Preliminary User's Manual

V850E/CA4^{TM} HELIOS

32-bit RISC Microcontroller

Hardware

µPD703174, µPD70F3175, µPD703175,
µPD703176

Document No. U16241EE1V1UM00
Date Published January 2005

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 America Inc.
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 1101
Fax: 0211-65 03 1327

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

Preface

Readers This manual is intented for users who want to understand the functions of the V850E/CA4 (nickname HELIOS).

Purpose This manual presents the hardware manual of V850E/CA4.

Organization This system specification describes the following sections:

- Pin function
- CPU function
- Internal peripheral function
- Flash memory

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. . . . 21

1.1 General . . . 21

1.2 Device Features . . . 22

1.3 Application Fields . . . 23

1.4 Ordering Information . . . 23

1.5 Pin Configuration (Top View) . . . 24

1.6 Configuration of Function Block . . . 26

1.6.1 Block Diagram of V850E/CA4. . . . 26

1.6.2 On-chip units. . . . 27

Chapter 2 Pin Functions . . . 29

2.1 List of Pin Functions . . . 29

2.2 Description of Pin Functions . . . 38

Chapter 3 CPU Functions . . . 53

3.1 Features . . . 53

3.2 CPU Register Set . . . 54

3.2.1 Program register set . . . 55

3.2.2 System register set . . . 56

3.2.3 Special registers . . . 62

3.3 Operation Modes . . . 65

3.4 Address Space . . . 66

3.4.1 CPU address space . . . 66

3.4.2 Image . . . 67

3.4.3 Wrap-around of CPU address space . . . 68

3.4.4 Memory map. . . . 69

3.4.5 Areas . . . 72

3.4.6 Area access time . . . 75

3.4.7 Peripheral I/O registers. . . . 76

3.4.8 FCAN RAM and registers . . . 83

Chapter 4 DMA Functions (DMA Controller) . . . 89

4.1 Features . . . 89

4.2 Control Registers. . . . 90

4.2.1 DMA source address registers high-word (DSA0H to DSA3H) . . . 90

4.2.2 DMA destination address registers high-word (DDA0H to DDA3H) . . . 92

4.2.3 DMA transfer count registers 0 to 3 (DBC0 to DBC3) . . . 94

4.2.4 DMA addressing control registers 0 to 3 (DADC0 to DADC3) . . . 95

4.2.5 DMA channel control registers 0 to 3 (DCHC0 to DCHC3) . . . 96

4.2.6 DMA disable status register (DDIS) . . . 97

4.2.7 DMA restart register (DRST) . . . 97

4.2.8 DMA trigger factor register 0 (DTFR0) . . . 98

4.2.9 DMA trigger factor register 1 (DTFR1) . . . 99

4.2.10 DMA trigger factor register 2 (DTFR2) . . . 100

4.2.11 DMA trigger factor register 3 (DTFR3) . . . 101

4.3 DMA Bus States . . . 102

4.3.1 Types of bus states. . . . 102

4.3.2 DMAC bus cycle state transition. . . . 103

4.4 Transfer Mode . . . 104

4.4.1 Single transfer mode. . . . 104

4.5 Transfer Types . . . 104

4.5.1 Two-cycle transfer . . . 104

4.6 Transfer Object . . . 104

4.6.1 Transfer type and transfer object 104
4.7 DMA Channel Priorities 105
4.8 Next Address Setting Function. 105
4.9 DMA Transfer Start Factors 106
4.10 Forcible Interruption 106
4.11 DMA Transfer End 107
4.11.1 DMA transfer end interrupt 107
4.11.2 Terminal count output upon DMA transfer end. 107
4.12 Forcible Termination 107
4.13 Precautions. 107

Chapter 5 Interrupt/Exception Processing Function 109
5.1 Outline. 109
5.1.1 Features 109
5.2 Non-Maskable Interrupt 112
5.2.1 NP flag 112
5.2.2 Watchdog Timer NMI Operation 113
5.2.3 External NMI Operation 115
5.2.4 Watchdog or external NMI Restore 117
5.3 Maskable Interrupts 118
5.3.1 Operation 118
5.3.2 Restore 120
5.3.3 Priorities of maskable interrupts 121
5.3.4 Interrupt control register (xxnIC) 125
5.3.5 Interrupt Mask Registers (IMRn). 126
5.3.6 In-service priority register (ISPR) 130
5.3.7 Global interrupt mask: ID (Interrupt Disable) 131
5.3.8 Edge detection function 132
5.4 Software Exception 133
5.4.1 Operation 133
5.4.2 Restore 134
5.4.3 EP flag 135
5.5 Exception Trap 136
5.5.1 Illegal op code definition 136
5.5.2 Operation 137
5.5.3 Restore 138
5.6 Priority Control. 139
5.6.1 Priorities of interrupts and exceptions. 139
5.6.2 Multiple interrupt processing. 140
5.7 Interrupt Latency Time 142
5.8 Periods Where Interrupt Is Not Acknowledged. 143

Chapter 6 Clock Generator 145
6.1 Features 145
6.2 Configuration 146
6.3 Clock Oscillator 147
6.4 Control Registers. 148
6.4.1 Clock control register CKC 149
6.4.2 PLL divider factor register PDIV 150
6.4.3 PCL and peripheral clock control register PCLCNT 151
6.4.4 PLL status register PSTAT 153
6.4.5 Power save mode register PSM 154
6.4.6 PSC - power save control register 155
6.5 System Operating Modes 156
6.5.1 Main clock operation mode. 157
6.5.2 HALT mode. 157
6.5.3 IDLE mode 159
6.5.4 WATCH mode. 161

6.5.5 STOP mode 165
6.5.6 Securing Oscillation Stabilization Time. 169

Chapter 7 Timer 171
7.1 Timer D 171
7.1.1 Features Timer D 171
7.1.2 Function overview Timer Dn. 171
7.1.3 Basic configuration 173
7.1.4 Control register 176
7.1.5 Operation 177
7.1.6 Application example 179
7.1.7 Precautions for Timer Dn 179
7.2 Timer G 180
7.2.1 Features of Timer G 180
7.2.2 Function Overview of each Timer Gn 181
7.2.3 Basic configuration 184
7.2.4 Control Registers 188
7.2.5 Output delay operation 194
7.2.6 Explanation of Basic Operation 195
7.2.7 Operation in Free-run mode 197
7.2.8 Match and clear mode 206
7.2.9 Edge noise elimination 215
7.2.10 Precautions Timer Gn. 216
7.3 Timer Input Select 218
7.3.1 UART6n connection to Timer Inputs. 218
7.3.2 Timer input select control register. 219

