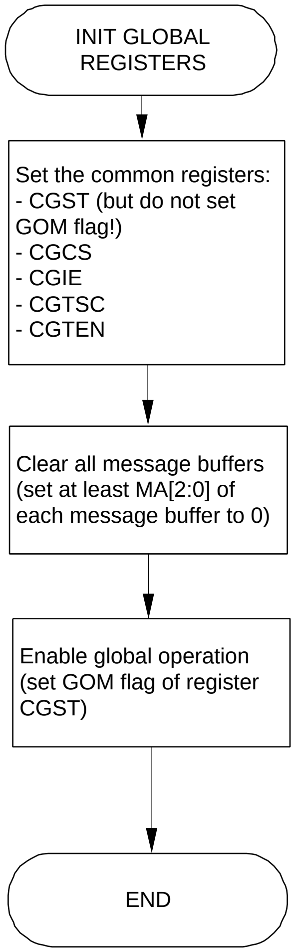
Below the necessary steps for correct start-up of the CAN interface are explained.

Caution: **It is very important that the software programmer observes the sequence given in the following paragraphs. Otherwise unexpected operation of the CAN interface or any CAN module can occur.**

(1) Global initialisation sequence for the CAN interface

Before any operation on the CAN memory can be done, it is essential that the common control register are initialised. The general initialisation sequence is shown in Figure 14-46.

Figure 14-46: General Initialisation Sequence for the CAN Interface



Remark: Enabling the global operation does not automatically enable any CAN module. Each CAN module must be initialised and enabled separately.

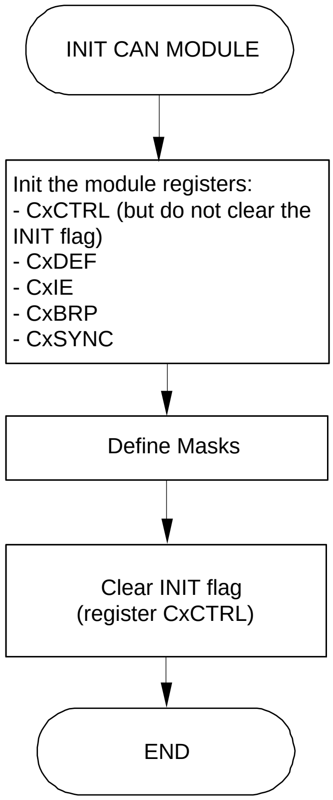
Example for C routine:

```
int CAN_GlobalInit (void)
{
    unsigned char i;

    // if GOM flag is already set
    if(CGST & 0x01)
    {
        // disable all CAN modules
        for(i=0; i <= CAN_MODULES; i++)
            CAN_ModuleStop(i);

        // clear GOM flag
        CGST = 0x0001;
    }

    CGST    = 0x00FF;       // clear all flags of CGST
    CGIE    = 0x00FF;       // disable global interrupts
    CGCS    = 0x0000;       // define internal clock
    CGTSC   = 0x0000;       // clear CAN global time system counter
    CGTEN   = 0x0000;       // disable all timer events

    // clear all message buffers
    for (i=0; i<CAN_MESSAGES; i++)
        CAN_ClearMessage(i);

    // set GOM bit
    CGST = 0x0100;

    return 0;
}
```

(2) Initialisation sequence for a CAN Module

Each CAN module must be initialised by the sequence according to Figure 14-47.

Figure 14-47: Initialisation Sequence for a CAN module



Remark: x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

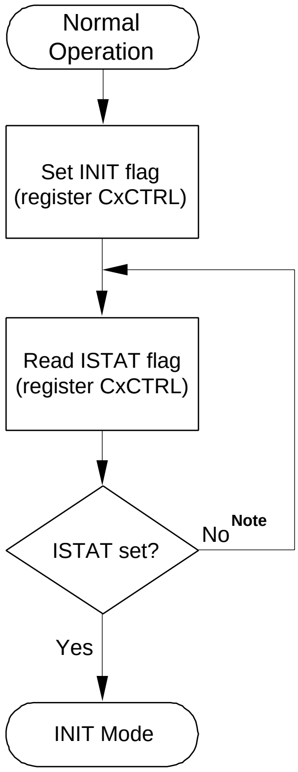
Example for C routine:

```
int CAN_ModuleInit (unsigned char   module_no,
                    unsigned short  brp_value,
                    unsigned short  sync_value)
{
    can_module_type *can_mod_ptr;                // define ptr
    can_mod_ptr = &can_module[module_no];        // load ptr

    can_mod_ptr->CxCTRL      = 0x00FE;           // clear CxCTRL
                                                 // except INIT
    can_mod_ptr->CxDEF       = 0x00FF;           // clear CxDEF
    can_mod_ptr->CxIE        = 0x00FF;           // clear CxIE
    can_mod_ptr->CxBRP       = brp_value;        // set CxBRP
    can_mod_ptr->CxSYNC      = sync_value;       // set CxSYNC
    can_mod_ptr->mask0_low   = 0x0000;           // clear mask0
    can_mod_ptr->mask0_high  = 0x0000;
    can_mod_ptr->mask1_low   = 0x0000;           // clear mask1
    can_mod_ptr->mask1_high  = 0x0000;
    can_mod_ptr->mask2_low   = 0x0000;           // clear mask2
    can_mod_ptr->mask2_high  = 0x0000;
    can_mod_ptr->mask3_low   = 0x0000;           // clear mask3
    can_mod_ptr->CxCTRL      = 0x0001;           // clear INIT flag

    return 0;
}
```

(3) Setting a CAN Module into initialisation state

The following routine is required if a CAN module has to be set from normal operation into initialisation mode.
Please notice that all CAN modules are automatically set to initialisation mode after reset. Therefore the sequence is only required if the CAN module is already in normal operation.

Figure 14-48: Setting CAN Module into Initialisation State



Note: In case of permanent bus activity the program loops for a long time. Therefore, a time-out mechanism should be provided in order to limit the runtime of the routine.

Remark: x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

Example for C routine:

```
int CAN_ModuleStop (unsigned char module_no)
{
    can_module_type *can_mod_ptr;                     // define CAN module ptr
    can_mod_ptr = &can_module[module_no];             // load CAN module ptr

    if ((can_mod_ptr->CxCTRL & 0x0001)==0)            // if INIT flag not yet set:
        can_mod_ptr->CxCTRL=0x0100;                   // set INIT flag

    while ((can_mod_ptr->CxCTRL & 0x0100)==0);        // wait until initialisation state is confirmed
                                                      // (ISTAT bit = 1)

    return 0;
}
```

(4) Shutdown of the FCAN system

If the clock to the CAN interface should be switched off for power saving, the following sequence has to be executed for correct termination of any CAN bus activity:

<1> For each CAN module x (x = 1 to 4 for the derivatives µPD703129(A) and µPD703129(A1), x = 1 to 2 for the derivative µPD703128(A)).

<a> Enter sleep mode
Set SLEEP bit = 1 (CxCTRL register)
or

<b> Enter initialisation mode
Set INIT bit = 1 (CxCTRL register) and wait for ISTAT bit = 1

<2> Disable event processing
Clear EVM flag (CGST register)

<3> Stop the CAN global time system counter
Clear the TSM flag (CGST register)

<4> Stop the global CAN operation
Clear GOM flag (CGST register)

<5> Switch off the CAN clock
Set CSTP bit (CSTOP register)

Caution: If the sequence is not observed, any active CAN module may cause malfunction on the corresponding CAN bus.

[MEMO]

Chapter 15 A/D Converter

15.1 Features

- 10-bit resolution on-chip A/D converter
- Analog inputs: 12 channels
- Standby function:
  - Current cut between AV_{DD} - AGND if A/D conversion is stopped
  - Current cut between AV_{REF} - AGND if A/D conversion is stopped
- A/D conversion trigger modes
  - A/D trigger mode
- Successive approximation technique.

15.2 Configuration

The A/D converter, which employs a successive approximation technique, performs A/D conversion operation using A/D converter mode register ADM, A/D converter register ADS and A/D conversion result registers ADCRL/ADCRH.

The A/D converter consists of the following hardware.

Table 15-1: A/D Converter Configuration

Item	Configuration
Analog input	12 channels (ANI0 to ANI11)
Register	Successive approximation register (SAR) A/D conversion result register (ADCR) A/D conversion result register (ADCRL): only lower 2-bits of A/D conversion result can be read A/D conversion result register (ADCRH): only higher 8-bits of A/D conversion result can be read
Control register	A/D converter mode register (ADM) Analog input channel setting register (ADS)

(1) Input circuit

The input circuit selects an analog input channel (ANIm) according to the mode set in the ADM register and sends it to the sample and hold circuit (m = 0 to 11).

(2) Sample and hold circuit

The sample and hold circuit samples the analog input sent from the input circuit and sends it to the comparator. It holds the sampled analog input during A/D conversion.

(3) Voltage comparator

The voltage comparator compares the analog input voltage from the input with the output voltage of the D/A converter.

(4) D/A converter

The D/A converter is used to generate a voltage that matches an analog input.
The output voltage of the D/A converter is controlled by the successive approximation register (SAR).

(5) Successive approximation register (SAR)

The SAR is a 10-bit register that controls the output value of the D/A converter for comparing with an analog input voltage value. When an A/D conversion terminates, the current contents of the SAR (conversion result) are stored in the A/D conversion result register (ADCR/ADCRL/ADCRH). When the specified A/D conversion terminates, there also is an A/D conversion termination interrupt (INTAD).

(6) A/D conversion result register (ADCR/ADCRL/ADCRH)

The ADCR register is an 16-bit register that holds all 10 bits of an A/D conversion result. ADCRL is an 8-bit register that holds the lower 2 bits of an A/D conversion result. ADCRH is an 8-bit register that holds the higher 8 bits of an A/D conversion result. Whenever an A/D conversion terminates, the conversion result from the successive approximation register (SAR) is loaded. $\overline{\text{RESET}}$ input sets these to 0000H.

(7) Controller

The controller selects an analog input, generates sample and hold circuit operation timing, controls the conversion trigger, specifies the conversion operation time.

(8) ANIm pins (m = 0 to 11)

The ANIm pins are the 12-channel analog input pins to analog converter.

(9) AV_{REF} pin

The AV_{REF} pin is used to input reference voltage to the A/D converter. A signal input to the ANIm pin is converted to a digital signal based on the voltage applied between AV_{REF} and AV_{SS} (m = 0 to 11). If not using the AV_{REF} pin, connect it to AV_{DD} or AV_{SS}**Note**.

(10) AV_{SS} pin

The AV_{SS} pin is the ground voltage pin of the A/D converter. Even if not using A/D converter, always ensure that this pin has the same DC potential as the V_{SS5} pin.

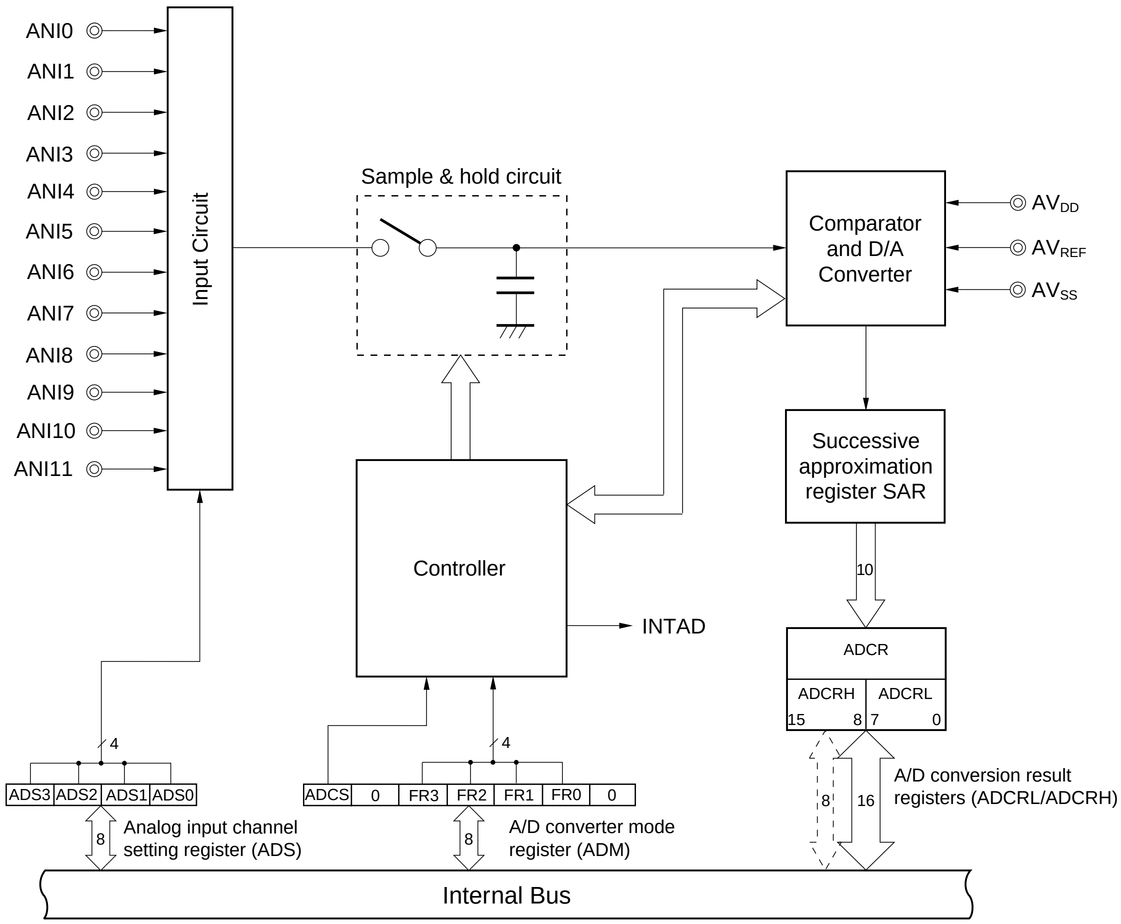
(11) AV_{DD} pin

The AV_{DD} pin is the analog power supply pin of A/D converter. Even if not using A/D converter, always ensure that this pin has the same potential as the V_{DD5} pin.

Note: When connecting the AV_{REF} pin to AV_{SS} the power consumption will be reduced.

Caution: Use input voltages to ANIm that are within the range of the ratings. In particular, if a voltage higher than AV_{DD} or lower than AV_{SS} (even one within the range of absolute maximum ratings) is input, the conversion value of that channel is undefined, and the conversion values of other channels also may be affected.

Figure 15-1: Block Diagram of A/D Converter



Cautions: 1. Noise at an analog input pin (ANIm) or reference voltage input pin (AV_{REF}) may give rise to an invalid conversion result.
Software processing is needed in order to prevent this invalid conversion result from adversely affecting the system.

2. The following are examples of software processing:

- **Use the average value of the results of multiple A/D conversions as the A/D conversion result.**
- **Perform A/D conversion multiple consecutive times and use conversion results with the exception of any abnormal conversion results that are obtained.**
- **If an A/D conversion result from which it is judged that an abnormality occurred in the system is obtained, do not perform abnormality processing at once but perform it upon reconfirming the occurrence of an abnormality.**

3. Be sure that voltages outside the range [AV_{SS} to AV_{REF}] are not applied to pins being used as A/D converter and input pins.

15.3 Control Registers

The following 2 types of registers are used to control A/D converter:

- A/D converter mode register (ADM)
- Analog input channel setting register (ADS)

15.3.1 Register format of A/D Converter Control Register

Table 15-2: Register format of A/D Converter Control Register

SFR name	Symbol	R/W	Manipulable Bit Unit			After Reset
			1 bit	8 bits	16 bits	
A/D converter mode register	ADM	R/W	×	×	---	00H
Analog input channel setting register	ADS	R/W	×	×	---	00H
A/D conversion result register	ADCR	R	---	---	×	undef.
A/D conversion result register	ADCRL	R	×	×	---	undef.
A/D conversion result register	ADCRH	R	×	×	---	undef.

(1) A/D converter mode register (ADM)

The ADM register is an 8-bit register that enables A/D conversion and the conversion time. It can be read or written in 1-bit or 8-bit units. However, writing to the ADM register during A/D conversion operation interrupts the conversion operation and the data is lost. The conversion operation restarts from the beginning. If the ADCS bit is cleared (0) during conversion operation, the conversion is aborted and the conversion operation is stopped.
While the ADCS bit is cleared (0), all DC current passes are cut between AV_{DD} and AV_{SS}, and between AV_{REF} and AV_{SS}.

Figure 15-2: A/D Converter Mode Register (ADM)

	7	6	5	4	3	2	1	0	Address	Initial value
ADM	ADCS	0	FR3	FR2	FR1	FR0	0	0	FFFFF200H	0000H

Bit Position	Bit Name	Function
7	ADCS	Specifies enabling or disabling A/D conversion. 0: Conversion operation stop 1: Conversion operation enable
5 to 2	FR3 to FR0	Specifies the conversion time (see table below) **Remark:** f_{PCLK} : Internal peripheral clock.

FR3	FR2	FR1	FR0	Conversion Clocks	Conversion Time [µs]	
					f_{PCLK} = 20 MHz	f_{PCLK} = 16 MHz
0	1	0	1	84	-	5.25
0	1	1	0	96	-	6.0
0	1	1	1	108	5.4	6.75
1	0	0	0	120	6.0	7.5
1	0	0	1	144	7.2	9.0
1	0	1	0	168	8.4	10.5
1	0	1	1	192	9.6	12
1	1	0	0	216	10.8	-
Others				Setting prohibited	-	-

Notes: 1. The bits FR0, FR1, FR2 and FR3 must not be changed while ADCS bit is set (1).

2. Conversion time (actual A/D conversion time)
Always set the time to 5 µs ≤ Conversion time ≤ 12 µs

Caution: Be sure not to change the setting of bits 0, 1 and 6 from their reset value (0). If these bits are set (1), the operation is not guaranteed.

(a) Conversion time setting

In order to prevent a drastic change of A/D conversion time even when the oscillation frequency is changed, the conversion speed of an A/D conversion can be adjusted. By the selection bits FR3 to FR0 in the ADM register the number of the conversion clocks can be set in the range of 84 to 216.

However, the settings modifying the conversion time (T_{CONV}) must keep the following relation :

$$5\ \mu s \leq T_{CONV} \leq 12\ \mu s$$

Example:

Provided that f_{PCLK} = 16 MHz

A setting of bits FR3 to FR0 = 0101B will be chosen. The conversion time is

$$T_{CONV} = Conversion\ Clocks\ (FR3\ to\ FR0)\ \frac{1}{f_{PCLK}} = 84\ \frac{1}{16\,MHz}$$

By this the conversion time results in

$$T_{CONV} = 5.25\ \mu s$$

(2) A/D converter register (ADS)

The ADS register is an 8-bit register that selects the analog input channel for the A/D conversion. It can be read or written in 1-bit or 8-bit units. Writing to the ADS register during A/D conversion operation interrupts the conversion operation and the data is lost. The conversion operation restarts from the beginning.