Chapter 8 Watch Timer 221
8.1 Function 221
8.2 Configuration 222
8.3 Watch Timer Control Register 222
8.4 Operations 224
8.4.1 Operation as watch timer 224
8.4.2 Operation as interval timer 224
8.4.3 Watch timer and Interval timer simultaneously. 224

Chapter 9 Watchdog Timer 225
9.1 Features 225
9.2 Configuration 227
9.2.1 Watchdog Timer control registers. 227
9.3 Operation 229
9.3.1 Watchdog timer generating a reset. 229
9.3.2 Watchdog timer as interval timer 230

Chapter 10 Serial Interface Function 231
10.1 Features 231
10.2 Serial Interface UART6n 232
10.2.1 Features 232
10.2.2 Functions of serial interfaces UART6n (UART60, UART61) 233
10.2.3 Configuration of Serial Interface UART6n 236
10.2.4 Registers Controlling Serial Interface UART6n 239
10.2.5 Operation of serial interface UART6n. 247
10.2.6 Dedicated baud rate generator. 267
10.3 Clocked Serial Interfaces 0, 1 (CSI00, CSI01) 274
10.3.1 Features 274
10.3.2 Configuration 275
10.3.3 Control registers 277
10.3.4 Operation 288

10.3.5 Output pins 301
10.4 Clocked Serial Interface 10 (CSI10) 302
10.4.1 Features 302
10.4.2 Configuration 303
10.4.3 Control registers 305
10.4.4 Operation 310
10.4.5 Output pins 315
10.5 Baud rate generator BRGC0 316

Chapter 11 FCAN Interface Function 319
11.1 Features 319
11.2 Outline of the FCAN System 320
11.2.1 General 320
11.2.2 CAN memory and register layout 321
11.2.3 Clock structure 329
11.2.4 Interrupt handling 330
11.2.5 Time stamp 331
11.2.6 Message handling 333
11.2.7 Mask handling 338
11.2.8 Remote frame handling 339
11.2.9 Outer priority inversion 342
11.3 Control and Data Registers 343
11.3.1 Bit set/clear function 343
11.3.2 Common registers 345
11.3.3 CAN interrupt pending registers 357
11.3.4 CAN message buffer registers 362
11.3.5 CAN Module Registers 375
11.4 Operating Considerations 400
11.4.1 Rules to be observed for correct baud rate settings 400
11.4.2 Example for baudrate setting of CAN module 401
11.4.3 Ensuring data consistency 403
11.4.4 Operating states of the CAN modules 405
11.4.5 Initialisation routines 406

Chapter 12 A/D Converter 413
12.1 Features 413
12.2 Configuration 414
12.3 Control Registers 417
12.4 Interrupt Requests 426
12.5 A/D Converter Operation 427
12.5.1 A/D converter basic operation 427
12.5.2 Operation modes and trigger modes 428
12.6 Operation in A/D Trigger Mode 431
12.6.1 Operation in select mode 431
12.6.2 Operation in scan mode 432
12.7 Operation in A/D Trigger Polling Mode 433
12.7.1 Operation in select mode 433
12.7.2 Operation in scan mode 434
12.8 Precautions. 435
12.8.1 Stopping conversion operation 435
12.8.2 Operation in standby modes 435

Chapter 13 Port Feature 437
13.1 Compositions of the port 437
13.2 Port 0 438
13.2.1 Port 0 register (P0) 438
13.2.2 Port 0 mode register (PM0) 439
13.2.3 Port 0 mode control register (PMC0) 440

13.3 Port 1 441
13.3.1 Port 1 register (P1) 441
13.3.2 Port 1 mode register (PM1) 442
13.3.3 Port 1 mode control register (PMC1) 443
13.4 Port 2 444
13.4.1 Port 2 register (P2) 444
13.4.2 Port 2 mode register (PM2) 445
13.4.3 Port 2 mode control register (PMC2) 446
13.5 Port 3 447
13.5.1 Port 3 register (P3) 447
13.5.2 Port 3 mode register (PM3) 448
13.5.3 Port 3 mode control register (PMC3) 449
13.6 Port 4 450
13.6.1 Port 4 register (P4) 450
13.6.2 Port 4 mode register (PM4) 451
13.6.3 Port 4 mode control register (PMC4) 452
13.7 Port 7 453
13.7.1 Port 7 register (P7) 453
13.7.2 Port 7 mode control register (PMC7) 454
13.8 Port DL 455
13.8.1 Port DL register (PDL) 455
13.8.2 Port DL mode register (PMDL) 456
13.9 Port DH 457
13.9.1 Port DH register (PDH) 457
13.9.2 Port DH mode register (PMDH) 458
13.10 Port CT 459
13.10.1 Port CT register (PCT) 459
13.10.2 Port CT mode register (PMCT) 459
13.11 Port CM 460
13.11.1 Port CM register (PCM) 460
13.11.2 Port CM mode register (PMCM) 460
13.12 Port CD 461
13.12.1 Port CD register (PCD) 461
13.12.2 Port CD mode register (PMCD) 461
13.13 Port Type 462
13.14 Port Block Type 465
13.14.1 Port Block Type A 465
13.14.2 Port Block Type B 466
13.14.3 Port Block Type C 467
13.14.4 Port Block Type D 468
13.14.5 Port Block Type E 469
13.14.6 Port Block Type F 470
13.14.7 Port Block Type G 471
13.14.8 Port Block Type H 472
13.14.9 Port Block Type I 473
13.15 Noise Elimination Filter and Edge detection 474
13.15.1 Filter and edge detection structure 474
13.15.2 Filter configuration 476
13.15.3 Register for interrupt edge detection 477

Chapter 14 RESET Function 479
14.1 Overview 479
14.2 Configuration 479
14.3 Operation 480
14.4 RESET cause identification 482