Figure 15-3: A/D Converter Register (ADS)

	7	6	5	4	3	2	1	0	Address	Initial value
ADS	0	0	0	0	ADS3	ADS2	ADS1	ADS0	FFFFF201H	0000H

Bit Position	Bit Name	Function
3 to 0	ADS3 to ADS0	The bits ADS3 to ADS0 specify the analog input channel for which the A/D conversion is performed. .

ADS3	ADS2	ADS1	ADS0	Analog input channel
0	0	0	0	AIN0
0	0	0	1	AIN1
0	0	1	0	AIN2
0	0	1	1	AIN3
0	1	0	0	AIN4
0	1	0	1	AIN5
0	1	1	0	AIN6
0	1	1	1	AIN7
1	0	0	0	AIN8
1	0	0	1	AIN9
1	0	1	0	AIN10
1	0	1	1	AIN11
Others than above				Setting prohibited

(3) A/D conversion result register (ADCR)

The ADCR registers is the A/D conversion result register that holds the result of the A/D conversion. When reading 10 bits of data of an A/D conversion result from the ADCR register, only the higher 10 bits are valid and the lower 6 bits are always read 0.

This register can be read in 16-bit units.

Figure 15-4: A/D Conversion Result Register (ADCR)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADCR	ADCR9	ADCR8	ADCR7	ADCR6	ADCR5	ADCR4	ADCR3	ADCR2	ADCR1	ADCR0	0	0	0	0	0	0	FFFFF202	undef.

Bit Position	Bit Name	Function
15 to 6	ADCR15 to ADCR6	The bits ADCR15 to ADCR6 hold the 10-bit result of the A/D conversion.

(4) A/D conversion result register L (ADCRL)

The ADCRL register is the A/D conversion result register that holds the lower 2-bit result of the A/D conversion. The ADCRL register is the same as the lower byte of the ADCR register. When reading all 8 bits of data of an A/D conversion result from the ADCRL register, only the higher 2 bits are valid and the lower 6 bits are always read 0.

This register can be read in 1-bit or 8-bit units.

Figure 15-5: A/D Conversion Result Register (ADCRL)

	7	6	5	4	3	2	1	0	Address	Initial value
ADCRL	ADCR1	ADCR0	0	0	0	0	0	0	FFFFF202H	undef.H

Bit Position	Bit Name	Function
7 to 6	ADCR1 to ADCR0	The bits ADCR1 to ADCR0 hold the lower 2-bits result of the A/D conversion.

(5) A/D conversion result register H (ADCRH)

The ADCRH register is the A/D conversion result register that holds the upper 8-bit result of the A/D conversion. The ADCRH register is the same as the higher byte of the ADCR register.

This register can be read in 1-bit or 8-bit units.

Figure 15-6: A/D Conversion Result Register (ADCRH)

	7	6	5	4	3	2	1	0	Address	Initial value
ADCRH	ADCR9	ADCR8	ADCR7	ADCR6	ADCR5	ADCR4	ADCR3	ADCR2	FFFFF203H	undef.H

Bit Position	Bit Name	Function
7 to 0	ADCR9 to ADCR2	The bits ADCR9 to ADCR2 hold the upper 8-bits result of the A/D conversion.

(6) Port function register 7/8 (PORT7/PORT8)

The PORT7/PORT8 register holds the digital input values of the A/D input channels ANI0 to ANI11 (P70 to P77, P80 to P83).

This register can only be read in 16-bit units.

Figure 15-7: Port Function Register (PORT7/PORT8)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
Port7/ Port8	0	0	0	0	P83/ ANI11	P82/ ANI10	P81/ ANI9	P80/ ANI8	P77/ ANI7	P76/ ANI6	P75/ ANI5	P74/ ANI4	P73/ ANI3	P72/ ANI2	P71/ ANI1	P70/ ANI0	FFFFF40CH	undef.

Bit Position	Bit Name	Function
11 to 8	P83 to P80	The bits P83 to P80 holds the digital input values of the A/D input channels ANI11 to ANI8.
7 to 0	P77 to P70	The bits P77 to P70 holds the digital input values of the A/D input channels ANI7 to ANI0.

Note: Reading the digital value of the analog input channel ANIx is disabled during A/D conversion operation.

Remark: x = 0 to 11

(7) Port function register 7 (PORT7)

The PORT7 register holds the digital input values of the A/D input channels ANI0 to ANI7 (P70 to P77).
This register can be read in 1-bit and 8-bit units.

Figure 15-8: Port Function Register 7 (PORT7)

	7	6	5	4	3	2	1	0	Address	Initial value
Port7	P77/ ANI7	P76/ ANI6	P75/ ANI5	P74/ ANI4	P73/ ANI3	P72/ ANI2	P71/ ANI1	P70/ ANI0	FFFFF40CH	undef.H

Bit Position	Bit Name	Function
7 to 0	P77 to P70	The bits P77 to P70 holds the digital input values of the A/D input channels ANI7 to ANI0.

Note: Reading the digital value of the analog input channel ANIx is disabled during A/D conversion operation.

Remark: x = 0 to 11

15.3.2 Input voltage and conversion results

The relation between the analog input voltage input to the analog input pins (ANI0 to ANI11) and the A/D conversion result (stored in the A/D conversion result registers (ADCR)) is shown by the following expression:

$$ADCR = INT\left[\left(\frac{(AV_{IN} \times 1024)}{AV_{REF}} \pm 0{,}5\right)\right] \times 64$$

or the following expression:

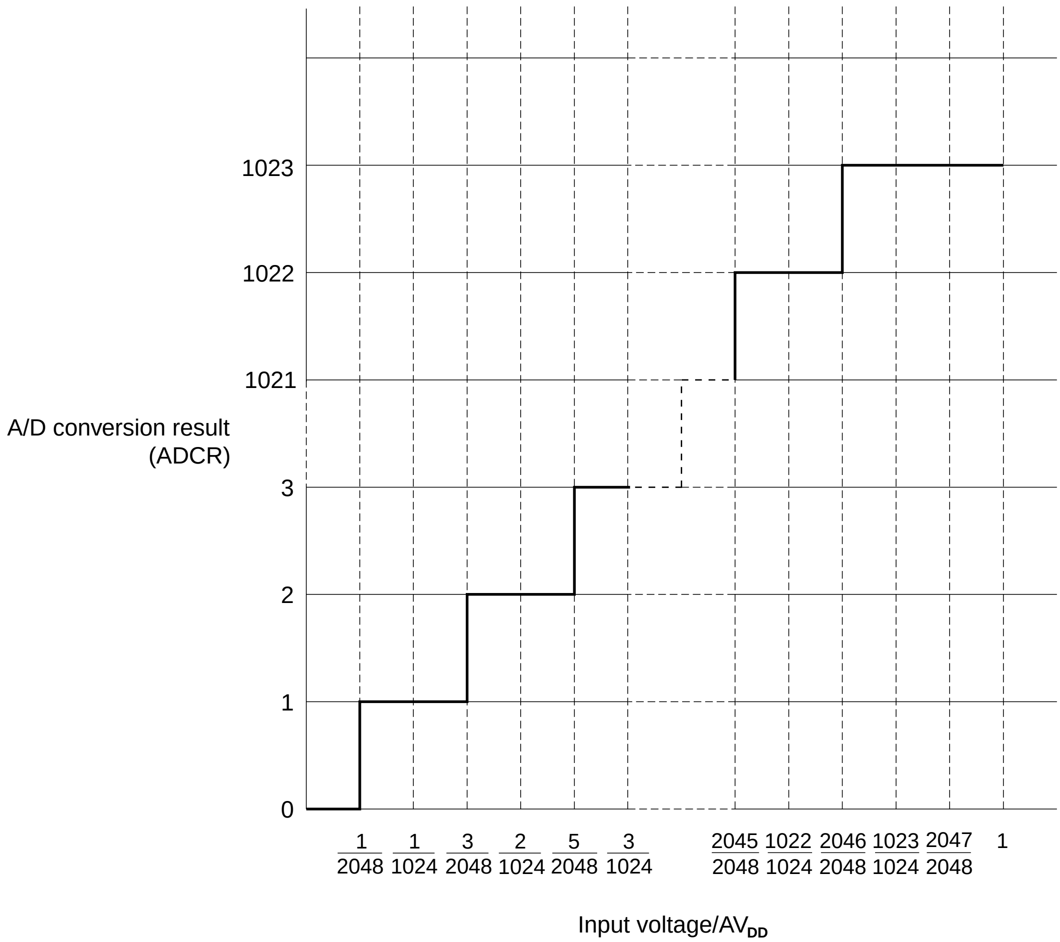
$$\frac{(ADCR/64 - 0.5) \times AV_{REF}}{1024} <= V_{IN} < \frac{(ADCR/64 + 0.5) \times AV_{REF}}{1024}$$

where,

INT( ): Function which returns integer part of value in parentheses
V_{IN}: Analog input voltage
AV_{DD}: AV_{DD} pin voltage and A/D converter power supply
ADCRL: A/D conversion result register (ADCRL) value

Figure 15-9, “Relation between Analog Input Voltage and A/D Conversion Result,” on page 536 shows the relation between the analog input voltage and the A/D conversion result.

Figure 15-9: Relation between Analog Input Voltage and A/D Conversion Result



15.4 Interrupt Request

The A/D converter generates a dedicated interrupt.

- A/D conversion termination interrupt (INTAD)

(1) A/D conversion termination interrupt (INTAD)

In A/D conversion enabled status, an A/D conversion termination interrupt is generated when the specified input channel A/D conversion has terminated.

15.5 A/D Converter Operation

15.5.1 A/D converter basic operation

A/D conversion is performed using the following procedure.

(1) Set the analog input selection using the ADS**Note 1** register. Setting (1) the ADCS**Note 2** bit of the ADM**Note 1** register starts the A/D conversion for the selected A/D input channel.

(2) When the A/D conversion starts, the selected analog input is compared with the voltage generated by the D/A converter.

(3) When the 10-bit comparison is completed, the conversion result is stored in the ADCR, ADCRL and ADCRH registers and a new conversion operation is started.

(4) An interrupt request signal (INTAD) is generated after completion of each conversion.

Notes: 1. If a write operation is carried out to the ADM or the ADS register during conversion operation, the conversion operation is aborted and restarts from the beginning.

2. If the ADCS bit of the ADM register is cleared (0) during a A/D conversion operation, the conversion operation is aborted and conversion operation is stopped.

15.5.2 Operation modes

The operation mode of the A/D converter is the soft-trigger mode. One analog channel is selected from among ANI0 to ANI11 with the analog input channel setting register (ADS).

(a) Soft-trigger mode

In the soft-trigger mode the A/D converter converts one analog input specified in the ADS register. The conversion result is stored in the ADCR, ADCRL, ADCRH register.

Figure 15-10: No write operation is made to ADM or ADS register during A/D conversion operation

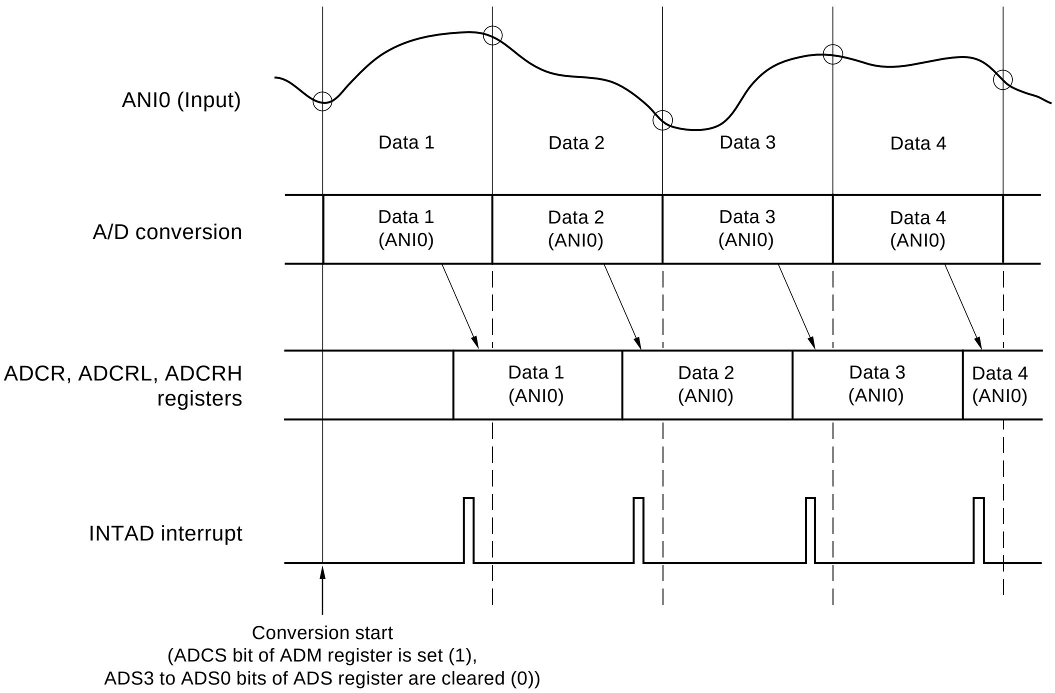


Figure 15-11: ADCS bit is cleared (0) during A/D conversion operation

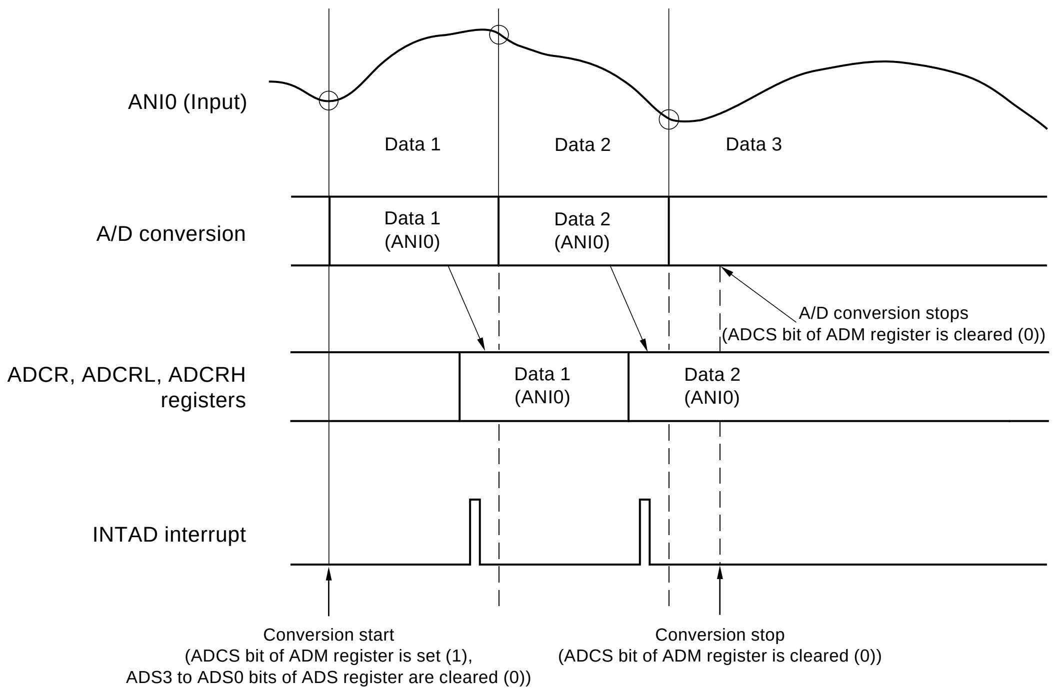
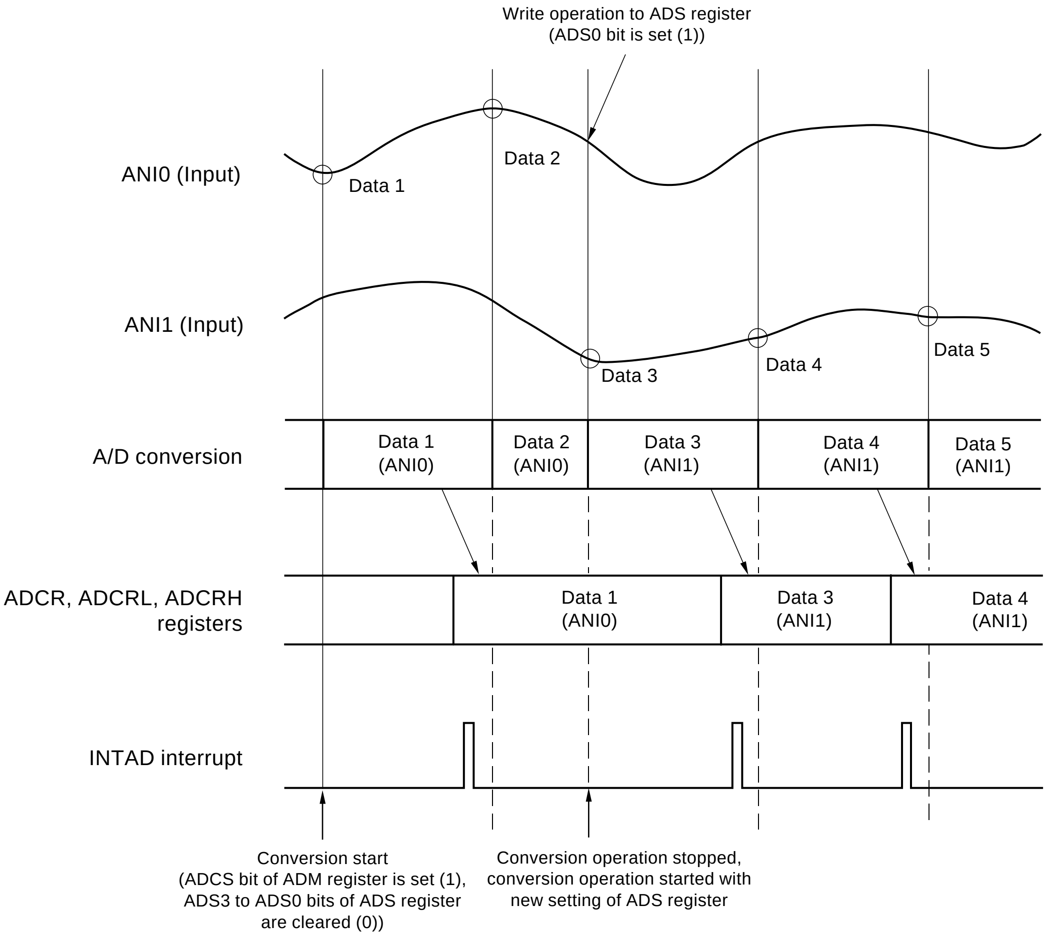


Figure 15-12: A write operation is made to the ADS register during A/D conversion operation



15.6 A/D Converter Precautions

(1) Current consumption in standby mode

A/D converter current consumption can be reduced by stopping the A/D conversion operation. A/D conversion operation is stopped by resetting the ADCS bit of the A/D converter mode register ADM to (0).