Chapter 15 Flash Memory (µPD70F3175 only) 483
15.1 Features 483

15.2 Writing by Flash writer 484
15.3 Programming Environment. 484
15.4 Communication System 485
15.5 Flash Programming Circuitry 487
15.6 Pin Handling 488
15.6.1 V_{PP} pin 488
15.6.2 Serial interface pins 489
15.6.3 $\overline{\text{RESET}}$ pin 491
15.6.4 NMI pin 491
15.6.5 Flash memory programming mode. 491
15.6.6 Port pins 491
15.6.7 Other signal pins. 491
15.6.8 Power supply 491
15.7 Programming Method 492
15.7.1 Flash memory control 492
15.7.2 Selection of communication mode 492
15.8 Selfprogramming Mode. 493
15.9 Secure Selfprogramming 494
15.9.1 General description. 494
15.9.2 Signature structure 494
15.9.3 Secure Selfprogramming flow. 494
15.9.4 Advantages of Secure Selfprogramming 496

Appendix A Instruction Set List. 497

Appendix B Index 505

List of Figures

Figure 1-1: Pin Configuration of the µPD70F317x ... 24
Figure 1-2: Block Diagram of the V850E/CA4 Microcontroller ... 26
Figure 2-1: Power Supply Connection ... 33
Figure 2-2: Pin I/O Circuits ... 51
Figure 3-1: CPU Register Set ... 54
Figure 3-2: Program Counter (PC) Format ... 55
Figure 3-3: Interrupt Context Saving Registers (EIPC, EIPSW) Format ... 57
Figure 3-4: NMI Context Saving Registers (FEPC, FEPSW) Format ... 58
Figure 3-5: Interrupt Source Register (ECR) Format ... 58
Figure 3-6: Program Status Word (PSW) Format ... 59
Figure 3-7: CALLT Execution Context Saving Registers (CTPC, CTPSW) Format ... 60
Figure 3-8: Exception/Debug Trap Context Saving Registers (DBPC, DBPSW) Format ... 61
Figure 3-9: CALLT Base Pointer (CTBP) Format ... 61
Figure 3-10: Processor Command Register Format ... 62
Figure 3-11: System Status Register Format ... 63
Figure 3-12: CPU Address Space ... 66
Figure 3-13: Address Space Image ... 67
Figure 3-14: Program Space ... 68
Figure 3-15: Data Space ... 68
Figure 3-16: Memory Map for 256 KB Flash Version (µPD70F3175, µPD703175) ... 69
Figure 3-17: Memory Map for 192 KB ROM Version (µPD703176) ... 70
Figure 3-18: Memory Map for 128 KB ROM Version (µPD703174) ... 71
Figure 3-19: Programmable Peripheral Area Control Register BPC ... 73
Figure 3-20: Programmable Peripheral Area ... 74
Figure 3-21: System Wait Control Register (VSWC) Format ... 75
Figure 4-1: DMA Source Address Registers DSA0H to DSA3H ... 90
Figure 4-2: DMA Source Address Registers DSA0L to DSA3L ... 91
Figure 4-3: DMA Destination Address Registers high-word (DDA0H to DDA3H) ... 92
Figure 4-4: DMA Destination Address Registers low-word (DDA0L to DDA3L) ... 93
Figure 4-5: DMA Transfer Count Registers 0 to 3 (DBC0 to DBC3) ... 94
Figure 4-6: DMA Addressing Control Registers 0 to 3 (DADC0 to DADC3) ... 95
Figure 4-7: DMA Channel Control Registers 0 to 3 (DCHC0 to DCHC3) ... 96
Figure 4-8: DMA Disable Status Register (DDIS) ... 97
Figure 4-9: DMA Restart Register (DRST) ... 97
Figure 4-10: DMA Trigger Factor Registers 0 (DTFR0) ... 98
Figure 4-11: DMA Trigger Factor Registers 1 (DTFR1) ... 99
Figure 4-12: DMA Trigger Factor Registers 2 (DTFR2) ... 100
Figure 4-13: DMA Trigger Factor Registers 3 (DTFR3) ... 101
Figure 4-14: DMAC Bus Cycle (Two-Cycle Transfer) State Transition ... 103
Figure 4-15: Buffer Register Configuration ... 105
Figure 4-16: Example of Forcible Interruption of DMA Transfer ... 106
Figure 5-1: NP Flag (NP) ... 112
Figure 5-2: Watchdog Timer Non-Maskable Interrupt Servicing ... 113
Figure 5-3: Acknowledging Watchdog Non-Maskable Interrupt Request ... 114
Figure 5-4: External Non-Maskable Interrupt Servicing ... 115
Figure 5-5: NMI Edge Specification: NMIM Register Format ... 116
Figure 5-6: RETI Instruction Processing ... 117
Figure 5-7: Maskable Interrupt Servicing ... 119
Figure 5-8: RETI Instruction Processing ... 120
Figure 5-9: Example of Interrupt Nesting Service (1/2) ... 122
Figure 5-10: Example of Servicing Interrupt Requests Simultaneously Generated ... 124
Figure 5-11: Interrupt Control Register (xxnIC) Format ... 125
Figure 5-12: Interrupt Mask Register Format IMR0 ... 126
Figure 5-13: Interrupt Mask Register Format IMR1 ... 127
Figure 5-14: Interrupt Mask Register Format IMR2 ... 128

Figure 5-15: Interrupt Mask Register Format IMR3 129
Figure 5-16: In-Service Priority Register (ISPR) Format.......... 130
Figure 5-17: Interrupt Disable Flag (ID) 131
Figure 5-18: External Interrupts Edge Selection Register INTM0.......... 132
Figure 5-19: External Interrupts Edge Selection Register INTM1.......... 132
Figure 5-20: External Interrupts Edge Selection Register INTM2.......... 132
Figure 5-21: External Interrupts Edge Selection Register INTM3.......... 132
Figure 5-22: Software Exception Processing.......... 133
Figure 5-23: RETI Instruction Processing 134
Figure 5-24: EP Flag (EP).......... 135
Figure 5-25: Illegal op Code.......... 136
Figure 5-26: Exception Trap Processing.......... 137
Figure 5-27: RETI Instruction Processing 138
Figure 5-28: Pipeline Operation at Interrupt Request Acknowledgement.......... 142
Figure 6-1: Clock Generator Configuration 146
Figure 6-2: Main System Clock Oscillator.......... 147
Figure 6-3: Clock Control Register CKC 149
Figure 6-4: PLL Divider Factor Register PDIV 150
Figure 6-5: PCL and Peripheral Clock Control Register PCLCNT (1/2) 151
Figure 6-6: PLL Status Register PSTAT 153
Figure 6-7: Power Save Mode Register Format 154
Figure 6-8: Power Save Control Register Format.......... 155
Figure 6-9: System Operating and Standby Modes Block Diagram 156
Figure 6-10: Watch Mode Release by NMI or INT.......... 163
Figure 6-11: Watch Mode Release by Reset (External or Watchdog) 164
Figure 6-12: Stop Mode Release by NMI or INT.......... 167
Figure 6-13: Stop Mode Release by Reset (External or Watchdog).......... 168
Figure 7-1: Block Diagram of Timer Dn (n = 0, 1).......... 172
Figure 7-2: Timer Dn Counter Register (TMDn) (n = 0, 1) 173
Figure 7-3: Timer Dn Compare Register (CMDn) (n = 0, 1) 174
Figure 7-4: Timing of Timer Dn Operation 175
Figure 7-5: Timer Dn Control Register (TMCDn) (n = 0, 1) 176
Figure 7-6: Timing of Compare Operation (1/2).......... 177
Figure 7-7: Writing to CMDn Register.......... 178
Figure 7-8: Block Diagram of Timer G0 182
Figure 7-9: Block Diagram of Timer G1 183
Figure 7-10: Timer Gn Counter 0 Value Registers TMGn0 185
Figure 7-11: Timer Gn Counter 1 Value Registers TMGn1 185
Figure 7-12: Timer Gn Counter TMGn0 Assigned Capture/Compare Register (GCCn0).......... 186
Figure 7-13: Timer Gn Counter TMGn1 Assigned Capture/Compare Register (GCCn5).......... 186
Figure 7-14: Timer Gn Free Assignable Capture/Compare Registers (GCCnm) (m = 1 to 4)....... 187
Figure 7-15: Timer Gn Mode Register (TMGMn) (1/2) 188
Figure 7-16: Timer Gn Channel Mode Register (TMGCMn).......... 190
Figure 7-17: Timer Gn Output Control Register OCTLGn (1/2).......... 191
Figure 7-18: Timer Gn Status Register (TMGSTn).......... 193
Figure 7-19: Timing of Output Delay Operation.......... 194
Figure 7-20: Timing when both Edges of TIGn0 are Valid (free run).......... 199
Figure 7-21: Timing of Capture Trigger Edge Detection (free run) 200
Figure 7-22: Timing of Starting Capture Trigger Edge Detection.......... 200
Figure 7-23: Timing of Compare Mode (free run) 201
Figure 7-24: Timing when GCCn1 is Rewritten During Operation (free run) 202
Figure 7-25: Timing of PWM Operation (free run) 204
Figure 7-26: Timing when 0000H is set in GCCnm (free run).......... 204
Figure 7-27: Timing when FFFFH is set in GCCnm (free run).......... 205
Figure 7-28: Timing when GCCnm is Rewritten During Operation (free run) 205
Figure 7-29: Timing when both Edges of TIGnm are Valid (Match and Clear).......... 207
Figure 7-30: Timing of Compare Operation (Match and Clear) 209
Figure 7-31: Timing when GCCnm is Rewritten During Operation (Match and Clear).......... 210

Figure 7-32: Timing of PWM Operation (Match and Clear) 212
Figure 7-33: Timing when 0000H is Set in GCCnm (Match and Clear) 212
Figure 7-34: Timing when the same Value as set in GCCn0/GCCn5 is set in GCCnm (match and clear) 213
Figure 7-35: Timing when the Value of GCCnm Exceeding GCCn0 or GCCn5 (match and clear) 214
Figure 7-36: Timing when GCCnm is Rewritten During Operation (Match and Clear) 215
Figure 7-37: Timer Input Select Block Diagram 218
Figure 7-38: Timer Input Select Control Register Format 219
Figure 8-1: Block Diagram of Watch Timer 221
Figure 8-2: Watch Timer Mode Control Register (WTM) (1/2) 222
Figure 8-3: Example Watch Timer and Interval Timer Simultaneously 224
Figure 9-1: Block Diagram of Watchdog Timer 226
Figure 9-2: Watchdog Timer Mode Register (WDTM) Format 227
Figure 9-3: Watchdog Timer Enable Register (WDTE) Format 229
Figure 9-4: Watchdog Operation at Timer Overflow 230
Figure 10-1: LIN Transmission Operation 233
Figure 10-2: LIN Reception Operation 234
Figure 10-3: Port Configuration for LIN Reception Operation 235
Figure 10-4: UART60, UART61 Block Diagram 236
Figure 10-5: Format of Receive Buffer Register (RXBn) 237
Figure 10-6: Format of Transmit Buffer Register (TXBn) 238
Figure 10-7: Format of Asynchronous Serial Interface Operation Mode Register (ASIMn) (1/2) .. 239
Figure 10-8: Format of Asynchronous Serial Interface Reception Error Status Register (ASISn). 241
Figure 10-9: Format of Asynchronous Serial Interface Transmission Status Register (ASIFn)..... 242
Figure 10-10: Format of Clock Selection Register (CKSRn) 243
Figure 10-11: Format of Dedicated Baud Rate Generator Control Register (BRGCn) 244
Figure 10-12: Format of Asynchronous Serial Interface Control Register (ASICLn) (1/2) 245
Figure 10-13: Register ASIMn in Operation STOP Mode (1/2) 247
Figure 10-14: Register ASIMn in Asynchronous Serial Interface (UART) Mode (1/2) 249
Figure 10-15: Register ASISn in Asynchronous Serial Interface (UART) Mode 251
Figure 10-16: Register ASIFn in Asynchronous Serial Interface (UART) Mode 252
Figure 10-17: Register ASICLn in Asynchronous Serial Interface (UART) Mode (1/2) 253
Figure 10-18: Format of Normal UART Transmitted/Received Data 255
Figure 10-19: Example of Normal UART Transmitted/Received Data Format 256
Figure 10-20: Normal Transmission Completion Interrupt Request Timing 258
Figure 10-21: Processing Flow of Continuous Transmission 260
Figure 10-22: Timing of Starting Continuous Transmission 261
Figure 10-23: Timing of Ending Continuous Transmission 262
Figure 10-24: Reception Completion Interrupt Request Timing 263
Figure 10-25: Reception Error Interrupt 264
Figure 10-26: Noise Filter Circuit 265
Figure 10-27: SBF Transmission 265
Figure 10-28: SBF Reception 266
Figure 10-29: Configuration of Dedicated Baud Rate Generator 268
Figure 10-30: Clock Selection Register (CKSRn) Format 269
Figure 10-31: Dedicated Baud Rate Generator Control Register (BRGCn) Format 270
Figure 10-32: Transfer Rate During Continuous Transmission 273
Figure 10-33: Block Diagram of Clocked Serial Interfaces 276
Figure 10-34: Clocked Serial Interface Mode Registers (CSIM0, CSIM1) 277
Figure 10-35: Clocked Serial Interface Clock Selection Registers (CSIC0, CSIC1) (1/2) 278
Figure 10-36: Clocked Serial Interface Reception Buffer Registers (SIRB0, SIRB1) 280
Figure 10-37: Clocked Serial Interface Reception Buffer Registers (SIRBL0, SIRBL1) 281
Figure 10-38: Clocked Serial Interface Transmission Buffer Registers (SOTB0, SOTB1) 282
Figure 10-39: Clocked Serial Interface Transmission Buffer Registers (SOTBnL) 283
Figure 10-40: Clocked Serial Interface Initial Transmission Buffer Registers 0, 1 (SOTBFn) 284
Figure 10-41: Clocked Serial Interface Initial Transmission Buffer Registers (SOTBFnL) 285
Figure 10-42: Serial I/O Shift Registers (SIO0, SIO1) 286
Figure 10-43: Serial I/O Shift Registers (SIO0L, SIO1L) 287