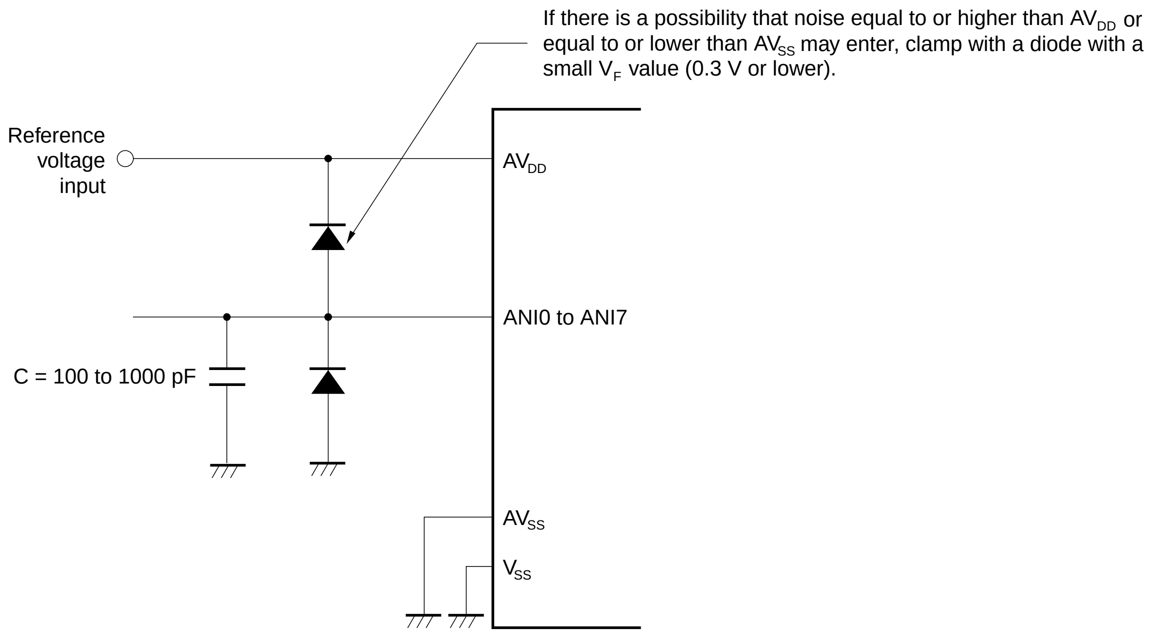
(2) Input range of ANI0 to ANI11

Keep the input voltage of the ANI0 through ANI11 pins to within the rated range. If a input voltage greater than AV_{DD} or lower than AV_{SS} (even within the range of the absolute maximum ratings) is input to a channel, the converted value of the channel becomes undefined. Moreover, the values of the other channels may also be affected.

(3) Noise counter measures

To maintain 10-bit resolution, attention must be paid to noise input to pin AV_{DD} and pins ANI0 to ANI11. Because the effect increases in proportion to the output impedance of the analog input source, it is recommended that a capacitor be connected externally as shown in Figure 15-13, “Analog Input Pin Handling,” on page 541 to reduce noise.

Figure 15-13: Analog Input Pin Handling



(4) ANI0 to ANI11

The analog input pins (ANI0 to ANI11) also function as input port pins (P80 to P83, P70 to P77). When A/D conversion is performed with any of pins ANI0 to ANI11 selected, do not execute a port input instruction while conversion is in progress, as this may reduce the conversion resolution. Also, if digital pulses are applied to a pin adjacent to the pin in the process of A/D conversion, the expected A/D conversion value may not be obtainable due to coupling noise. Therefore, avoid applying pulses to pins adjacent to the pin undergoing A/D conversion.

15.7 How to read the A/D Converter Characteristics Table

Here, special terms unique to the A/D converter are explained.

(1) Resolution

This is the minimum analog input voltage that can be identified. That is, the percentage of the analog input voltage per bit of digital output is called 1LSB (Least Significant Bit). The percentage of 1LSB with respect to the full scale is expressed by %FSR (Full Scale Range).

When the resolution is 10 bits,

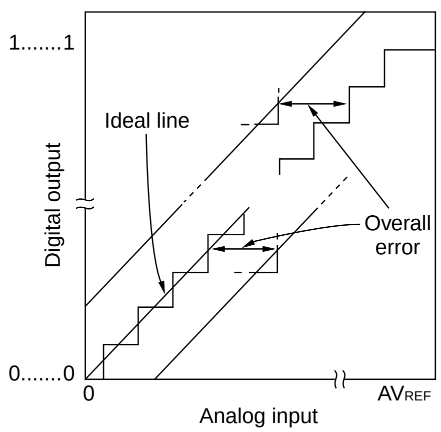
$1LSB = 1/2^{10} = 1/1024 = 0.098\%FSR$

Accuracy has no relation to resolution, but is determined by the overall error.

(2) Overall Error

This shows the maximum error value between the actual measured value and the theoretical value. Zero scale error, full scale error, non-linearity error and errors which are combinations of these express the overall error.
Note that the quantization error is not included in the overall error in the characteristics table.

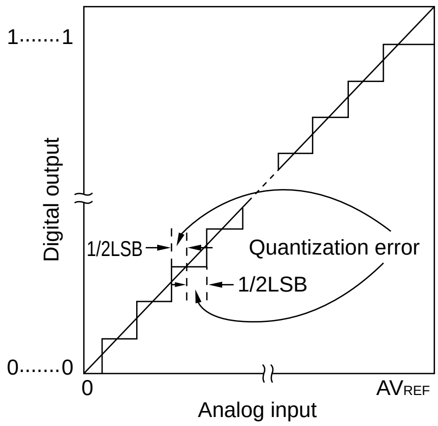
Figure 15-14: Overall Error



(3) Quantization Error

When analog values are converted to digital values, a ±1/2 LSB error naturally occurs. In an A/D converter, an analog input voltage in a range of ±1/2 LSB is converted to the same digital code, so a quantization error cannot be avoided.
Note that the quantization error is not included in the overall error, zero scale error, full scale error and non-linearity error in the characteristics table.

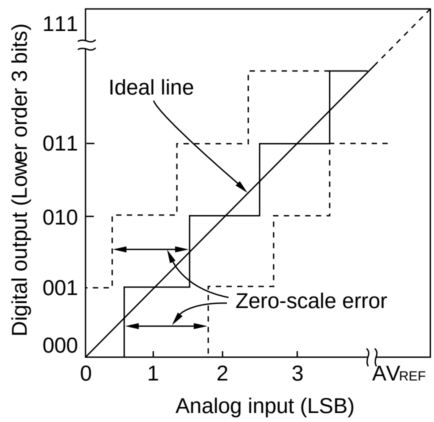
Figure 15-15: Quantization Error



(4) Zero-scale error

This shows the difference between the actual measurement value of the analog input voltage and the theoretical value (1/2 LSB) when the digital output changes from 0...000 to 0...001. If the actual measurement value is greater than the theoretical value, it shows the difference between the actual measurement value of the analog input voltage and the theoretical value (3/2 LSB) when the digital output changes from 0...000 to 0...010.

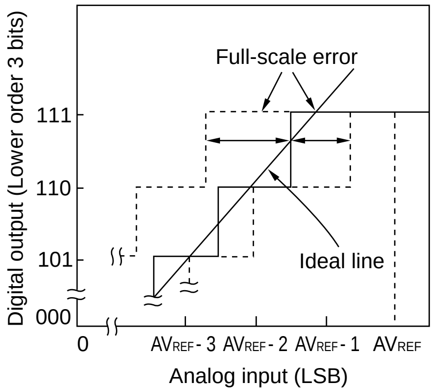
Figure 15-16: Zero-Scale Error



(5) Full-scale Error

This shows the difference between the actual measurement value of the analog input voltage and the theoretical value (3/2 LSB) when the digital output changes from 1...110 to 1...111.

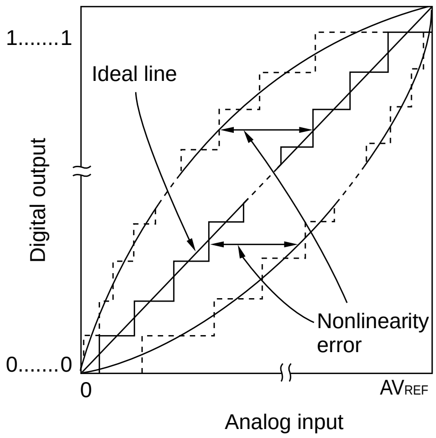
Figure 15-17: Full-Scale Error



(6) Nonlinearity Error

This shows the degree to which the conversion characteristics deviate from the ideal linear relationship. It expresses the maximum value of the difference between the actual measurement value and the ideal straight line when the zero scale error and full scale error are 0.

Figure 15-18: Nonlinearity Error



Chapter 16 Port Functions

16.1 Features

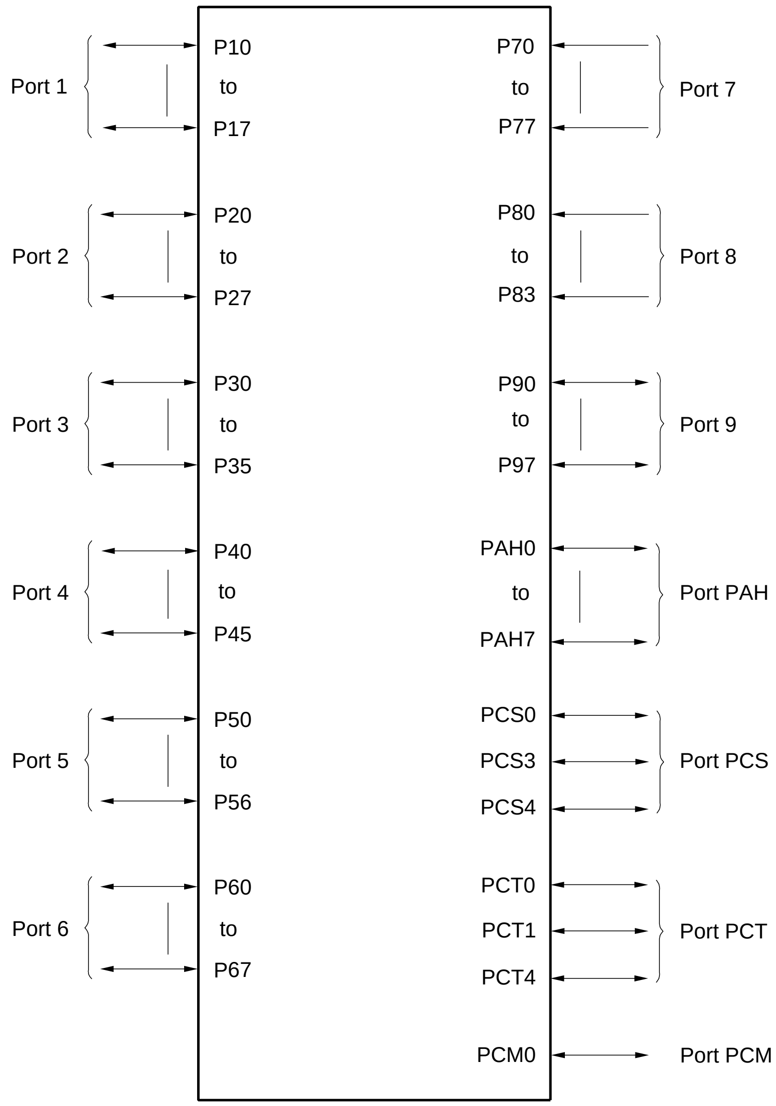
- Input/Output ports (5 V): 51
- Input ports (5 V): 12
- Input/Output ports (3.3 V): 15
- Ports alternate as input/output pins of other peripheral functions
- Input or output can be specified in bit units

16.2 Port Configuration

The V850E/CA2 incorporates a total of 78 input/output ports (12 ports are input only, 15 input/output ports have 3.3 V power). The ports are named ports P1 through P9, and PAH, PCM, PCT and PCS.

The configuration is shown below.

Figure 16-1: Port Configuration



(1) Functions of each port

The V850E/CA2 has the ports shown below.
Any port can operate in 8- or 1-bit units and can provide a variety of controls.
Moreover, besides its function as a port, some have functions as the input/output pins of on-chip peripheral I/O in control mode.
Refer to “(3) Port block diagram” for a block diagram of the block type of each port.

Table 16-1: Functions of each port

Port Name	Pin Name	Port Function	Function In Control Mode	Block Type
Port 1	P10 to P17	8-bit input/output	Serial interface input/output (FCAN1 to FCAN3**Note**, UART1)	A
Port 2	P20 to P27	8-bit input/output	Serial interface input/output (CSI0, CSI1, UART0)	A
Port 3	P30 to P35	6-bit input/output	Real-time pulse unit (RPU) input/output External interrupt input	A
Port 4	P40 to P45	6-bit input/output	Real-time pulse unit (RPU) input/output External interrupt input	A
Port 5	P50 to P56	7-bit input/output	Real-time pulse unit (RPU) input/output Serial interface input/output (FCAN4**Note**) External interrupt input	A
Port 6	P60 to P67	8-bit input/output	Serial interface input/output (CSI2) External interrupt input	A
Port 7	P70 to P77	8-bit input (Digital input of ANI0 to ANI7)	-	-
Port 7/8	P70 to P77, P80 to P83	12-bit input (Digital input of ANI0 to ANI11)	-	-
Port 9	P90 to P97	8-bit input/output	-	A
Port AH	PAH0 to PAH7	8-bit input/output	External address bus (A16 to A23)	B
Port CS	PCS0, PCS3, PCS4	3-bit input/output	External bus interface control signal output	C
Port CT	PCT0, PCT1, PCT4	3-bit input/output	External bus interface control signal output	D
Port CM	PCM0	1-bit input/output	External bus interface control signal input	E

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

(2) Functions of each port pin on reset and registers that set port or control mode

Table 16-2: Port Pin Functions (1/3)

Port Name	Pin Name	Pin Function after Reset	Mode-Setting Register
Port 1	P10/CRXD1	P10 (Input mode)	PMC1
	P11/CTXD1	P11 (Input mode)	
	P12/CRXD2	P12 (Input mode)	
	P13/CTXD2	P13 (Input mode)	
	P14/CRXD3[Note]	P14 (Input mode)	
	P15/CTXD3[Note]	P15 (Input mode)	
	P16/RXD1	P16 (Input mode)	
	P17/TXD1	P17 (Input mode)	
Port 2	P20/SI0	P20 (Input mode)	PMC2
	P21/SO0	P21 (Input mode)	
	P22/$\overline{\text{SCKO0}}$/$\overline{\text{SCKI0}}$	P22 (Input mode)	
	P23/SI1	P23 (Input mode)	
	P24/SO1	P24 (Input mode)	
	P25/$\overline{\text{SCKO1}}$/$\overline{\text{SCKI1}}$	P25 (Input mode)	
	P26/RXD0	P26 (Input mode)	
	P27/TXD0	P27 (Input mode)	
Port 3	P30/TIG00/INTP00	P30 (Input mode)	PMC3
	P31/TOG01/TIG01	P31 (Input mode)	
	P32/TOG02/TIG02	P32 (Input mode)	
	P33/TOG03/TIG03	P33 (Input mode)	
	P34/TOG04/TIG04	P34 (Input mode)	
	P35/TIG05/INTP05	P35 (Input mode)	
Port 4	P40/TIG10/INTP10	P40 (Input mode)	PMC4
	P41/TOG11/TIG11	P41 (Input mode)	
	P42/TOG12/TIG12	P42 (Input mode)	
	P43/TOG13/TIG13	P43 (Input mode)	
	P44/TOG14/TIG14	P44 (Input mode)	
	P45/TIG15/INTP15	P45 (Input mode)	
Port 5	P50/CRXD4[Note]	P50 (Input mode)	PMC5
	P51/CTXD4[Note]	P51 (Input mode)	
	P52/INTP4	P52 (Input mode)	
	P53/INTP5	P53 (Input mode)	
	P54/TI0/INTP20	P54 (Input mode)	
	P55/TI1/INTP21	P55 (Input mode)	
	P56/TO0	P55 (Input mode)	

Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Table 16-2: Port Pin Functions (2/3)

Port Name	Pin Name	Pin Function after Reset	Mode-Setting Register
Port 6	P60/NMI	P60 (Input mode)	PMC6
	P61/IINTP0	P61 (Input mode)	
	P62/INTP1	P62 (Input mode)	
	P63/INTP2	P63 (Input mode)	
	P64/INTP3	P64 (Input mode)	
	P65/SI2	P65 (Input mode)	
	P66/SO2	P66 (Input mode)	
	P67/$\overline{\text{SCKO2}}$/$\overline{\text{SCKI2}}$	P65 (Input mode)	
Port 7	P70/ ANI0	P70 (Input)/ ANI0	-
	P71/ ANI1	P71 (Input)/ ANI1	
	P72/ ANI2	P72 (Input)/ ANI2	
	P73/ ANI3	P73 (Input)/ ANI3	
	P74/ ANI4	P74 (Input)/ ANI4	
	P75/ ANI5	P75 (Input)/ ANI5	
	P76/ ANI6	P76 (Input)/ ANI6	
	P77/ ANI7	P77 (Input)/ ANI7	
Port 8	P80/ ANI8	P80 (Input)/ ANI8	-
	P81/ ANI9	P81 (Input)/ ANI9	
	P82/ ANI10	P82 (Input)/ ANI10	
	P83/ ANI11	P83 (Input)/ ANI11	
Port 9	P90	P90 (Input mode)	-
	P91	P91 (Input mode)	
	P92	P92 (Input mode)	
	P93	P93 (Input mode)	
	P94	P94 (Input mode)	
	P95	P95 (Input mode)	
	P96	P96 (Input mode)	
	P97	P97 (Input mode)	
Port AH	PAH0/A16	A16 (Address output mode)	PMCAH
	PAH1/A17	A17 (Address output mode)	
	PAH2/A18	A18 (Address output mode)	
	PAH3/A19	A19 (Address output mode)	
	PAH4/A20	A20 (Address output mode)	
	PAH5/A21	A21 (Address output mode)	
	PAH6/A22	A22 (Address output mode)	
	PAH7/A23	A23 (Address output mode)	
Port CS	PCS0/$\overline{\text{CS0}}$	$\overline{\text{CS0}}$ (Chip select output mode)	PMCCS
	PCS3/$\overline{\text{CS3}}$	PCS3 (Input mode)	
	PCS4/$\overline{\text{CS4}}$	PCS4 (Input mode)	