Figure 10-44: Timing Chart in Single Transfer Mode (DAPn = 0) (1/2) 289
Figure 10-45: Timing Chart According to Clock Phase Selection (1/2).......... 291
Figure 10-46: Timing Chart of Interrupt Request Signal Output in Delay Mode (1/2).......... 293
Figure 10-47: Repeat Transfer (Receive-Only) Timing Chart.......... 295
Figure 10-48: Repeat Transfer (Transmission/Reception) Timing Chart.......... 297
Figure 10-49: Timing Chart of Next Transfer Reservation Period (1/2) 298
Figure 10-50: Transfer Request Clear and Register Access Contention.......... 299
Figure 10-51: Interrupt Request and Register Access Contention 300
Figure 10-52: Block Diagram of Clocked Serial Interface CSI10.......... 304
Figure 10-53: Clocked Serial Interface Mode Registers (CSIM10) 305
Figure 10-54: Clocked Serial Interface Clock Selection Register (CSIC10) (1/2) 306
Figure 10-55: Clocked Serial Interface Transmission Buffer Register (SOTB10) 308
Figure 10-56: Serial I/O Shift Register (SIO10) 309
Figure 10-57: Timing Chart in Single Transfer Mode (DAP10 = 0) (1/2) 311
Figure 10-58: Timing Chart According to Clock Phase Selection (1/2).......... 313
Figure 10-59: Baud Rate Generator 0 (BRG0) Block Diagram.......... 316
Figure 10-60: Prescaler Mode Register (PRSM0) Format.......... 317
Figure 10-61: Prescaler Compare Register (PRSCM0) Format 318
Figure 11-1: Functional Blocks of the FCAN Interface.......... 320
Figure 11-2: Memory Area of the FCAN System 321
Figure 11-3: Clock Structure of the FCAN System 329
Figure 11-4: FCAN Interrupt Bundling of V850E/CA4.......... 330
Figure 11-5: Time Stamp Capturing at Message Reception.......... 331
Figure 11-6: Time Stamp Capturing at Message Transmission.......... 332
Figure 11-7: 16-Bit Data Write Operation for Specific Registers 344
Figure 11-8: CAN Stop Register (CSTOP) 345
Figure 11-9: CAN Main Clock Select Register (CGSC) (1/2).......... 346
Figure 11-10: Configuration of FCAN System Main Clock.......... 347
Figure 11-11: Configuration of FCAN Global Time System Clock.......... 347
Figure 11-12: CAN Global Status Register (CGST) (1/3) 348
Figure 11-13: CAN Global Interrupt Enable Register (CGIE) (1/2) 351
Figure 11-14: CAN Global Time System Counter (CGTSC) 353
Figure 11-15: CAN Message Search Start Register (CGMSS) 354
Figure 11-16: CAN Message Search Start Register (CGMSS) 355
Figure 11-17: CAN Test Bus Register (CTBR) 356
Figure 11-18: Internal CAN Test Bus Structure.......... 356
Figure 11-19: CAN Interrupt Pending Registers (CCINTPL, CCINTPH) 357
Figure 11-20: CAN Global Interrupt Pending Register (CGINTP) (1/2) 358
Figure 11-21: CAN 1 to 4 Interrupt Pending Registers (C1INTP to C4INTP) (1/2) 360
Figure 11-22: Message Identifier Registers L00 to L31 and H00 to H31 (M_IDL00 to M_IDL31, M_IDH00 to M_IDH31) 362
Figure 11-23: Message Configuration Registers 00 to 31 (M_CONF00 to M_CONF31) (1/2) 363
Figure 11-24: Message Status Registers 00 to 31 (M_STAT00 to M_STAT31) 365
Figure 11-25: Message Set/Clear Status Registers 00 to 31 (SC_STAT00 to SC_STAT31) 367
Figure 11-26: Message Data Registers m0 to m7 (M_DATAm0 to M_DATAm7) (m = 00 to 31) (1/2) 368
Figure 11-27: Message Data Length Code Registers 00 to 31 (M_DLC00 to M_DLC31) 370
Figure 11-28: Message Control Registers 00 to 31 (M_CTRL00 to M_CTRL31) (1/2) 371
Figure 11-29: Message Time Stamp Registers 00 to 31 (M_TIME00 to M_TIME31).......... 373
Figure 11-30: CAN 1 to 2 Mask 0 to 3 Registers L, H (CxMASKL0 to CxMASKL3, CxMASKH0 to CxMASKH3) (x = 1 to 4) 375
Figure 11-31: CAN 1 to 2 Control Registers (C1CTRL to C2CTRL) (1/5) 377
Figure 11-32: CAN 1 to 2 Definition Registers (C1DEF to C2DEF) (1/4) 382
Figure 11-33: CAN 1 to 4 Information Registers (C1LAST to C4LAST) 386
Figure 11-34: CAN 1 to 2 Error Counter Registers (C1ERC to C2ERC) 387
Figure 11-35: CAN 1 to 2 Interrupt Enable Registers (C1IE to C2IE) (1/4) 388
Figure 11-36: CAN 1 to 2 Bus Activity Registers (C1BA to C2BA) (1/2) 392
Figure 11-37: CAN 1 to 4 Bit Rate Prescaler Registers (C1BRP to C4BRP) (1/2) 394