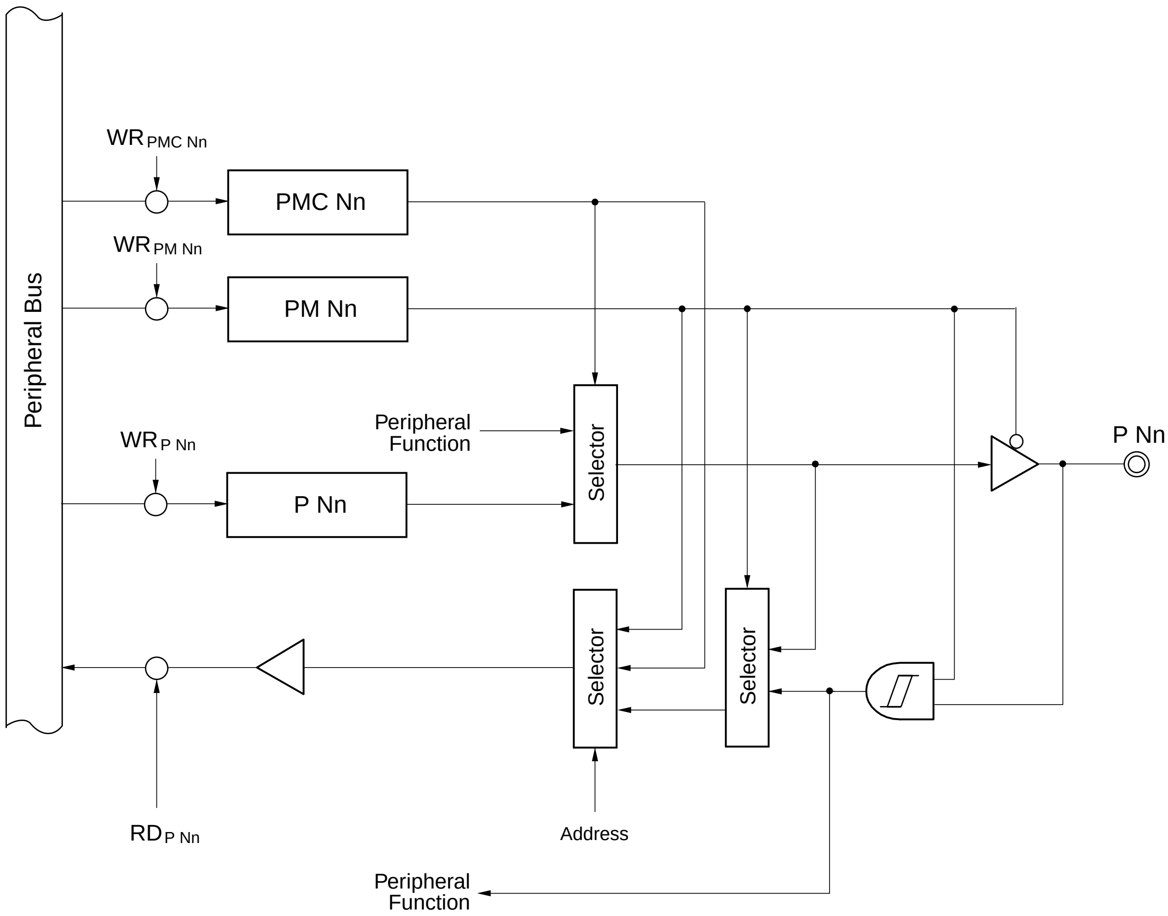
Note: CAN module 3 and CAN module 4 are available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

Table 16-2: Port Pin Functions (3/3)

Port Name	Pin Name	Pin Function after Reset	Mode-Setting Register
Port CT	PCT0/$\overline{\text{LWR}}$	PCT0 (Input mode)	PMCCT
	PCT1/$\overline{\text{UWR}}$	PCT1 (Input mode)	
	PCT4/$\overline{\text{RD}}$	$\overline{\text{RD}}$ (Read strobe signal output mode)	
Port CM	PCM0/$\overline{\text{WAIT}}$	$\overline{\text{WAIT}}$ (Wait insertion signal input mode)	PMCCM

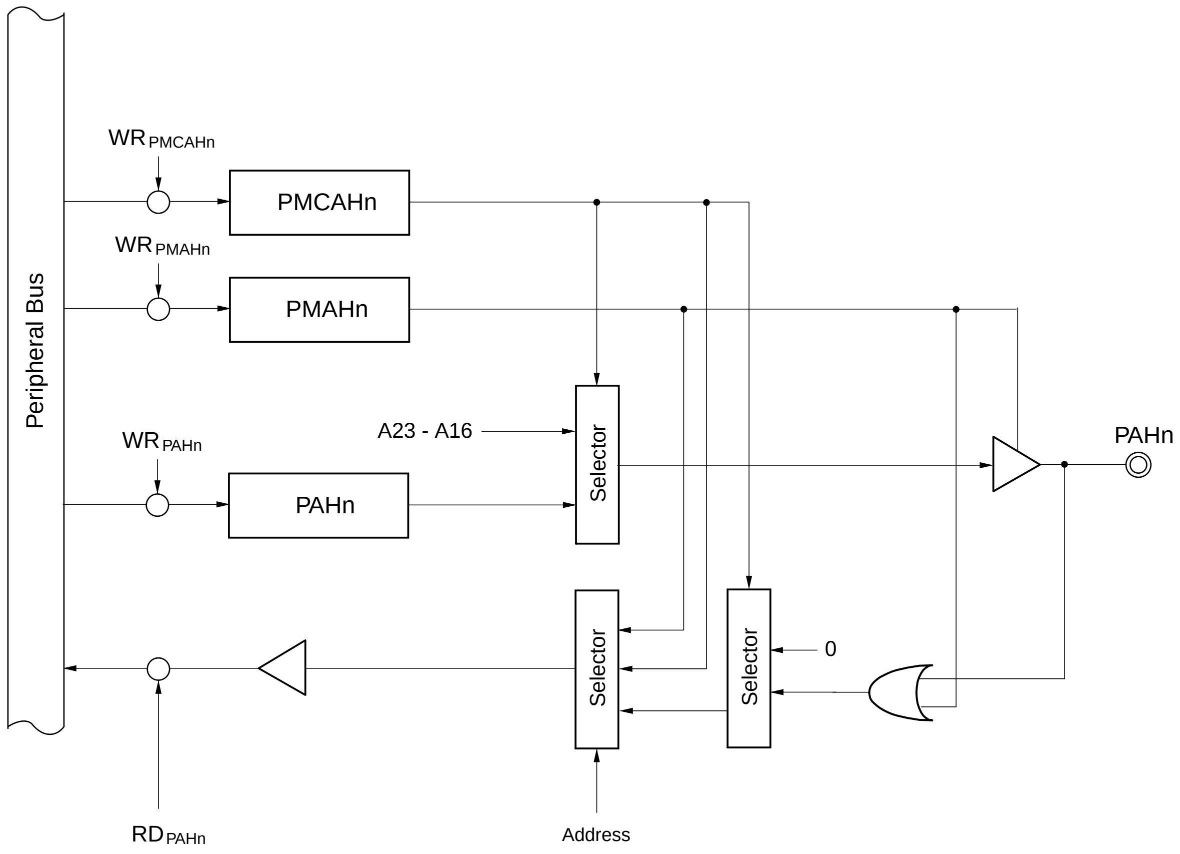
(3) Port block diagram

Figure 16-2: Type A Block Diagram



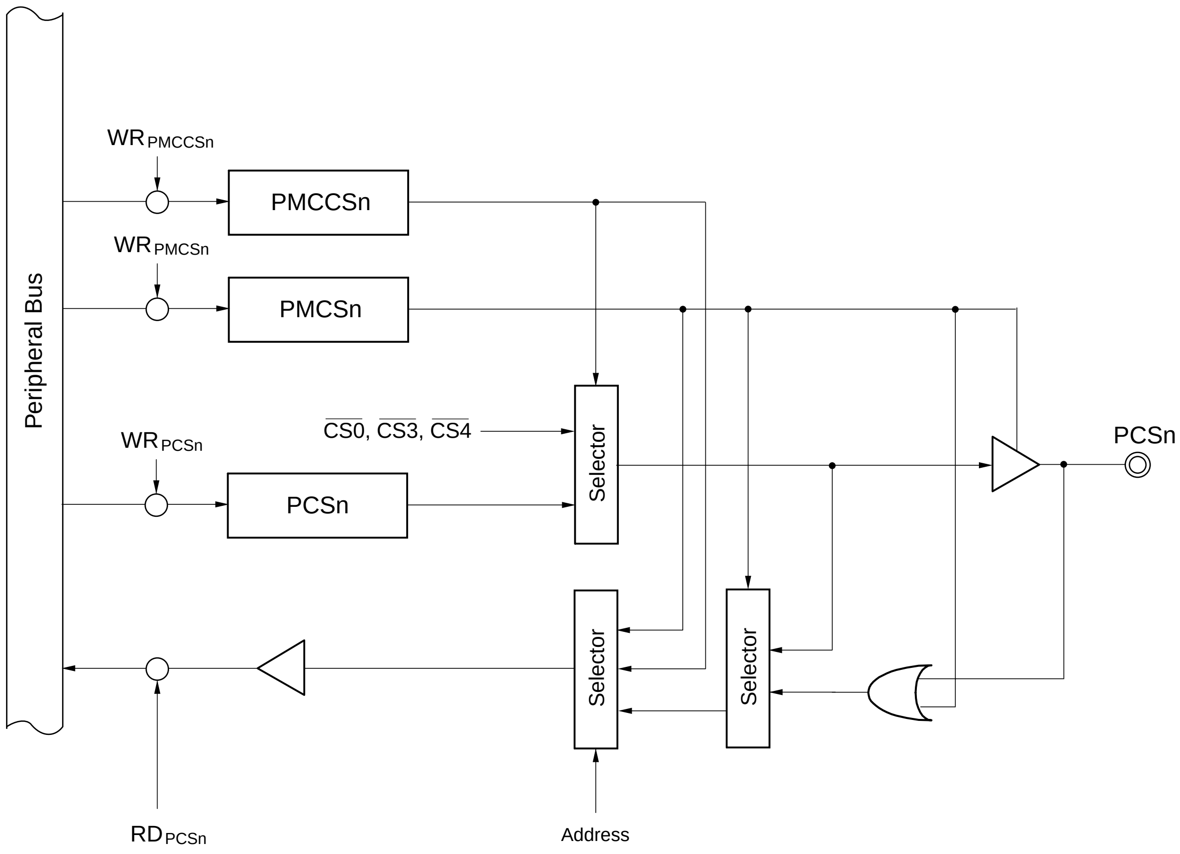
Remark: N = 1 to 6, 9: Port number
n = 0 to 7: Port pin for port number 1, 2, 6, 9
n = 0 to 6: Port pin for port number 5
n = 0 to 3: Port pin for port number 3, 4

Figure 16-3: Type B Block Diagram



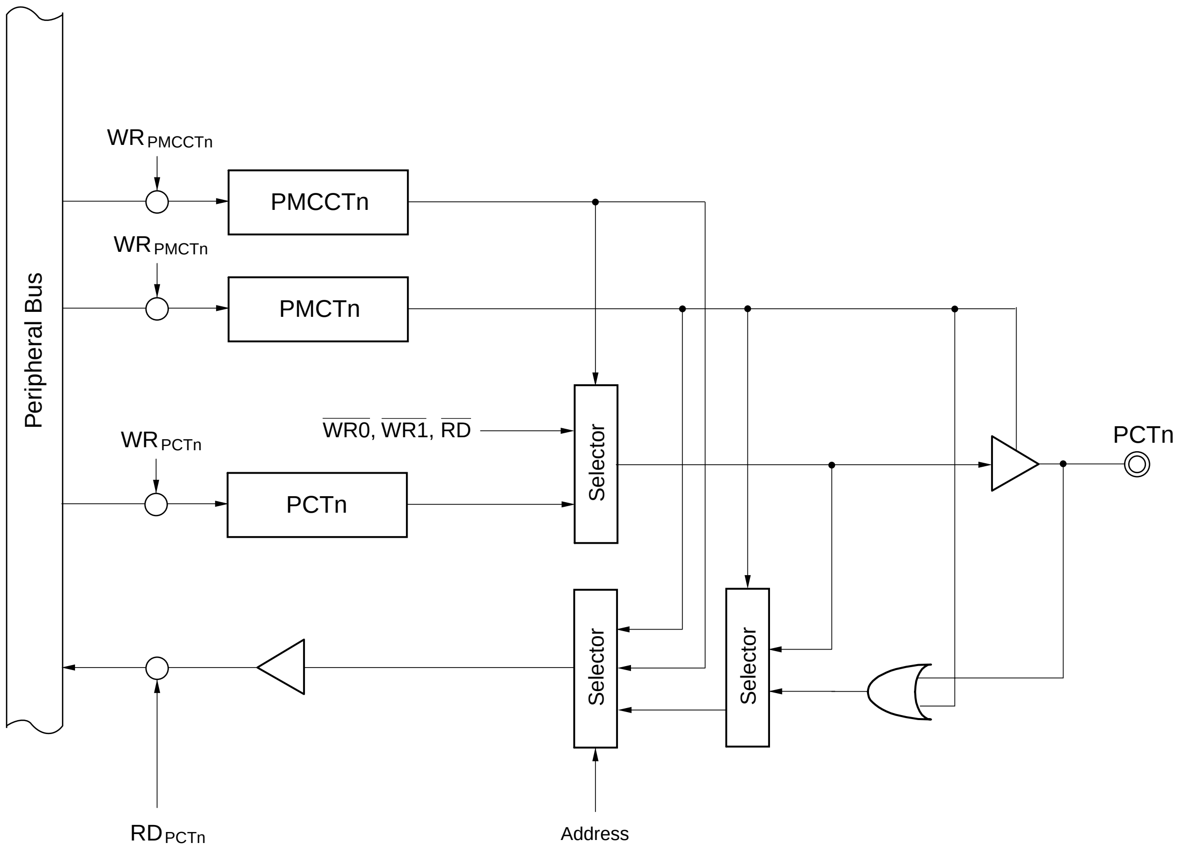
Remark: n = 0 to 7

Figure 16-4: Type C Block Diagram



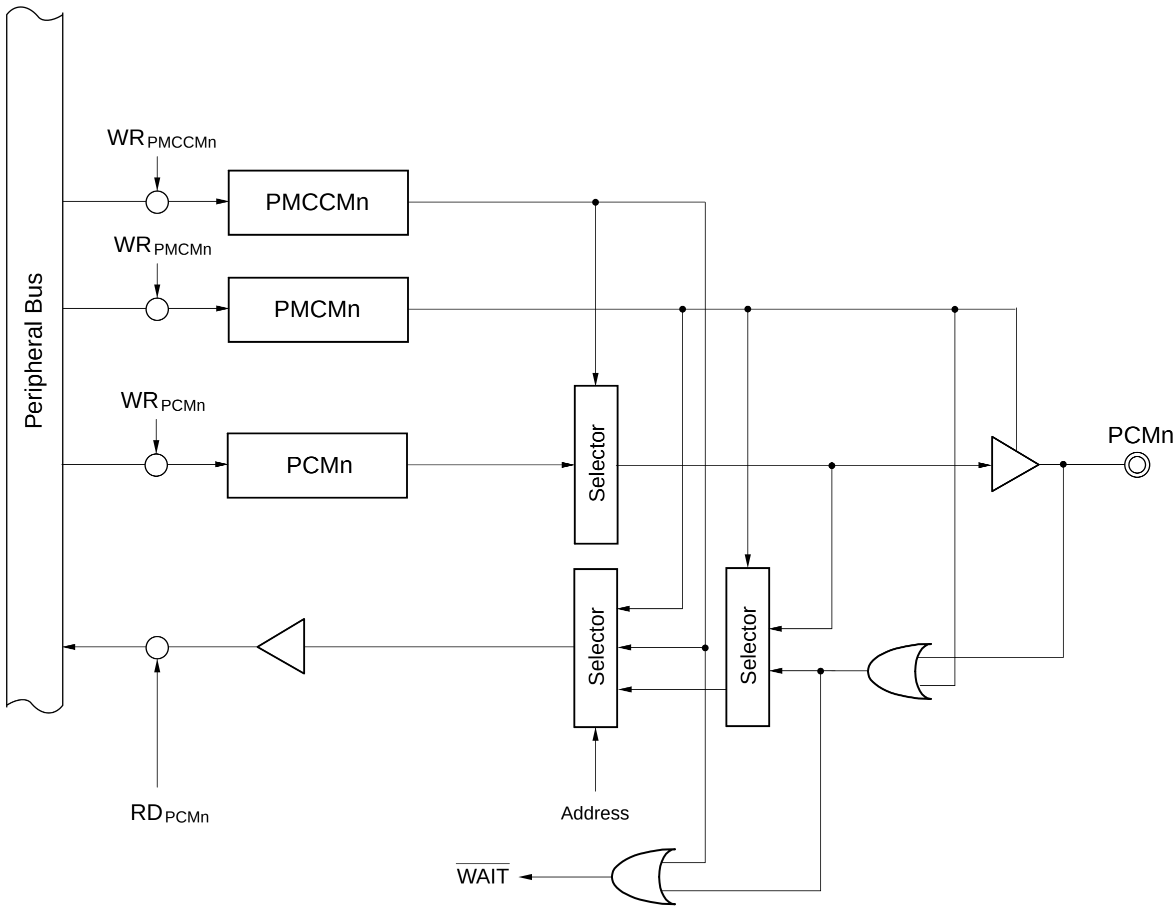
Remark: n = 0, 3, 4

Figure 16-5: Type D Block Diagram



Remark: n = 0, 1, 4

Figure 16-6: Type E Block Diagram



16.3 Pin Functions of Each Port

16.3.1 Port 1

Port 1 is a 8-bit input/output port in which input or output can be specified in 1-bit units. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read or written in 1-bit and 8-bit units.

Figure 16-7: Port 1 (P1)

	7	6	5	4	3	2	1	0	Address	At Reset
P1	P17	P16	P15	P14	P13	P12	P11	P10	FFFFF400H	00H[Note 2]

Bit Position	Bit Name	Function
7 to 0	P1n (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the P1 register is read, the pin levels at that time are read. Writing to the P1 register writes the values to that register. This does not affect the input pins.
In Output Mode: When the P1 register is read, the values of P1 are read. Writing to the P1 register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as the serial interface (UART1, FCAN1, FCAN2, FCAN3[Note 3]) input/output.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 1	P10	CRXD1	Serial interface (UART1, FCAN1, FCAN2, FCAN3[Note 2]) input/output.	A
	P11	CTXD1		
	P12	CRXD2		
	P13	CTXD2		
	P14	CRXD3[Note 2]		
	P15	CTXD3[Note 2]		
	P16	RXD1		
	P17	TXD1		

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register P1 is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

3. CAN module 3 is available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

(2) Setting in input/output mode and control mode

Port 1 is set in input/output mode using the port 1 mode register (PM1). In control mode, it is set using the port 1 mode control register (PMC1).

(a) Port 1 mode register (PM1)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-8: Port 1 Mode Register (PM1)

	7	6	5	4	3	2	1	0	Address	At Reset
PM1	PM17	PM16	PM15	PM14	PM13	PM12	PM11	PM10	FFFFF420H	FFH

Bit Position	Bit Name	Function
7 to 0	PM1n (n = 7 to 0)	Specifies input/output mode of P1n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 1 mode control register (PMC1)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-9: Port 1 Mode Control Register (PMC1)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC1	PMC17	PMC16	PMC15	PMC14	PMC13	PMC12	PMC11	PMC10	FFFFF440H	00H

Bit Position	Bit Name	Function
7	PMC17	Specifies operation mode of P17 pin 0: Input/output port mode 1: TXD1 output mode
6	PMC16	Specifies operation mode of P16 pin 0: Input/output port mode 1: RXD1 input mode
5	PMC15	Specifies operation mode of P15 pin 0: Input/output port mode 1: CTXD3 output mode
4	PMC14	Specifies operation mode of P14 pin 0: Input/output port mode 1: CRXD3 input mode
3	PMC13	Specifies operation mode of P13 pin 0: Input/output port mode 1: CTXD2 output mode
2	PMC12	Specifies operation mode of P12 pin 0: Input/output port mode 1: CRXD2 input mode
1	PMC11	Specifies operation mode of P11 pin 0: Input/output port mode 1: CTXD1 output mode
0	PMC10	Specifies operation mode of P10 pin 0: Input/output port mode 1: CRXD1 input mode

16.3.2 Port 2

Port 2 is an 8-bit input/output port in which input or output can be specified in 1-bit units. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read or written in 1-bit and 8-bit units.

Figure 16-10: Port 2 (P2)

	7	6	5	4	3	2	1	0	Address	At Reset
P2	P27	P26	P25	P24	P23	P22	P21	P20	FFFFF402H	00H[Note 2]

Bit Position	Bit Name	Function
7 to 0	P2n (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the P2 register is read, the pin levels at that time are read. Writing to the P2 register writes the values to that register. This does not affect the input pins.
In Output Mode: When the P2 register is read, the values of P2 are read. Writing to the P2 register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as the serial interface (UART0, CSI0,CS1) input/output.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register P2 is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 2	P20	SI0	Serial interface (UART1, CSI0, CSI1) input/output.	A
	P21	SO0		
	P22	SCK0		
	P23	SI1		
	P24	SO1		
	P25	SCK1		
	P26	RXD0		
	P27	TXD0		

(2) Setting in input/output mode and control mode

Port 2 is set in input/output mode using the port 1 mode register (PM2). In control mode, it is set using the port 2 mode control register (PMC2).

(a) Port 2 mode register (PM2)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-11: Port 2 Mode Register (PM2)

	7	6	5	4	3	2	1	0	Address	At Reset
PM2	PM27	PM26	PM25	PM24	PM23	PM22	PM21	PM20	FFFFF422H	FFH

Bit Position	Bit Name	Function
7 to 0	PM2n (n = 7 to 0)	Specifies input/output mode of P2n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 2 mode control register (PMC2)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-12: Port 2 Mode Control Register (PMC2)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC2	PMC27	PMC26	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20	FFFFF442H	00H

Bit Position	Bit Name	Function
7	PMC27	Specifies operation mode of P27 pin 0: Input/output port mode 1: TXD0 output mode
6	PMC26	Specifies operation mode of P26 pin 0: Input/output port mode 1: RXD0 input mode
5	PMC25	Specifies operation mode of P25 pin 0: Input/output port mode 1: $\overline{\text{SCK1}}$ input/output mode
4	PMC24	Specifies operation mode of P24 pin 0: Input/output port mode 1: SO1 output mode
3	PMC23	Specifies operation mode of P23 pin 0: Input/output port mode 1: SI1 input mode
2	PMC22	Specifies operation mode of P22 pin 0: Input/output port mode 1: $\overline{\text{SCK0}}$ input/output mode
1	PMC21	Specifies operation mode of P21 pin 0: Input/output port mode 1: SO0 output mode
0	PMC20	Specifies operation mode of P20 pin 0: Input/output port mode 1: SI0 input mode

16.3.3 Port 3

Port 3 is a 6-bit input/output port in which input or output can be specified in 1-bit units. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read or written in 1-bit and 8-bit units.

Figure 16-13: Port 3 (P3)

	7	6	5	4	3	2	1	0	Address	At Reset
P3	0	0	P35	P34	P33	P32	P31	P30	FFFFF404H	00H[Note 2]

Bit Position	Bit Name	Function
7 to 0	P3n (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the P3 register is read, the pin levels at that time are read. Writing to the P3 register writes the values to that register. This does not affect the input pins.
In Output Mode: When the P3 register is read, the values of P3 are read. Writing to the P3 register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as the real-time pulse unit (RPU) input/output and external interrupt request input.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register P3 is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 3	P30	TIG00/INTP00	Real-time pulse unit (RPU) input or external interrupt request input.	A
	P31	TOG01/TIG01		
	P32	TOG02/TIG02		
	P33	TOG03/TIG03		
	P34	TOG04/TIG04		
	P35	TIG05/INTP05		

(2) Setting in input/output mode and control mode

Port 3 is set in input/output mode using the port 3 mode register (PM3). In control mode, it is set using the port 3 mode control register (PMC3).

(a) Port 3 mode register (PM3)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-14: Port 3 Mode Register (PM3)

	7	6	5	4	3	2	1	0	Address	At Reset
PM3	0	0	PM35	PM34	PM33	PM32	PM31	PM30	FFFFF424H	3FH

Bit Position	Bit Name	Function
7 to 0	PM2n (n = 7 to 0)	Specifies input/output mode of P2n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 3 mode control register (PMC3)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-15: Port 3 Mode Control Register (PMC3)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC3	0	0	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	FFFFF444H	00H

Bit Position	Bit Name	Function
5	PMC35	Specifies operation mode of P35 pin 0: Input/output port mode 1: TIG05 input mode or external interrupt request (INTP05) input mode
4	PMC34	Specifies operation mode of P34 pin 0: Input/output port mode 1: TIG04/TOG04 input/output mode
3	PMC33	Specifies operation mode of P33 pin 0: Input/output port mode 1: TIG03/TOG03 input/output mode
2	PMC32	Specifies operation mode of P32 pin 0: Input/output port mode 1: TIG02/TOG02 input/output mode
1	PMC31	Specifies operation mode of P31 pin 0: Input/output port mode 1: TIG01/TOG01 input/output mode
0	PMC30	Specifies operation mode of P30 pin 0: Input/output port mode 1: TIG00 input mode or external interrupt request (INTP00) input mode

16.3.4 Port 4

Port 4 is a 6-bit input/output port in which input or output can be specified in 1-bit units. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read or written in 1-bit and 8-bit units.

Figure 16-16: Port 4 (P4)

	7	6	5	4	3	2	1	0	Address	At Reset
P4	0	0	P45	P44	P43	P42	P41	P40	FFFFF406H	00H[Note 2]

Bit Position	Bit Name	Function
7 to 0	P4n (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the P4 register is read, the pin levels at that time are read. Writing to the P4 register writes the values to that register. This does not affect the input pins.
In Output Mode: When the P4 register is read, the values of P4 are read. Writing to the P4 register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as the real-time pulse unit (RPU) input/output and external interrupt request input.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register P4 is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 4	P40	TIG10/INTP10	Real-time pulse unit (RPU) input or external interrupt request input.	A
	P41	TOG11/TIG11		
	P42	TOG12/TIG12		
	P43	TOG13/TIG13		
	P44	TOG14/TIG14		
	P45	TIG15/INTP15		

(2) Setting in input/output mode and control mode

Port 4 is set in input/output mode using the port 4 mode register (PM4). In control mode, it is set using the port 4 mode control register (PMC4).

(a) Port 4 mode register (PM4)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-17: Port 4 Mode Register (PM4)

	7	6	5	4	3	2	1	0	Address	At Reset
PM4	0	0	PM45	PM44	PM43	PM42	PM41	PM40	FFFFF426H	3FH

Bit Position	Bit Name	Function
7 to 0	PM4n (n = 7 to 0)	Specifies input/output mode of P4n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 4 mode control register (PMC4)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-18: Port 4 Mode Control Register (PMC4)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC4	0	0	PMC45	PMC44	PMC43	PMC42	PMC41	PMC40	FFFFF446H	00H

Bit Position	Bit Name	Function
5	PMC45	Specifies operation mode of P45 pin 0: Input/output port mode 1: TIG15 input mode or external interrupt request (INTP15) input mode
4	PMC44	Specifies operation mode of P44 pin 0: Input/output port mode 1: TIG14/TOG14 input/output mode
3	PMC43	Specifies operation mode of P43 pin 0: Input/output port mode 1: TIG13/TOG13 input/output mode
2	PMC42	Specifies operation mode of P42 pin 0: Input/output port mode 1: TIG12/TOG12 input/output mode
1	PMC41	Specifies operation mode of P41 pin 0: Input/output port mode 1: TIG11/TOG11 input/output mode
0	PMC40	Specifies operation mode of P40 pin 0: Input/output port mode 1: TIG10 input mode or external interrupt request (INTP10) input mode

16.3.5 Port 5

Port 5 is a 7-bit input/output port in which input or output can be specified in 1-bit units. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read or written in 1-bit and 8-bit units.

Figure 16-19: Port 5 (P5)

	7	6	5	4	3	2	1	0	Address	At Reset
P5	0	P56	P55	P54	P53	P52	P51	P50	FFFFF408H	00H[Note 2]

Bit Position	Bit Name	Function
7 to 0	P5n (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the P5 register is read, the pin levels at that time are read. Writing to the P5 register writes the values to that register. This does not affect the input pins.
In Output Mode: When the P5 register is read, the values of P5 are read. Writing to the P5 register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as the real-time pulse unit (RPU) input/output, as the serial interface (FCAN4[Note 3]) and as external interrupt request input.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 5	P50	CRXD4	Real-time pulse unit (RPU) input/output, serial interface (FCAN4[Note 3]) input/output or external interrupt request input.	A
	P51	CTXD4		
	P52	INTP4		
	P53	INTP5		
	P54	TI0/INTP20		
	P55	TI1/INTP21		
	P56	TO0		

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register P4 is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

3. CAN module 4 is available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

(2) Setting in input/output mode and control mode

Port 5 is set in input/output mode using the port 5 mode register (PM5). In control mode, it is set using the port 5 mode control register (PMC5).

(a) Port 5 mode register (PM5)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-20: Port 5 Mode Register (PM5)

	7	6	5	4	3	2	1	0	Address	At Reset
PM5	0	PM56	PM55	PM54	PM53	PM52	PM51	PM50	FFFFF428H	7FH

Bit Position	Bit Name	Function
7 to 0	PM5n (n = 7 to 0)	Specifies input/output mode of P5n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 5 mode control register (PMC5)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-21: Port 5 Mode Control Register (PMC5)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC5	0	PMC56	PMC55	PMC54	PMC53	PMC52	PMC51	PMC50	FFFFF448H	00H

Bit Position	Bit Name	Function
6	PMC56	Specifies operation mode of P55 pin 0: Input/output port mode 1: TO0 output mode
5	PMC55	Specifies operation mode of P55 pin 0: Input/output port mode 1: TI1 input mode or external interrupt request (INTP21) input mode
4	PMC54	Specifies operation mode of P54 pin 0: Input/output port mode 1: TI0 input mode or external interrupt request (INTP20) input mode
3	PMC53	Specifies operation mode of P53 pin 0: Input/output port mode 1: External interrupt request (INTP5) input mode
2	PMC52	Specifies operation mode of P52 pin 0: Input/output port mode 1: External interrupt request (INTP4) input mode
1	PMC51	Specifies operation mode of P51 pin 0: Input/output port mode 1: CTXD4[Note] output mode
0	PMC50	Specifies operation mode of P50 pin 0: Input/output port mode 1: CRXD4[Note] input mode

Note: CAN module 4 is available in the derivatives µPD703129 (A) and µPD703129 (A1) only.

16.3.6 Port 6

Port 6 is an 8-bit input/output port in which input or output can be specified in 1-bit units. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read or written in 1-bit and 8-bit units.

Figure 16-22: Port 6 (P6)

	7	6	5	4	3	2	1	0	Address	At Reset
P6	P67	P66	P65	P64	P63	P62	P61	P60	FFFFF40AH	00H[Note 2]

Bit Position	Bit Name	Function
7 to 0	P6n (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the P6 register is read, the pin levels at that time are read. Writing to the P6 register writes the values to that register. This does not affect the input pins.
In Output Mode: When the P6 register is read, the values of P6 are read. Writing to the P6 register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as the serial interface (CSI2) or as external interrupt request input.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register P6 is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port 6	P60	NMI	Serial interface (CSI2) input/output or external interrupt request input.	A
	P61	INTP0		
	P62	INTP1		
	P63	INTP2		
	P64	INTP3		
	P65	SI2		
	P66	SO2		
	P67	SCK2		

(2) Setting in input/output mode and control mode

Port 6 is set in input/output mode using the port 6 mode register (PM6). In control mode, it is set using the port 6 mode control register (PMC6).

(a) Port 6 mode register (PM6)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-23: Port 6 Mode Register (PM6)

	7	6	5	4	3	2	1	0	Address	At Reset
PM6	PM66	PM66	PM65	PM64	PM63	PM62	PM61	PM60	FFFFF42AH	FFH

Bit Position	Bit Name	Function
7 to 0	PM6n (n = 7 to 0)	Specifies input/output mode of P6n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port 6 mode control register (PMC6)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-24: Port 6 Mode Control Register (PMC6)

	7	6	5	4	3	2	1	0	Address	At Reset
PMC6	PMC66	PMC66	PMC65	PMC64	PMC63	PMC62	PMC61	PMC60	FFFFF44AH	00H

Bit Position	Bit Name	Function
7	PMC67	Specifies operation mode of P65 pin 0: Input/output port mode 1: $\overline{\text{SCK2}}$ input/output mode
6	PMC66	Specifies operation mode of P65 pin 0: Input/output port mode 1: SO2 output mode
5	PMC65	Specifies operation mode of P65 pin 0: Input/output port mode 1: SI2 input mode
4	PMC64	Specifies operation mode of P64 pin 0: Input/output port mode 1: External interrupt request (INTP3) input mode
3	PMC63	Specifies operation mode of P63 pin 0: Input/output port mode 1: External interrupt request (INTP2) input mode
2	PMC62	Specifies operation mode of P62 pin 0: Input/output port mode 1: External interrupt request (INTP1) input mode
1	PMC61	Specifies operation mode of P61 pin 0: Input/output port mode 1: External interrupt request (INTP0) input mode
0	PMC60	Specifies operation mode of P60 pin 0: Input/output port mode 1: NMI interrupt input mode

Remark: To avoid unintended disable of the $\overline{\text{NMI}}$, the bit PMC60 can only be set (1). Once the bit PMC60 is set (1), clearing that bit PMC60 (0) is not possible. The bit PMC60 is cleared by the generation of a $\overline{\text{RESET}}$.

16.3.7 Port 7

Port 7 is an 8-bit input port which is shared with the ADC input channels ANI0 to ANI7. Port 7 holds the digital input values of the A/D input channels ANI0 to ANI7 (P70 to P77). Port mode and port mode control are not available for port 7.

This register can be read in 1-bit or 8-bit units.

Figure 16-25: Port Function Register 7 (P7)

	7	6	5	4	3	2	1	0	Address	Initial value
P7	P77/ ANI7	P76/ ANI6	P75/ ANI5	P74/ ANI4	P73/ ANI3	P72/ ANI2	P71/ ANI1	P70/ ANI0	FFFFF40CH	undef.H

Bit Position	Bit Name	Function
7 to 0	P77 to P70	The bits P77 to P70 holds the digital input values of the A/D input channels ANI7 to ANI0.

Note: Reading the digital value of the analog input channel ANIx is disabled during A/D conversion operation.

Remark: x = 0 to 7

16.3.8 Port 7/8

Port 7/8 is a 16-bit input port which is shared with the ADC input channels ANI0 to ANI11. Port 7/8 holds the digital input values of the A/D input channels ANI0 to ANI11 (P70 to P77, P80 to P83). Port mode and port mode control are not available for port 7/8.

This register can only be read in 16-bit units.

Figure 16-26: Port Function Register 7/8 (P7/P8)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
P7/P8	0	0	0	0	P83/ ANI11	P82/ ANI10	P81/ ANI9	P80/ ANI8	P77/ ANI7	P76/ ANI6	P75/ ANI5	P74/ ANI4	P73/ ANI3	P72/ ANI2	P71/ ANI1	P70/ ANI0	FFFFF40CH	undef.

Bit Position	Bit Name	Function
11 to 8	P83 to P80	The bits P83 to P80 holds the digital input values of the A/D input channels ANI11 to ANI8.
7 to 0	P77 to P70	The bits P77 to P70 holds the digital input values of the A/D input channels ANI7 to ANI0.

Note: Reading the digital value of the analog input channel ANIx is disabled during A/D conversion operation.

Remark: x = 0 to 11

16.3.9 Port 9

Port 9 is an 8-bit input/output port in which input or output can be specified in 1-bit units. Port mode control is not available for port 9.

This register can be read or written in 1-bit and 8-bit units.

Figure 16-27: Port 9 (P9)

	7	6	5	4	3	2	1	0	Address	At Reset
P9	P97	P96	P95	P94	P93	P92	P91	P90	FFFFF40EH	00H[Note]

Bit Position	Bit Name	Function
7 to 0	P9n (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the P9 register is read, the pin levels at that time are read. Writing to the P9 register writes the values to that register. This does not affect the input pins.
In Output Mode: When the P9 register is read, the values of P9 are read. Writing to the P9 register writes the values to that register and those values are immediately output.

Note: The reset value of register P9 is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Setting in input/output mode

Port 9 is set in input/output mode using the port 9 mode register (PM9).

(a) Port 9 mode register (PM9)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-28: Port 9 Mode Register (PM9)

	7	6	5	4	3	2	1	0	Address	At Reset
PM9	PM97	PM96	PM95	PM94	PM93	PM92	PM91	PM90	FFFFF42EH	FFH

Bit Position	Bit Name	Function
7 to 0	PM9n (n = 7 to 0)	Specifies input/output mode of P9n pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

16.3.10 Port AH

Port AH is an 8-bit input/output port in which input or output can be specified in 1-bit units. After reset, the port AH pins operate as an address bus to address external memories respectively peripherals. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read in 1-bit and 8-bit units.