Figure 11-38: CAN Bus Bit Timing 396
Figure 11-39: CAN 1 to 2 Synchronization Control Registers (C1SYNC to C2SYNC) (1/2)) 397
Figure 11-40: CAN 1 to 4 Bus Diagnostic Information Registers (C1DINF to C4DINF) 399
Figure 11-41: Sequential CAN Data Read by CPU 403
Figure 11-42: State Transition Diagram for CAN Modules 405
Figure 11-43: General Initialisation Sequence for the CAN Interface 406
Figure 11-44: Initialisation Sequence for a CAN module 408
Figure 11-45: Setting CAN Module into Initialisation State 410
Figure 12-1: Block Diagram of A/D Converter 416
Figure 12-2: A/D Scan Mode Register 0 (ADSCM0) (1/2) 417
Figure 12-3: A/D Scan Mode Register 1 (ADSCM1) (1/2) 419
Figure 12-4: A/D Voltage Detection Mode Register (ADETM) 422
Figure 12-5: A/D Conversion Result Registers 0 to 13 (ADCR0 to ADCR13) 423
Figure 12-6: Relationship Between Analog Input Voltages and A/D Conversion Results 425
Figure 12-7: Example of Select Mode Operation Timing (ANI1) 429
Figure 12-8: Example of Scan Mode Operation Timing (4-Channel Scan (ANI0 to ANI3)) 430
Figure 12-9: Example of Select Mode (A/D Trigger Select) Operation (ANI2) 431
Figure 12-10: Example of Scan Mode (A/D Trigger Scan) Operation (ANI2-ANI5) 432
Figure 12-11: Example of Select Mode (A/D Trigger Polling Select) Operation (ANI2) 433
Figure 12-12: Example of Scan Mode (A/D Trigger Polling Scan) Operation (ANI2 to ANI5) 434
Figure 13-1: Ports Block Diagram 437
Figure 13-2: Port 0 Register (P0) Format 438
Figure 13-3: Port Mode Register (PM0) Format 439
Figure 13-4: Port 0 Mode Control Register (PMC0) Format 440
Figure 13-5: Port 1 Register (P1) Format 441
Figure 13-6: Port Mode Register (PM1) Format 442
Figure 13-7: Port Mode Control Register (PMC1) Format 443
Figure 13-8: Port 2 Register (P2) Format 444
Figure 13-9: Port Mode Register (PM2) Format 445
Figure 13-10: Port Mode Control Register (PMC2) Format 446
Figure 13-11: Port 3 Register (P3) Format 447
Figure 13-12: Port Mode Register (PM3) Format 448
Figure 13-13: Port 3 Mode Control Register (PMC3) Format 449
Figure 13-14: Port 4 Register (P4) Format 450
Figure 13-15: Port Mode Register (PM3) Format 451
Figure 13-16: Port 4 Mode Control Register (PMC4) Format 452
Figure 13-17: Port 7 Register (P7) Format 453
Figure 13-18: Port 7 Mode Control Register (PMC7) Format 454
Figure 13-19: Port DL Register (PDL) Format 455
Figure 13-20: Port DL Mode Register (PMDL) Format 456
Figure 13-21: Port DH Register (PDH) Format 457
Figure 13-22: Port DH Mode Register (PMDH) Format 458
Figure 13-23: Port CT Register (PCT) Format 459
Figure 13-24: Port CT Mode Register (PMCT) Format 459
Figure 13-25: Port CM Register (PCM) Format 460
Figure 13-26: Port CM Mode Register (PMCM) Format 460
Figure 13-27: Port CD Register (PCD) Format 461
Figure 13-28: Port CD Mode Register (PMCD) Format 461
Figure 13-29: Type A Block Diagram 465
Figure 13-30: Type B Block Diagram 466
Figure 13-31: Type C Block Diagram 467
Figure 13-32: Type D Block Diagram 468
Figure 13-33: Type E Block Diagram 469
Figure 13-34: Type F Block Diagram 470
Figure 13-35: Type G Block Diagram 471
Figure 13-36: Type H Block Diagram 472
Figure 13-37: Type I Block Diagram 473
Figure 13-38: Type A: Input Filter and Edge Detection Overview for NMI 474

Figure 13-39: Type B: Input Filter and Edge Detection Overview for P0, P1, P2, P3 474
Figure 13-40: Edge Detect Circuit for INTPn / NMI 475
Figure 13-41: INTM Register for Interrupt Edge Detection Format 477
Figure 13-42: NMIM Register for Interrupt Edge Detection Format 478
Figure 14-1: Reset Block Diagram 479
Figure 14-2: Hardware Status on $\overline{\text{RESET}}$ Input or Watchdog Request 481
Figure 14-3: Operation on Power Application 481
Figure 14-4: Reset Cause Identification Register RSM 482
Figure 15-1: Programming Environment in Conjunction with External Flash Writer 484
Figure 15-2: Flash Writer Communication via CSI00 485
Figure 15-3: Flash Writer Communication via UART60 486
Figure 15-4: Minimal Circuitry for Flash Selfprogramming 487
Figure 15-5: Pin Handling of V_{PP} pin 488
Figure 15-6: Conflict between Flash Writer and Other Output Pin 489
Figure 15-7: Malfunction of Other Input Pins 490
Figure 15-8: Conflict between Flash Writer Reset Line and Reset Signal Generation Circuit 491
Figure 15-9: Flow Chart of Flash Memory Manipulation 492
Figure 15-10: Configuration in Selfprogramming Mode 493
Figure 15-11: Secure Selfprogramming Flow (1/2) 495

List of Tables

Table 1-1: Product Versions 22
Table 2-1: Port Pins 29
Table 2-2: Non-Port Pins 32
Table 2-3: Pin Related to V850E/CA4 Status 33
Table 2-4: Pin Functions 34
Table 3-1: Program Registers 55
Table 3-2: System Register Numbers 56
Table 3-3: Saturated Operation Results 60
Table 3-4: Peripheral I/O Registers 76
Table 3-5: CAN RAM Mapping 83
Table 3-6: CAN Registers Mapping 86
Table 4-1: Relationship Between Transfer Type and Transfer Object 104
Table 5-1: Interrupt Source List 109
Table 5-2: Priorities of Interrupts and Exceptions 139
Table 6-1: Operating Status in HALT Mode 157
Table 6-2: Operation after Releasing HALT Mode by Interrupt Request 158
Table 6-3: Operating Status in IDLE Mode 159
Table 6-4: Operation after Releasing IDLE Mode by Interrupt Request 160
Table 6-5: Operating Status in WATCH Mode 161
Table 6-6: Operation after Releasing Watch Mode by Interrupt Request 163
Table 6-7: Operating Status in STOP Mode 165
Table 6-8: Operation after Releasing Stop Mode by Interrupt Request 167
Table 7-1: Timer Dn Configuration List (n = 0, 1) 173
Table 7-2: Timer Gn Configuration List 184
Table 7-3: Interrupt Output and Timer Output States Dependent on the Register Setting Values 195
Table 7-4: Interrupt Output and Timer Output States Dependent on the Register Setting Values 196
Table 8-1: Configuration of Watch Timer 222
Table 9-1: Configuration of Watchdog Timer 227
Table 9-2: Watchdog Timer Clock Selection 228
Table 10-1: Configuration of Serial Interface UART6n 236
Table 10-2: Write processing and writing to TXBn during execution of continuous transmission .. 259
Table 10-3: Cause of Reception Error 264
Table 10-4: Set Data of Dedicated Baud Rate Generator 272
Table 11-1: Configuration of the CAN Message Buffer Section 323
Table 11-2: CAN Message Buffer Registers Layout 324
Table 11-3: Relative Addresses of CAN Interrupt Pending Registers 325
Table 11-4: Relative Addresses of CAN Common Registers 326
Table 11-5: Relative Addresses of CAN Module 1 Registers 327
Table 11-6: Relative Addresses of CAN Module 2 Registers 328
Table 11-7: Transmitted Data on the CAN Bus (ATS = 1) 332
Table 11-8: Example for Automatic Transmission Priority Detection 334
Table 11-9: Example for Transmit Buffer Allocation When More Than 5 Buffers
Linked to a CAN Module 335
Table 11-10: Storage Priority for Reception of Data Frames 336
Table 11-11: Storage Priority for Reception of Remote Frames 336
Table 11-12: Inner Storage Priority Within a Priority Class 337
Table 11-13: Remote Frame Handling upon Reception into a Transmit Message Buffer 341
Table 11-14: CAN Message Processing by TRQ and RDY Bits 366
Table 11-15: Address Offsets of the CAN 1 to 2 Mask Registers 376
Table 12-1: Correspondence between ADCRm (m = 0 to 13) Register Names and Addresses 423
Table 12-2: Correspondence Between each Analog Input Pin and ADCRm Registers 424
Table 13-1: Port Type 462
Table 13-2: Filter Assignment 476
Table 14-1: Hardware Status on $\overline{\text{RESET}}$ Pin Input or Occurrence of WDTRES 480
Table 15-1: Serial Interface Pins 489

Table 15-2: List of Communication Systems 492

Chapter 1 Introduction

1.1 General

The V850E/CA4 HELIOS single chip 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 V850E CPU offers easy pipeline handling and programming, resulting in compact code size comparable to 16-bit CISC CPUs.

The V850E/CA4 offers an excellent combination of general purpose peripheral functions, like serial communication interfaces (UART, clocked SI), timers 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/CA4 HELIOS is ideally suited for automotive applications.

(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 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) On-chip flash memory

The µPD70F3175 has on-chip an high speed flash memory, which is able to fetch one instruction within one clock cycle. It is possible to program the user application directly on the target board, on which the V850E/CA4 is mounted. In such case system development time can be reduced and system maintainability after shipping can be markedly improved.

(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:	V850E
- Number of instructions:	81
- Min. instruction execution time:	31.2 ns (@ 32 MHz operation, 4.5 V to 5.5 V)
- General-purpose registers:	32 bits × 32 registers:

- Instruction set:
  - V850E
  - Signed multiplication
    (16 bits × 16 bits → 32 bits or 32 bits × 32 bits → 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

- Internal memory

Table 1-1: Product Versions

Part Number	Main Clock Type	Program Memory	Internal RAM	CAN RAM
µPD70F3175	Crystal	256 Kbytes (flash memory)	12 Kbytes	1 Kbyte 32 message buffers
µPD703175	Crystal	256 Kbytes (mask ROM)	12 Kbytes	1 Kbyte 32 message buffers
µPD703176	Crystal	192 Kbytes (mask ROM)	10 Kbytes	1 Kbyte 32 message buffers
µPD703174	Crystal	128 Kbytes (mask ROM)	8 Kbytes	1 Kbyte 32 message buffers

- Flash secure selfprogramming support

• Clock Generator	$f_X = f_{XX}$, $2f_{XX}$, $4f_{XX}$,
- Main oscillator frequency range	f_{XX} = 6 or 8 MHz
	(f_{XX} =connected X1, X2 oscillator)
• Built-in power saving modes:	HALT, IDLE, WATCH, STOP
• Power supply voltage range	4.5 V ≤ V_{DD} ≤ 5.5 V
• Temperature range:	Ta = -40°C to +85°C @ 32 MHz (A grade)
• I/O lines:	76
- Input only lines:	14
- Input/output lines:	62
• A/D Converter:	10-bit resolution; 14 channels
• Serial Interfaces	
- 3-wire mode:	3 channels
- UART mode:	2 channels