Figure 16-29: Port AH (PAH)

	7	6	5	4	3	2	1	0	Address	At Reset
PAH	PAH7	PAH6	PAH5	PAH4	PAH3	PAH2	PAH1	PAH0	FFFFF002H	00H[Note 2]

Bit Position	Bit Name	Function
7 to 0	PAHn (n = 7 to 0)	Input/output port

Remark: In Input Mode: When the PAH register is read, the pin levels at that time are read. Writing to the PAH register writes the values to that register. This does not affect the input pins.
In Output Mode: When the PAH register is read, the values of PAH are read. Writing to the PAH register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as an address bus.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register PAH is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port AH	PAH7 to PAH0	A23 to A16	Address bus	B

(2) Setting in input/output mode and control mode

Port AH is set in input/output mode using the port AH mode register (PMAH). In control mode, it is set using the port AH mode control register (PMCAH).

(a) Port AH mode register (PMAH)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-30: Port AH Mode Register (PMAH)

	7	6	5	4	3	2	1	0	Address	At Reset
PMAH	PMAH7	PMAH6	PMAH5	PMAH4	PMAH3	PMAH2	PMAH1	PMAH0	FFFF022FH	00H

Bit Position	Bit Name	Function
7 to 0	PMAHn (n = 7 to 0)	Specifies input/output mode of PAHn pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port AH mode control register (PMCAH)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-31: Port AH Mode Control Register (PMCAH)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCAH	PMCAH7	PMCAH6	PMCAH5	PMCAH4	PMCAH3	PMCAH2	PMCAH1	PMCAH0	FFFFF042H	FFH

Bit Position	Bit Name	Function
7 to 0	PMCAHn (n = 7 to 0)	Specifies operation mode of PMCAHn pin 0: Input/output port mode 1: A23 to A16 address output

16.3.11 Port CS

Port CS is a 3-bit input/output port in which input or output can be specified in 1-bit units. After reset, port PCS0 operates as a chip select output pin (CS0). The port pins PCS3 and PCS4 operate as port input after reset. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read in 1-bit and 8-bit units.

Figure 16-32: Port CS (PCS)

	7	6	5	4	3	2	1	0	Address	At Reset
PCS	0	0	0	PCS4	PCS3	0	0	PCS0	FFFFF008H	00H[Note 2]

Bit Position	Bit Name	Function
4, 3, 0	PCSn (n = 4, 3, 0)	Input/output port

Remark: In Input Mode: When the PCS register is read, the pin levels at that time are read. Writing to the PCS register writes the values to that register. This does not affect the input pins.
In Output Mode: When the PCS register is read, the values of PCS are read. Writing to the PCS register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, it also can operate as the chip select signal output when memory is accesses externally.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register PCS is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port AH	PCS4	CS0	Chip select signal output	C
	PCS3	CS3	Chip select signal output	
	PCS0	CS4	Chip select signal output	

Caution: In case that a port pin CS0, CS3 or CS4 operates as a chip select output port, it is recommended to plug in an external pull up resistor to that pin.

(2) Setting in input/output mode and control mode

Port CS is set in input/output mode using the port CS mode register (PMCS). In control mode, it is set using the port CS mode control register (PMCS).

(a) Port CS mode register (PMCS)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-33: Port CS Mode Register (PMCS)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCS	0	0	0	PMCS4	PMCS3	0	0	PMCS0	FFFFF028H	18H

Bit Position	Bit Name	Function
4	PMCS4	Specifies input/output mode of PCS4 pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)
3	PMCS3	Specifies input/output mode of PCS3 pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)
0	PMCS0	Specifies input/output mode of PCS0 pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port CS mode control register (PMCCS)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-34: Port CS Mode Control Register (PMCCS)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCCS	0	0	0	PMCCS4	PMCCS3	0	0	PMCCS0	FFFFF048H	01H

Bit Position	Bit Name	Function
4	PMCCS4	Specifies operation mode of PMCCS4 pin 0: Input/output port mode 1: $\overline{CS4}$ Chip select output
3	PMCCS4	Specifies operation mode of PMCCS3 pin 0: Input/output port mode 1: $\overline{CS3}$ Chip select output
0	PMCCS4	Specifies operation mode of PMCCS0 pin 0: Input/output port mode 1: $\overline{CS0}$ Chip select output

16.3.12 Port CT

Port CT is a 3-bit input/output port in which input or output can be specified in 1-bit units. After reset, port pin PCT4 operates as a read strobe signal output (RD). The port pins PCT0 and PCT1 operate as port input after reset. Each port bit can be independently configured to port input, port output or peripheral function[Note 1].

This register can be read in 1-bit and 8-bit units.

Figure 16-35: Port CT (PCT)

	7	6	5	4	3	2	1	0	Address	At Reset
PCT	0	0	0	PCT4	0	0	PCT1	PCT0	FFFFF00AH	00H[Note 2]

Bit Position	Bit Name	Function
4, 1, 0	PCTn (n = 4, 1, 0)	Input/output port

Remark: In Input Mode: When the PCT register is read, the pin levels at that time are read. Writing to the PCT register writes the values to that register. This does not affect the input pins.
In Output Mode: When the PCT register is read, the values of PCT are read. Writing to the PCT register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, PCT0 and PCT1 can operate as the write strobe signal outputs when memory is accessed externally. PCT4 can also operate as the read strobe signal input.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register PCT is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port CT	PCT4	RD	Read strobe output	D
	PCT1	UWR	Upper write strobe signal output	
	PCT0	LWR	Lower write strobe signal output	

Caution: In case that a port pin PCT0, PCT1 or PCT4 operates as a control signal for the external memory interface (LWR, UWR or RD), it is recommended to plug in an external pull up resistor to that pin

(2) Setting in input/output mode and control mode

Port CT is set in input/output mode using the port CT mode register (PMCT). In control mode, it is set using the port CT mode control register (PMCT).

(a) Port CT mode register (PMCT)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-36: Port CT Mode Register (PMCT)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCT	0	0	0	PMCT4	0	0	PMCT1	PMCT0	FFFFF02AH	03H

Bit Position	Bit Name	Function
4	PMCT4	Specifies input/output mode of PCT4 pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)
1	PMCT1	Specifies input/output mode of PCT1 pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)
0	PMCT0	Specifies input/output mode of PCT0 pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port mode control register (PMCCT)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-37: Port CT Mode Control Register (PMCCT)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCCT	0	0	0	PMCCT4	0	0	PMCCT1	PMCCT0	FFFFF04AH	10H

Bit Position	Bit Name	Function
4	PMCCT4	Specifies operation mode of PMCCT4 pin 0: Input/output port mode 1: $\overline{RD}$ Read strobe signal output
1	PMCCT1	Specifies operation mode of PMCCT1 pin 0: Input/output port mode 1: $\overline{UWR}$ Upper write strobe signal output
0	PMCCT0	Specifies operation mode of PMCCT0 pin 0: Input/output port mode 1: $\overline{LWR}$ Lower write strobe signal output

16.3.13 Port CM

Port CM is an 1-bit input/output port in which input or output can be specified in 1-bit units. After reset, port pin PCM0 operates as the wait insertion input (WAIT). This port bit can be configured to port input, port output or peripheral function[Note 1].

This register can be read in 1-bit and 8-bit units.

Figure 16-38: Port CM (PCM)

	7	6	5	4	3	2	1	0	Address	At Reset
PCM	0	0	0	0	0	0	0	PCM0	FFFFF00CH	00H[Note 2]

Bit Position	Bit Name	Function
0	PCM0	Input/output port

Remark: In Input Mode: When the PCM register is read, the pin levels at that time are read. Writing to the PCM register writes the values to that register. This does not affect the input pins.
In Output Mode: When the PCM register is read, the values of PCM are read. Writing to the PCM register writes the values to that register and those values are immediately output.

Besides functioning as a port, in control mode, PCM0 can operate as the wait insertion signal input when external slow memory/peripherals are connected.

Notes: 1. If using peripheral functions, the direction setting for the respective port bit is not forced automatically. Port bit direction has to be programmed according to the peripheral function requirement by setting the port mode register.

2. The reset value of register PCM is 00H. Due to the input mode of the port after reset, the read input value is determined by the port pins.

(1) Operation in control mode

Port		Alternate Pin Name	Remarks	Block Type
Port CM	PCM0	WAIT	Wait insertion signal input	E

Caution: In case that the port pin PCM0 operates as a control signal for the external memory interface (WAIT), it is recommended to plug in an external pull up resistor to that pin

(2) Setting in input/output mode and control mode

Port CM is set in input/output mode using the port CM mode register (PMCM). In control mode, it is set using the port CM mode control register (PMCM).

(a) Port CM mode register (PMCM)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-39: Port CM Mode Register (PMCM)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCM	0	0	0	0	0	0	0	PMCM0	FFFFF02CH	01H

Bit Position	Bit Name	Function
0	PMCM0	Specifies input/output mode of PCM0 pin. 0: Output mode (Output buffer on) 1: Input mode (Output buffer off)

(b) Port mode control register (PMCCM)

This register can be read or written in 8-bit or 1-bit units.

Figure 16-40: Port CM Mode Control Register (PMCCM)

	7	6	5	4	3	2	1	0	Address	At Reset
PMCCM	0	0	0	0	0	0	0	PMCCM0	FFFFF04CH	01H

Bit Position	Bit Name	Function
0	PMCCM0	Specifies operation mode of PMCCM0 pin 0: Input/output port mode 1: $\overline{\text{WAIT}}$ wait insertion input

Chapter 17 RESET

17.1 Reset Overview

Jupiter needs a system reset in order to initialize on power-up or re-initialize to escape from power save mode or system malfunction by Watchdog Timer. Regarding to the Mode setting and source of reset, different actions are performed on reset.

When a low level is input to the $\overline{\text{RESET}}$ pin, there is a system reset and each hardware item of the V850E/CA2 is initialized to its initial status.
When the $\overline{\text{RESET}}$ pin changes from low level to high level, reset status is released and the CPU starts program execution. The user has to initialize the contents of various registers as needed within the program.

17.2 Features

- Noise elimination of $\overline{\text{RESET}}$ pin using analog delay.

17.3 Pin Functions

During a system reset, most pins (all but the $\overline{\text{RESOUT}}$, V_{DDn}, V_{SSn}, CV_{DD}, CV_{SS}, AV_{DD}, AV_{REF} pins) enter the high impedance state.
Therefore, when memory is connected externally, a pull-up or pull-down resistor must be connected to the memory control pins of **alphabet ports (PCS, PCT)**. If no resistors are connected, the memory contents may be lost when these pins enter the high impedance state.
For the same reason, the output pins of the internal peripheral I/O function and other output port should be handled in the same manner.

Table 17-1 shows the operation status of each pin during Reset period.

Table 17-1: Operation Status of each pin during Reset period

Pin Function	RESET
D[15:0]	Hi-Z
A[23-0]	Hi-Z
$\overline{\text{CS}}$[4:3, 0]	Hi-Z
$\overline{\text{WR}}$[1:0]	Hi-Z
$\overline{\text{RD}}$	Hi-Z
$\overline{\text{WAIT}}$	---
$\overline{\text{RESOUT}}$	LOW
TIG05 to TIG00, TIG15 to TIG10, TIC01 to TIC00	N.A.
INTP05, INTP00, INTP15, INTP10, INTP21, INTP20, INTP5 to INTP0,NMI	N.A.
TOG04 to TOG01, TOG14 to TOG11, TOC0	N.A.
SO02, SO01,SO00	N.A.
SI02, SI01,SI00	N.A.
$\overline{\text{SCK02}}$, $\overline{\text{SCK01}}$,$\overline{\text{SCK00}}$	N.A.
RXD51, RXD50	N.A.
TXD51, TXD50	N.A.
FCRXD3 to FCRXD0**Note**	N.A.
FCTXD3 to FCTXD0**Note**	N.A.
ANI11 to ANI0	---
P1,P2,P3,P4,P5,P6,P9	Hi-Z
PAH[7:0], PCS[4,3,0], PCT[1:0], PCT[4],PCM[0]	N.A.

Remarks: 1. N.A.: This configuration is not available.

2. ---: Input data is not sampled.

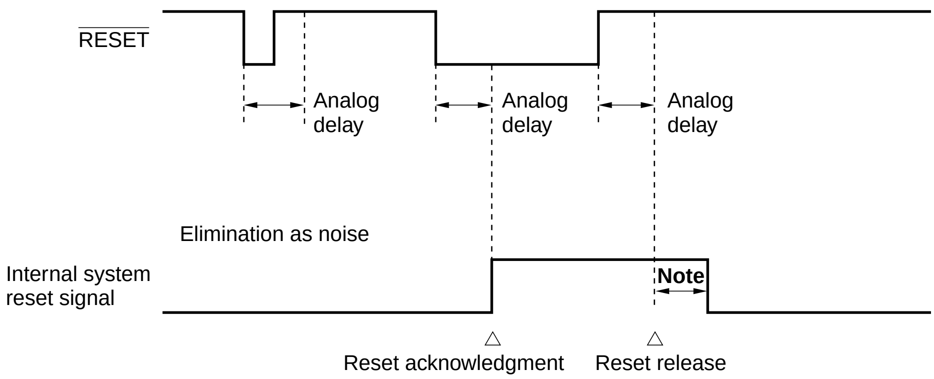
Note: FCTXD4 to FCTXD3 / FCRXD4 to FCRXD3 - only for µPD703129.

17.4 Reset by $\overline{\text{RESET}}$ Pin

If a low-level signal is input to the $\overline{\text{RESET}}$ pin, a system reset is performed and the hardware is initialized. When the $\overline{\text{RESET}}$ pin level changes from low to high, the Reset State is released and the program execution is started. All register will be initialized. The $\overline{\text{RESET}}$ pin incorporates a noise eliminator, which uses analogue delay to prevent malfunction due to noise.

(1) Reset signal acknowledgment

Figure 17-1: Reset signal acknowledgment

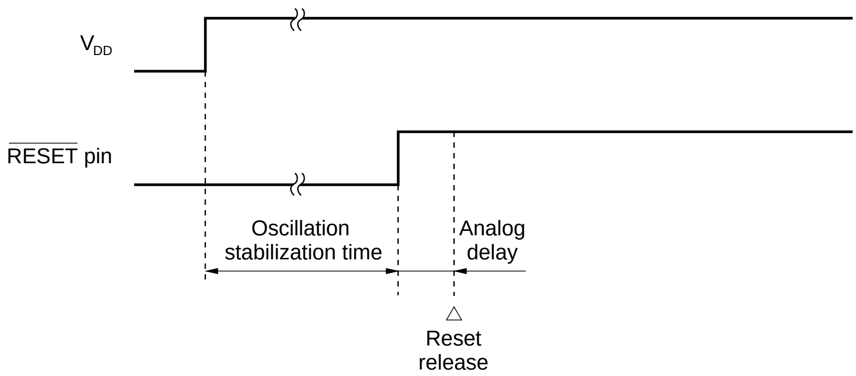


Note: The internal system reset signal keeps its active level for at least four system clock cycles after a $\overline{\text{RESET}}$ pin is released.

(2) Reset at power-on

A low level for the oscillator stabilization time has to be applied to the $\overline{\text{RESET}}$ pin.
This is to secure the clock stabilization time that is necessary after the power is turned on and before a reset signal can be acknowledged.
Please refer to the Electrical Data Sheet for Jupiter.

Figure 17-2: Reset at power-on



17.5 Reset by Watchdog Timer

Jupiter’s watchdog timer can be configured to generate a Reset in case watchdog time expires. This signal is expanded by the clock controller.
An output from clock controller is input into the Reset circuit. Oscillation stabilization time is not required after this reset.
Except WDT reset was triggered in sub-watch mode or stop mode. This case is handled by the clock controller.

17.6 Reset Output

Jupiter has an output $\overline{\text{RESOUT}}$ pin to indicate an internal system reset caused by $\overline{\text{RESET}}$ pin or Watchdog timer. This reset output is used to terminate any ongoing internal erasing or programming operation in the external FLASH memory connected to the Jupiter device.

17.7 Initialization

Initialize the contents of each register as needed within a program.
Table 17-2 shows the initial values of the CPU, internal RAM, and on-chip peripheral I/O’s after reset.

Table 17-2: Initial Values of CPU and Internal RAM After Reset

On-Chip Hardware		Register Name	Initial Value After Reset
CPU	Program registers	General-purpose register (r0)	00000000H
		General-purpose registers (r1 to r31)	Undefined
		Program counter (PC)	00000000H
	System registers	Status save registers during interrupt (EIPC, EIPSW)	Undefined
		Status save registers during NMI (FEPC, FEPSW)	Undefined
		Interrupt cause register (ECR)	00000000H
		Program status word (PSW)	00000020H
		Status save registers during CALLT execution (CTPC, CTPSW)	Undefined
		Status save registers during exception/debug trap (DBPC, DBPSW)	Undefined
		CALLT base pointer (CTBP)	Undefined
Internal RAM		-	Undefined

Caution: In the table above, “Undefined” means either undefined at the time of a power-on reset or undefined due to data destruction when $\overline{\text{RESET}}\downarrow$ input and data write timing are synchronized. On a $\overline{\text{RESET}}\downarrow$ other than this, data is maintained in its previous status.

Appendix A List of Instruction Sets

Figure A-1: How to Read Instruction Set List

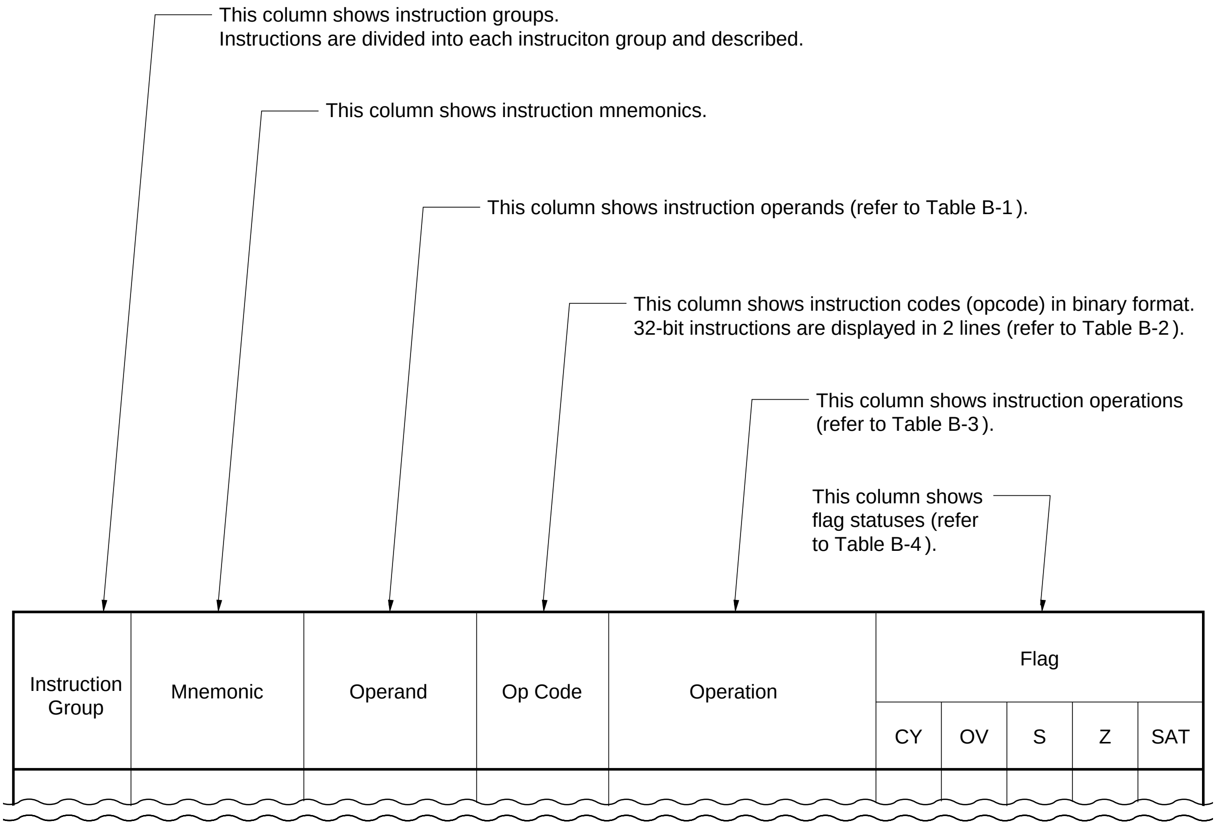


Table A-1: Symbols in Operand Description

Symbol	Description
reg1	General register (r0 to r31): Used as source register
reg2	General register (r0 to r31): Mainly used as destination register
ep	Element pointer (r30)
bit#3	3-bit data for bit number specification
imm×	×-bit immediate data
disp×	×-bit displacement
regID	System register number
vector	5-bit data that specifies trap vector number (00H to 1FH)
cccc	4-bit data that indicates condition code

Table A-2: Symbols Used for Op Code

Symbol	Description
R	1-bit data of code that specifies reg1 or regID
r	1-bit data of code that specifies reg2
d	1-bit data of displacement
i	1-bit data of immediate data
cccc	4-bit data that indicates condition code
bbb	3-bit data that specifies bit number

Table A-3: Symbols Used for Operation Description

Symbol	Description
←	Assignment
GR[ ]	General register
SR[ ]	System register
zero-extend (n)	Zero-extends n to word length.
sign-extend (n)	Sign-extends n to word length.
load-memory (a,b)	Reads data of size b from address a.
store-memory (a,b,c)	Writes data b of size c to address a.
load-memory-bit (a,b)	Reads bit b from address a.
store-memory-bit (a,b,c)	Writes c to bit b of address a
saturated (n)	Performs saturated processing of n. (n is 2ís complements). Result of calculation of n: If n is n ≥ 7FFFFFFFH as result of calculation, 7FFFFFFFH. If n is n ≤ 80000000H as result of calculation, 80000000H.
result	Reflects result to a flag.
Byte	Byte (8 bits)
Halfword	Half-word (16 bits)
Word	Word (32 bits)
+	Add
-	Subtract
||	Bit concatenation
×	Multiply
÷	Divide
AND	Logical product
OR	Logical sum
XOR	Exclusive logical sum
NOT	Logical negate
logically shift left by	Logical left shift
logically shift right by	Logical right shift
arithmetically shift right by	Arithmetic right shift

Table A-4: Symbols Used for Flag Operation

Symbol	Description
(blank)	Not affected
0	Cleared to 0
×	Set of cleared according to result
R	Previously saved value is restored

Table A-5: Condition Codes

Condition Name (cond)	Condition Code (cccc)	Conditional Expression	Description
V	0000	OV = 1	Overflow
NV	1000	OV = 0	No overflow
C/L	0001	CY = 1	Carry Lower (Less than)
NC/NL	1001	CY = 0	No carry No lower (Greater than or equal)
Z/E	0010	Z = 1	Zero Equal
NZ/NE	1010	Z = 0	Not zero Not equal
NH	0011	(CY OR Z) = 1	Not higher (Less than or equal)
H	1011	(CY OR Z) = 0	Higher (Greater than)
N	0100	S = 1	Negative
P	1100	S = 0	Positive
T	0101	-	Always (unconditional)
SA	1101	SAT = 1	Saturated
LT	0110	(S XOR OV) = 1	Less than signed
GE	1110	(S XOR OV) = 0	Greater than or equal signed
LE	0111	((S XOR OV) OR Z) = 1	Less than or equal signed
GT	1111	((S XOR OV) OR Z) = 0	Greater than signed

Table A-6: Instruction Set List (1/7)

Instruction Group	Mne-monic	Operand	Opcode	Operation	Flag				
					CY	OV	S	Z	SAT
Load/store	SLD.B	disp7 [ep], reg2	rrrrr0110 ddddddd	adr ← ep + zero-extend (disp7) GR [reg2] ← sign-extend (Load-memory (adr, Byte))					
	SLD.H	disp8 [ep], reg2	rrrrr1000 ddddddd **Note 1**	adr ← ep + zero-extend (disp8) GR [reg2] ← sign-extend (Load-memory (adr, Halfword))					
	SLD.W	disp8 [ep], reg2	rrrrr1010 dddddd0 **Note 2**	adr ← ep + zero-extend (disp8) GR [reg2] ← Load-memory (adr, Word)					
	LD.B	disp16 [reg1], reg2	rrrrr111000 RRRRR dddddddd dddddddd	adr ← GR [reg1] + sign-extend (disp16) GR [reg2] ← sign-extend (Load-memory (adr, Byte))					
	LD.H	disp16 [reg1], reg2	rrrrr1110 01RRRRR dddddddd ddddddd0 **Note 3**	adr ← GR [reg1] + sign-extend (disp16) GR [reg2] ← sign-extend (Load-memory (adr, Halfword))					
	LD.W	disp16 [reg1], reg2	rrrrr1110 01RRRRR dddddddd ddddddd1 **Note 3**	adr ← GR [reg1] + sign-extend (disp16) GR [reg2] ← Load-memory (adr, Word))					
	SST.B	reg2, disp7 [ep]	rrrrr0111 ddddddd	adr ← ep + zero-extend (disp7) Store-memory (adr, GR [reg2], Byte)					
	SST.H	reg2, disp8 [ep]	rrrrr1001 ddddddd **Note 1**	adr ← ep + zero-extend (disp8) Store-memory (adr, GR [reg2], Halfword)					
	SST.W	reg2, disp8 [ep]	rrrrr1010 dddddd1 **Note 2**	adr ← ep + zero-extend (disp8) Store-memory (adr, GR [reg2], Word)					
	ST.B	reg2, disp16 [reg1]	rrrrr1110 10RRRRR dddddddd dddddddd	adr ← GR [reg1] + sign-extend (disp16) Store-memory (adr, GR [reg2], Byte)					

Notes: 1. ddddddd is the higher 7 bits of disp8.

2. dddddd is the higher 6 bits of disp8.

3. ddddddddddddddd is the higher 15 bits of disp16.

4. Only the lower half-word data is valid.

5. ddddddddddddddddddddd is the higher 21 bits of dip22.

6. dddddddd is the higher 8 bits of disp9.

7. The op code of this instruction uses the field of reg1 through the source register is shown as reg2 in the above table. Therefore, the meaning of register specification for mnemonic description and op code is different from that of the other instructions
rrr = regID specification
RRRRR = reg2 specification

Table A-6: Instruction Set List (2/7)

Instruction Group	Mne-monic	Operand	Opcode	Operation	Flag				
					CY	OV	S	Z	SAT
Load/store	ST.H	reg2, disp16 [reg1]	rrrrr1110 11RRRRR ddddddddd dddddd0 **Note 3**	adr ← GR [reg1] + sign-extend (disp16) Store-memory (adr, GR [reg2], Halfword)					
	ST.W	reg2, disp16 [reg1]	rrrrr1110 11RRRRR ddddddddd dddddd1 **Note 3**	adr ← GR [reg1] + sign-extend (disp16) Store-memory (adr, GR [reg2], Word)					
Arithmetic operation	MOV	reg1, reg2	rrrrr0000 00RRRRR	GR [reg2] ← GR [reg1]					
	MOV	imm5, reg2	rrrrr0100 00iiiii	GR [reg2] ← sign-extend (imm5)					
	MOVHI	imm16, reg1, reg2	rrrrr1100 10RRRRR iiiiiiii- iiiiiiii	GR [reg2] ← GR [reg1] + (imm16 || 0^{16})					
	MOVEA	imm16, reg1, reg2	rrrrr1100 01RRRRR iiiiiiii- iiiiiiii	GR [reg2] ← GR [reg1] + sign-extend (imm16)					
	LD.H	disp16 [reg1], reg2	rrrrr1110 01RRRRR ddddddddd dddddd0 **Note 3**	adr ← GR [reg1] + sign-extend (disp16) GR [reg2] ← sign-extend (Load-memory (adr, Halfword))					
	LD.W	disp16 [reg1], reg2	rrrrr1110 01RRRRR ddddddddd dddddd1 **Note 3**	adr ← GR [reg1] + sign-extend (disp16) GR [reg2] ← Load-memory (adr, Word))					
	SST.B	reg2, disp7 [ep]	rrrrr0111 ddddddd	adr ← ep + zero-extend (disp7) Store-memory (adr, GR [reg2], Byte)					
	ADD	reg1, reg2	rrrrr001110 RRRRR	GR [reg2] ← GR [reg2] + GR [reg1]					
	ADD	imm5, reg2	rrrrr010010i iiii	GR [reg2] ← GR [reg2] + sign-extend (imm5)	×	×	×	×	

Notes: 1. ddddddd is the higher 7 bits of disp8.

2. dddddd is the higher 6 bits of disp8.

3. ddddddddddddddd is the higher 15 bits of disp16.

4. Only the lower half-word data is valid.

5. ddddddddddddddddddddd is the higher 21 bits of dip22.

6. dddddddd is the higher 8 bits of disp9.

7. The op code of this instruction uses the field of reg1 through the source register is shown as reg2 in the above table. Therefore, the meaning of register specification for mnemonic description and op code is different from that of the other instructions
rrr = regID specification
RRRRR = reg2 specification

Table A-6: Instruction Set List (3/7)

Instruction Group	Mne-monic	Operand	Opcode	Operation	Flag				
					CY	OV	S	Z	SAT
Arithmetic operation	ADDI	imm16, reg1, reg2	rrrrr110 000RRRRR iiiiiiii iiiiiiii	GR [reg2] ← GR [reg1] + sign-extend (imm16)	×	×	×	×	
	SUB	reg1, reg2	rrrrr001 101RRRRR	GR [reg2] ← GR [reg2] - GR [reg1]	×	×	×	×	
	SUBR	reg1, reg2	rrrrr001 100RRRRR	GR [reg2] ← GR [reg1] - GR [reg2]	×	×	×	×	
	MULH	reg1,reg2	rrrrr000 111RRRRR	GR [reg2] ← GR [reg2] Note 4 × GR [reg1] Note 4 (Signed multiplication)	×	×	×	×	
	MULH	imm5, reg2	rrrrr010 111iiiii	GR [reg2] ← GR [reg2] Note 4 × sign-extend (imm5) (Signed multiplication)					
	MULHI	imm16, reg1, reg2	rrrrr110 111RRRRR iiiiiiii iiiiiiii	GR [reg2] ← GR [reg1] Note 4 × imm16 (signed multiplication)					
	DIVH	reg1, reg2	rrrrr000 010RRRRR	GR [reg2] ← GR [reg2] ÷ GR [reg2] Note 4 (Signed division)					
	CMP	reg1, reg2	rrrrr001 111RRRRR	result ← GR [reg2] - GR [reg1]		×	×	×	
	CMP	imm5, reg2	rrrrr010 011iiiii	result ← GR [reg2] - sign-extend (imm5)	×	×	×	×	
	SETF	cccc, reg2	rrrrr111 1110cccc 00000000 00000000	if conditions are satisfied then GR [reg2] ← 00000001H else GR [reg2] ← 00000000H	×	×	×	×	
Saturated operation	SAT-ADD	reg1, reg2	rrrrr000 110RRRRR	GR [reg2] ← saturated (GR [reg2] + GR [reg1])					
	SAT-ADD	imm5, reg2	rrrrr010 001iiiii	GR [reg2] ← saturated (GR [reg2] + sign-extend (imm5))	×	×	×	×	×
	SAT-SUB	reg1, reg2	rrrrr000 101RRRRR	GR [reg2] ← saturated (GR [reg2] - GR [reg1])	×	×	×	×	×

Notes: 1. ddddddd is the higher 7 bits of disp8.

2. dddddd is the higher 6 bits of disp8.

3. ddddddddddddddd is the higher 15 bits of disp16.

4. Only the lower half-word data is valid.

5. ddddddddddddddddddddd is the higher 21 bits of dip22.

6. dddddddd is the higher 8 bits of disp9.

7. The op code of this instruction uses the field of reg1 through the source register is shown as reg2 in the above table. Therefore, the meaning of register specification for mnemonic description and op code is different from that of the other instructions
rrr = regID specification
RRRRR = reg2 specification

Table A-6: Instruction Set List (4/7)

Instruction Group	Mne-monic	Operand	Opcode	Operation	Flag CY	OV	S	Z	SAT
Saturated operation	SAT-SUBI	imm16, reg1, reg2	rrrrr110 011RRRRR iiiiiiii iiiiiiii	GR [reg2] ← saturated (GR [reg1] - sign-extend (imm16))	×	×	×	×	×
	SAT-SUBR	reg1, reg2	rrrrr000 100RRRRR	GR [reg2] ← saturated (GR [reg1] - GR [reg2])	×	×	×	×	×
Logic operation	TST	reg1, reg2	rrrrr001 011RRRRR	result ← GR [reg2] AND GR [reg1]	×	×	×	×	×
	OR	reg1, reg2	rrrrr001 000RRRRR	GR [reg2] ← GR [reg2] OR GR [reg1]		0	×	×	
	ORI	imm16, reg1, reg2	rrrrr110 100RRRRR iiiiiiii iiiiiiii	GR [reg2] ← GR [reg1] OR zero-extend (imm16)		0	×	×	
	AND	reg1, reg2	rrrrr001 010RRRRR	GR [reg2] ← GR [reg2] AND GR [reg1]		0	×	×	
	ANDI	imm16, reg1, reg2	rrrrr110 110RRRRR iiiiiiii iiiiiiii	GR [reg2] ← GR [reg1] AND zero-extend (imm16)		0	×	×	
	XOR	reg1, reg2	rrrrr0010 01RRRRR	GR [reg2] ← GR [reg2] XOR GR [reg1]		0	×	×	
	XORI	imm16, reg1, reg2	rrrrr1101 01RRRRR iiiiiiii-iiiiiiii	GR [reg2] ← GR [reg1] XOR zero-extend (imm16)		0	×	×	
	NOT	reg1, reg2	rrrrr0000 01RRRRR	GR [reg2] ← NOT (GR [reg1])		0	×	×	
	SHL	reg1, reg2	rrrrr1111 11RRRRR 000000001 1000000	GR [reg2] ← GR [reg2] logically shift left by GR [reg1])	×	0	×	×	
	SHL	imm5, reg2	rrrrr0101 10iiiii	GR [reg2] ← GR [reg2] logically shift left by zero-extend (imm5)	×	0	×	×	

Notes: 1. ddddddd is the higher 7 bits of disp8.

2. dddddd is the higher 6 bits of disp8.

3. ddddddddddddddd is the higher 15 bits of disp16.

4. Only the lower half-word data is valid.

5. ddddddddddddddddddddd is the higher 21 bits of dip22.

6. dddddddd is the higher 8 bits of disp9.

7. The op code of this instruction uses the field of reg1 through the source register is shown as reg2 in the above table. Therefore, the meaning of register specification for mnemonic description and op code is different from that of the other instructions
rrr = regID specification
RRRRR = reg2 specification

Table A-6: Instruction Set List (5/7)

Instruction Group	Mnemonic	Operand	Opcode	Operation	Flag CY	OV	S	Z	SAT
Logic operation	SHR	reg1, reg2	rrrrr1111 111cccc 000000001 0000000	GR [reg2] ← GR [reg2] logically shift right by GR [reg1]	×	0	×	×	
	SHR	imm5, reg2	rrrrr0101 00iiiii	GR [reg2] ← GR [reg2] logically shift right by zero-extend (imm5)	×	0	×	×	
	SAR	reg1, reg2	rrrrr1111 11RRRRR 000000001 0100000	GR [reg2] ← GR [reg2] arithmetically shift right by GR [reg1]	×	0	×	×	
	SAR	imm5, reg2	rrrrr0101 01iiiii	GR [reg2] ← GR [reg2] arithmetically shift right by zero-extend (imm5)	×	0	×	×	
Jump	JMP	[reg1]	000000000 11RRRRR	PC ← GR [reg1]					
	JR	disp22	000001111 0dddddd ddddddddd dddddd0 **Note 5**	PC ← PC + sign-extend (disp22)					
	JARL	disp22, reg2	rrrrr1111 0dddddd ddddddddd dddddd0 **Note 5**	GR [reg2] ← PC + 4 PC ← PC + sign-extend (disp22)					
	Bcond	disp9	ddddd1011 dddcccc **Note 6**	if conditions are satisfied then PC ← PC + sign-extend (disp9)					
Bit manipulate	SET1	bit#3, disp16 [reg1]	00bbb1111 10RRRRR ddddddddd ddddddd	adr ← GR [reg1] + sign-extend (disp16) Z flag ← Not (Load-memory-bit (adr, bit#3) Store memory-bit (adr, bit#3, 1)				×	
	CLR1	bit#3, disp16 [reg1]	10bbb1111 10RRRRR ddddddddd ddddddd	adr ← GR [reg1] + sign-extend (disp16) Z flag ← Not (Load-memory-bit (adr, bit#3)) Store memory-bit (adr, bit#3, 0)				×	

Notes: 1. dddddd is the higher 7 bits of disp8.

2. ddddd is the higher 6 bits of disp8.

3. ddddddddddddddd is the higher 15 bits of disp16.

4. Only the lower half-word data is valid.

5. ddddddddddddddddddddd is the higher 21 bits of dip22.

6. dddddddd is the higher 8 bits of disp9.

7. The op code of this instruction uses the field of reg1 through the source register is shown as reg2 in the above table. Therefore, the meaning of register specification for mnemonic description and op code is different from that of the other instructions
rrr = regID specification
RRRRR = reg2 specification

Table A-6: Instruction Set List (6/7)

Instruction Group	Mne-monic	Operand	Opcode	Operation	Flag				
					CY	OV	S	Z	SAT
Bit manip-ulate	NOT1	bit#3, disp16 [reg1]	01bbb1111 10RRRRR ddddddddd ddddddd	adr ← GR [reg1] + sign-extend (disp16) Z flag ← Not (Load-memory-bit (adr, bit#3)) Store-memory-bit (adr, bit#3, Z flag)				×	
	TST1	bit#3, disp16 [reg1]	11bbb1111 10RRRRR ddddddddd ddddddd	adr ← GR [reg1] + sign-extend (disp16) Z flag ← Not (Load-memory-bit (adr, bit#3))				×	
Special	LDSR	reg2, regID	rrrrr1111 11RRRRR 000000000 0100000 **Note 7**	SR [regID] ←GR [reg2]					
				regID = EIPSW, FEPSW					
				regID = PSW	×	×	×	×	×
	STSR	regID, reg2	rrrrr1111 11RRRRR 000000000 1000000	GR [reg2] ← SR [regID]					
	TRAP	vector	000001111 11iiiii 000000010 0000000	EIPC ← PC + 4 (Restored PC) EIPSW ← PSW ECR.EICC ← Interrupt code PSW.EP ← 1 PSW.ID ← 1 PC ← 00000040H (vector = 00H to 0FH) 00000050H (vector = 10H to 1FH)					
	RETI		000001111 1100000 000000010 1000000	if PSW.EP = 1 then PC ← EIPC PSW ← EIPSW else if PSW.NP = 1 then PC ← FEPC PSW ← FEPSW else PC ← EIPC PSW ← EIPSW	R	R	R	R	R
	HALT		000001111 1100000 000000010 0100000	Stops					

Notes: 1. ddddddd is the higher 7 bits of disp8.

2. dddddd is the higher 6 bits of disp8.

3. ddddddddddddddd is the higher 15 bits of disp16.

4. Only the lower half-word data is valid.

5. ddddddddddddddddddddd is the higher 21 bits of dip22.

6. dddddddd is the higher 8 bits of disp9.

7. The op code of this instruction uses the field of reg1 through the source register is shown as reg2 in the above table. Therefore, the meaning of register specification for mnemonic description and op code is different from that of the other instructions
rrr = regID specification
RRRRR = reg2 specification

Table A-6: Instruction Set List (7/7)

Instruction Group	Mne-monic	Operand	Opcode	Operation	Flag				
					CY	OV	S	Z	SAT
	DI		000001111 1100000 000000010 1100000	PSW.ID ← 1 (Maskable interrupt disabled)					
	EI		100001111 1100000 000000010 1100000	PSW.ID ← 0 (Maskable interrupt enabled)					
	NOP		000000000 0000000	Uses 1 clock cycle without doing anything					

Notes: 1. ddddddd is the higher 7 bits of disp8.

2. dddddd is the higher 6 bits of disp8.

3. ddddddddddddddd is the higher 15 bits of disp16.

4. Only the lower half-word data is valid.

5. ddddddddddddddddddddd is the higher 21 bits of dip22.

6. dddddddd is the higher 8 bits of disp9.

7. The op code of this instruction uses the field of reg1 through the source register is shown as reg2 in the above table. Therefore, the meaning of register specification for mnemonic description and op code is different from that of the other instructions
rrr = regID specification
RRRRR = reg2 specification

[MEMO]

Appendix B Index

Numerics

2-way-associative instruction memory . . . 155

A

A/D conversion result register . . . 525, 531
A/D conversion termination . . . 536
A/D Converter Mode Register . . . 528
A/D converter mode register . . . 528
A/D converter register . . . 530
A/D trigger mode . . . 523
ADCR . . . 525, 531
address set up wait states . . . 137
address setup wait control register . . . 132
address space . . . 65
address wait control register . . . 132
ADM . . . 528
ADS . . . 530
analog delay . . . 583
Analog inputs . . . 523
ANI00 to ANI11 . . . 49
ASC . . . 132
ASIF0, ASIF1 . . . 371
ASIM0 . . . 367
ASIM1 . . . 367
ASIS0 to ASIS2 . . . 370
Asynchronous reset . . . 275
Asynchronous serial interface mode registers . . . 367
Asynchronous serial interface status registers . . . 370
Asynchronous serial interface transmission status registers . . . 371
Autofill Function 165
autofill function 155
AVDD . . . 50, 525
AVREF . . . 50, 525
AVSS . . . 50, 525

B

Baud rate generator control registers . . . 386
BCC . . . 134
BCT0 . . . 115
BCT1 . . . 115
BEC . . . 117
BHC . . . 118
Bit manipulation instructions . . . 24
Boot Loader . . . 24
Boot-Loader . . . 62
BPC . . . 84, 114
BRG0, BRG1 . . . 423
BRGC0, BRGC1 . . . 386
BSC . . . 116
Bus cycle configuration registers 0, 1(BCT0, BCT1) . . . 115
Bus cycle control register . . . 134
Bus size configuration register . . . 116
Byte access (8 bits) . . . 119

C

Cache configuration register (BHC) . . . 118
CALLT . . . 58
Capture operation . . . 285
capture trigger edge detection . . . 327
Capture/compare registers . . . 276
CCC00 . . . 276
CCC01 . . . 276
Chip area selection control registers 0, 1(CSC0, CSC1) . . . 111
CKC . . . 241, 243, 244, 245, 247, 248, 249, 250
CKSR0, CKSR1 . . . 385
Clock Control Register . . . 241
Clock control register . . . 241, 243, 244, 245, 247, 248, 249, 250
Clock select registers . . . 385
Clock sources . . . 239
clock stabilization time . . . 586
Clocked serial interface clock selection registers . . . 396
Clocked serial interface initial transmission buffer register . . . 404
Clocked serial interface initial transmission buffer registers Low . . . 405
Clocked serial interface mode registers . . . 395
Clocked serial interface read-only reception buffer registers . . . 400
Clocked serial interface read-only reception buffer registers Low . . . 401
Clocked serial interface reception buffer registers . . . 398
Clocked serial interface reception buffer registers Low . . . 399
Clocked serial interface transmission buffer registers . . . 402
Clocked serial interface transmission buffer registers Low . . . 403
COM0 to COM1 . . . 49
Command Register . . . 104
Compare operation . . . 289, 302, 337
Continuous transmission . . . 377
conversion time . . . 528
Core . . . 24
Count operation . . . 283
CSC0 . . . 111
CSC1 . . . 111
CSIC0 to CSIC2 . . . 396
CSIM0 to CSIM2 . . . 395
CTBP . . . 58
CTPC . . . 58
CTPSW . . . 58
Current consumption . . . 541
CVDD . . . 49
CVSS . . . 49
Cycle measurement . . . 287

D

D/A converter . . . 524
DADC0 to DADC3 . . . 175
data frame . . . 375
Data space . . . 67
Data wait control registers 0, 1 . . . 131
DBC0 . . . 174
DBC3 . . . 174
DBPC . . . 58
DBPSW . . . 58
DCHC0 to DCHC3 . . . 177
DDAH0 . . . 172
DDAH3 . . . 172

DDAL0 . . . 173
DDAL3 . . . 173
DDIS . . . 178
debug trap . . . 232
dedicated baud rate generator . . . 364
Dedicated baud rate generators (BRG) . . . 384
Dedicated baud rate generators 0, 1 . . . 423
DMA . . . 169
DMA addressing control registers 0 to 3 . . . 175
DMA channel control registers 0 to 3 . . . 177
DMA controller (DMAC) . . . 169
Block transfer mode . . . 192
Bus cycle state transition . . . 186
Bus states . . . 184
Channel priorities . . . 194
Forcible interruption . . . 195
Forcible termination . . . 196
Next address setting function . . . 183
Precautions . . . 198
Single transfer mode . . . 187
Single-step transfer mode . . . 189
Transfer end . . . 197
Transfer mode . . . 187
Transfer object . . . 193
Transfer start factors . . . 194
Transfer types . . . 193
Two-cycle transfer . . . 193
DMA destination address registers . . . 172, 173
DMA disable status register . . . 178
DMA restart register . . . 178
DMA source address registers . . . 170, 171
DMA transfer count registers . . . 174
DMA trigger factor registers 0 to 3 . . . 179, 180, 181, 182
DMAC . . . 169
DN . . . 446, 447, 451, 464, 475, 477, 481, 482, 489, 512
DRST . . . 178
DSAH0 . . . 170
DSAH3 . . . 170
DSAL0 . . . 171
DSAL3 . . . 171
DTFR0 to DTFR3 . . . 179, 180, 181, 182
DWC0 . . . 145
DWC0, DWC1 . . . 131
DWC1 . . . 145

E

ECR . . . 58, 59
edge detection circuit . . . 346
Edge noise elimination . . . 346
EIPC . . . 58
EIPSW . . . 58
Element pointer . . . 57
Endian configuration register . . . 117
EP . . . 229
escape from power save mode . . . 583
Even parity . . . 368
event counter . . . 274, 306
Exception status flag . . . 229

exception table 70
Exception trap 230
External I/O 137
External ROM 137
External signal cycle measurement 271
External wait function 133

F
FCAN
  Baud rate setting 509
  Data new flag 446, 447, 451, 464, 475, 477, 481, 482, 489, 512
  Priority storage 446
  Time stamp 441, 442
FEPC 58
FEPSW 58
fixed time intervals 353
Framing error 364
free run 329, 331
free-run mode 322
free-running timer 274, 275
Full-duplex 364, 375
full-duplex communication 363
Full-scale Error 544
Functions of each port 547
Functions of each port pin on reset and registers that set port or control mode 548

G
GCCn0 311
GCCn5 311
GCCnm 312
General registers 56, 57
general registers 55, 57
Global pointer 57

H
Halfword access (16 bits) 121
HALT mode 253, 256
handler addresses 70
high impedance state 583

I
iCache 55
ID 220
IDLE Mode 258
IDLE mode 253
idle states 137
Illegal opcode definition 230
images 66
IMR0 to IMR3 219
Input range 541
In-service priority register 220
Instruction Cache 155
Instruction Cache Control Register (ICC) 158
Instruction Cache Data Configuration Register (ICD) 159
Instruction Cache Hit 160
Instruction Cache Initial Register (ICI) 159
Instruction Cache Miss 161
instruction cycle 55
Internal peripheral I/O area 74

Internal RAM area 73
Internal ROM area 70
Interrupt control register 216
Interrupt mask registers 0 to 3 219
Interrupt mode 357
Interrupt mode register 0 223
Interrupt mode register 1 224
Interrupt mode register 2 225
Interrupt mode register 3 226
Interrupt response time 236
Interrupt Source Register 59
Interrupt source register 59
Interval timer 271, 290, 304, 306, 329, 337, 349
interval timer 296
INTM0 223
INTM1 224
INTM2 225
INTM3 226
INTSER0 374
INTSER1 374
INTSR0 374
INTSR1 374
INTST0 374
INTST1 374
ISPR 220

L

Least Recently Used 155
Link pointer 57
Load/store instructions with long/short format 24
LOCK0 165
low-power systems 251
LRU algorithm 155

M

malfunction 583
Maskable interrupt status flag 220
Maskable interrupts 209
    Priorities 212
Match and clear mode 335
match and clear mode 322
MODE 62, 159
MODE0 to MODE3 49
Multiple interrupt processing control 234

N

NMI 49
NMI pin input 203
NMI0 . . . . 203, 207
NMIWDT 203
No parity 369
Noise elimination 583
noise elimination 346
noise eliminator 328
noise-elimination 327
Nonlinearity Error 544
Non-maskable interrupt 203
Non-maskable interrupt status flag 208
Non-maskable watchdog timer interrupt request 203

Non-port pins 36
NP 208

O

OCTLGn 318
Odd parity 368
off-page 145
on-page 145
Operation mode 63
Oscillation stabilization time 587
oscillator stabilization time 586
Output delay operation 321
Overall Error 542
Overflow 284
Overrun error 364

P

P1 . . . . . 556, 559, 562, 564, 566, 569, 574, 575, 577, 579, 581
Page ROM configuration register 149
parity 375
parity check 365
Parity error 364
PC 56, 57
period measurement 325, 335
Peripheral area selection control register (BPC) 114
Peripheral I/O 76
peripheral I/O 74
Peripheral Status 106
PHCMD 105
PHS 106
Pin functions 33, 40
Pin I/O circuits 54
Pin Identification 28
pipeline 55
PLL synthesizer 239
PM1 . . . . . 557, 560, 563, 565, 567, 570, 574, 576, 578, 580, 582
PMC1 . . . . . 558, 561, 563, 565, 568, 571, 576, 578, 580, 582
PMW 340
Port 1 . . . . . 40, 556, 559, 562, 564, 566, 569, 572, 573, 574, 575, 577, 579
Port 2 41
Port 3 . . . . . 42, 43
Port 5 44
Port 6 . . . . . 45, 46
Port AH 47
Port CM 48
Port configuration 546
Port CS 47
Port CT 48
Port functions 545
Port pins 33
Power save control register 268
Power save mode register 270
Power save modes 239
Power saving functions 251
power-on 586
power-up 583
PRC 149
PRCMD 104
Precautions

DMA controller (DMAC) 198
Prescaler compare registers 0, 1 425
Prescaler mode registers 0, 1 424
Program counter 56, 57
program register 56
program runaway 357
Program space 67
Program Status Word 60
Program status word 60
Programmable peripheral I/O registers 83, 113
Programmable pulse generator function 271
Programmable pulse output 306
PRSCM0, PRSCM1 425
PRSM0, PRSM1 424
PSC 268
PSM 270
PSW 58, 60
pulse cycle measurement 285
Pulse interval and frequency measurement counter 306
Pulse width measurement 335
PWM 271, 291, 306, 331

Q
Quantization Error 543

R
RAM 30
Reception buffer registers 372
Reception completion interrupt 374
Reception error interrupt 374
re-initialize 583
RESET 49
RESET mode 357
Reset signal acknowledgment 585
ROM 30
ROMC 145
ROM-less 62
RXB0, RXB1 372

S
Sample and hold circuit 524
SAR 524
Saturated operation instructions 24
Saturation 61
Serial I/O shift register 406
Serial I/O shift registers Low 407
SESC0 282
Signed load instructions 24
Signed multiplication 24
SIO0 to SIO2 406
SIOL0 to SIOL2 407
SIRB0 to SIRB2 398
SIRBE0 to SIRBE2 400
SIRBEL0 to SIRBEL2 401
SIRBL0 to SIRBL2 399
Software exception 227
Software STOP mode 253, 265
SOTB0 to SOTB2 402
SOTBF0 to SOTBF2 404

SOTBFL0 to SOTBFL2 . . . 405
SOTBL0 to SOTBL2 . . . 403
specific instructions . . . 360
Specific Registers . . . 103
Spread Spectrum PLL . . . 239
SRAM . . . 137
Stack pointer . . . 57
Standby function . . . 523
standby mode . . . 541
Sub Oscillator Calibration Function . . . 293
sub oscillator calibration function . . . 271
Successive approximation . . . 523
Successive approximation register . . . 524
synchronized . . . 328
system malfunction . . . 583
system register . . . 56
System registers . . . 56, 58
system reset . . . 583

T
tag clear function 155
Text pointer . . . 57
Time base status register . . . 320
Timer C . . . 271
Timer C control register 0 . . . 278
Timer C control register 1 . . . 280
Timer D counter Register . . . 298
Timer Dn compare register . . . 299
Timer Dn control register . . . 301
Timer G capture/compare registers with external PWW-output function . . . 312
Timer Gn 16-bit counter registers . . . 310
Timer Gn capture/compare registers of the 2 counters . . . 311
Timer Gn Channel Mode Register . . . 316
Timer Gn Mode Register . . . 313, 315
Timer Gn output control register . . . 318
TMCC00 . . . 278
TMCC01 . . . 280
TMCDn . . . 301
TMDn . . . 298
TMGMn . . . 313
TMGMnH . . . 315
TMGMnL . . . 315
TMGn0 . . . 310
TMGn1 . . . 310
TMGSTn . . . 320
Transfer rate . . . 364
Transmission buffer registers . . . 373
Transmission completion interrupt . . . 374
TXB0, TXB1 . . . 373

V
Valid edge selection register . . . 282
VCMPM . . . 208
VDD3 . . . 50
VDD5 . . . 49
Voltage comparator . . . 524
Voltage comparator mode register . . . 208
VSS3 . . . 50
VSS5 . . . 50

VSWC . . . 107

W

WAIT . . . 137
Wait function . . . 109
wait states . . . 107
WATCH mode . . . 253, 259, 261
Watch timer . . . 349
    Operation . . . 352
Watch Timer Clock . . . 352
Watch timer clock frequency . . . 355
Watch timer mode control register . . . 350
watchdog timer . . . 587
Watchdog timer clock selection register . . . 359
Watchdog timer command register . . . 361
Watchdog timer command status register . . . 361
Watchdog Timer Control Register . . . 359
Watchdog timer mode register . . . 360
WCMD . . . 361
WDCS . . . 359
WDTM . . . 360
Word access (32 bits) . . . 123
WPHS . . . 361
Wrap-around . . . 67
WTM . . . 350

X

X1 . . . 49
X2 . . . 49

Z

Zero register . . . 57
Zero-scale error . . . 543

NEC

Facsimile Message

From:

Name

Company

Tel. FAX

Address

Although NEC has taken all possible steps to ensure that the documentation supplied to our customers is complete, bug free and up-to-date, we readily accept that errors may occur. Despite all the care and precautions we've taken, you may encounter problems in the documentation. Please complete this form whenever you'd like to report errors or suggest improvements to us.

Thank you for your kind support.

North America
NEC Electronics Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49-211-6503-274

South America
NEC do Brasil S.A.
Fax: +55-11-6465-6829

Hong Kong, Philippines, Oceania
NEC Electronics Hong Kong Ltd.
Fax: +852-2886-9022/9044

Korea
NEC Electronics Hong Kong Ltd.
Seoul Branch
Fax: 02-528-4411

Taiwan
NEC Electronics Taiwan Ltd.
Fax: 02-2719-5951

Asian Nations except Philippines
NEC Electronics Singapore Pte. Ltd.
Fax: +65-6250-3583

Japan
NEC Semiconductor Technical Hotline
Fax: +81- 44-435-9608

I would like to report the following error/make the following suggestion:

Document title: ______________________________

Document number: ______________________ Page number: ____________

____

____

____

If possible, please fax the referenced page or drawing.

Document Rating	Excellent	Good	Acceptable	Poor
Clarity	☐	☐	☐	☐
Technical Accuracy	☐	☐	☐	☐
Organization	☐	☐	☐	☐