• CAN Interface (2.0B active):	2 channels
• Timers	
- 16-bits dual time-base timer (TMG)	2 channels
- 16-bits compare timer (TMD)	2 channels
- Watch timer:	1 channel
- Watchdog timer:	1 channel
• Interrupts	54 vectored interrupts
- Internal interrupt sources	45 (44 maskable + 1 non-maskable)
- External interrupt sources	11 (10 maskable + 1 non-maskable)
- Non maskable interrupts	2 (1 external: NMI, 1 internal: Watchdog)
- Software exceptions	32
- Exception trap	1
- Eight levels of priorities can be set (maskable interrupts)	
• Direct Memory Access	4 independent channels
• Package	100-pin plastic LQFP, 0.5 mm pin-pitch, 14 × 14 mm

1.3 Application Fields

The V850E/CA4 is a device designed for car manufacturers. It is ideally suited for automotive applications, like Safety System. It is also an excellent choice for other applications where a combination of sophisticated peripheral functions with CAN network support is required like Body Electronics Application.

1.4 Ordering Information

Part number	Package	Internal ROM
µPD70F3175GC-32	100-pin plastic LQFP, 0.5 mm pin-pitch, 14 × 14 mm	Flash memory
µPD703175GC-32-xxx	100-pin plastic LQFP, 0.5 mm pin-pitch, 14 × 14 mm	Mask ROM
µPD703175GC-24-xxx	100-pin plastic LQFP, 0.5 mm pin-pitch, 14 × 14 mm	Mask ROM
µPD703176GC-32-xxx	100-pin plastic LQFP, 0.5 mm pin-pitch, 14 × 14 mm	Mask ROM
µPD703176GC-24-xxx	100-pin plastic LQFP, 0.5 mm pin-pitch, 14 × 14 mm	Mask ROM
µPD703174GC-24-xxx	100-pin plastic LQFP, 0.5 mm pin-pitch, 14 × 14 mm	Mask ROM

Remarks: 1. xxx indicates ROM code suffix

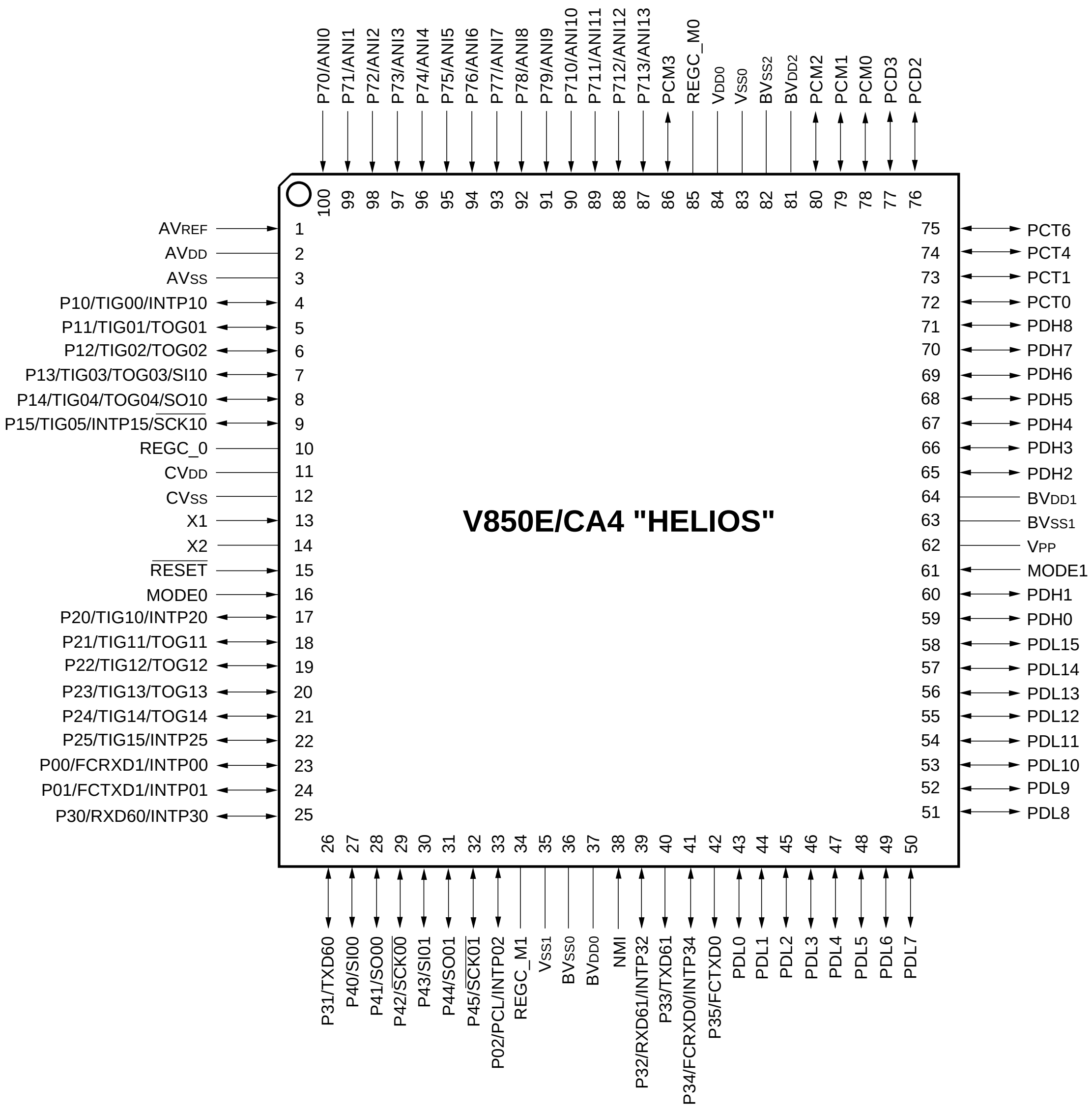
2. -24- and -32- indicate CPU frequency

Note: Only available in grade A.

1.5 Pin Configuration (Top View)

100-pin LQFP (0.5 mm pin pitch) (14 × 14 mm)

Figure 1-1: Pin Configuration of the μPD70F317x



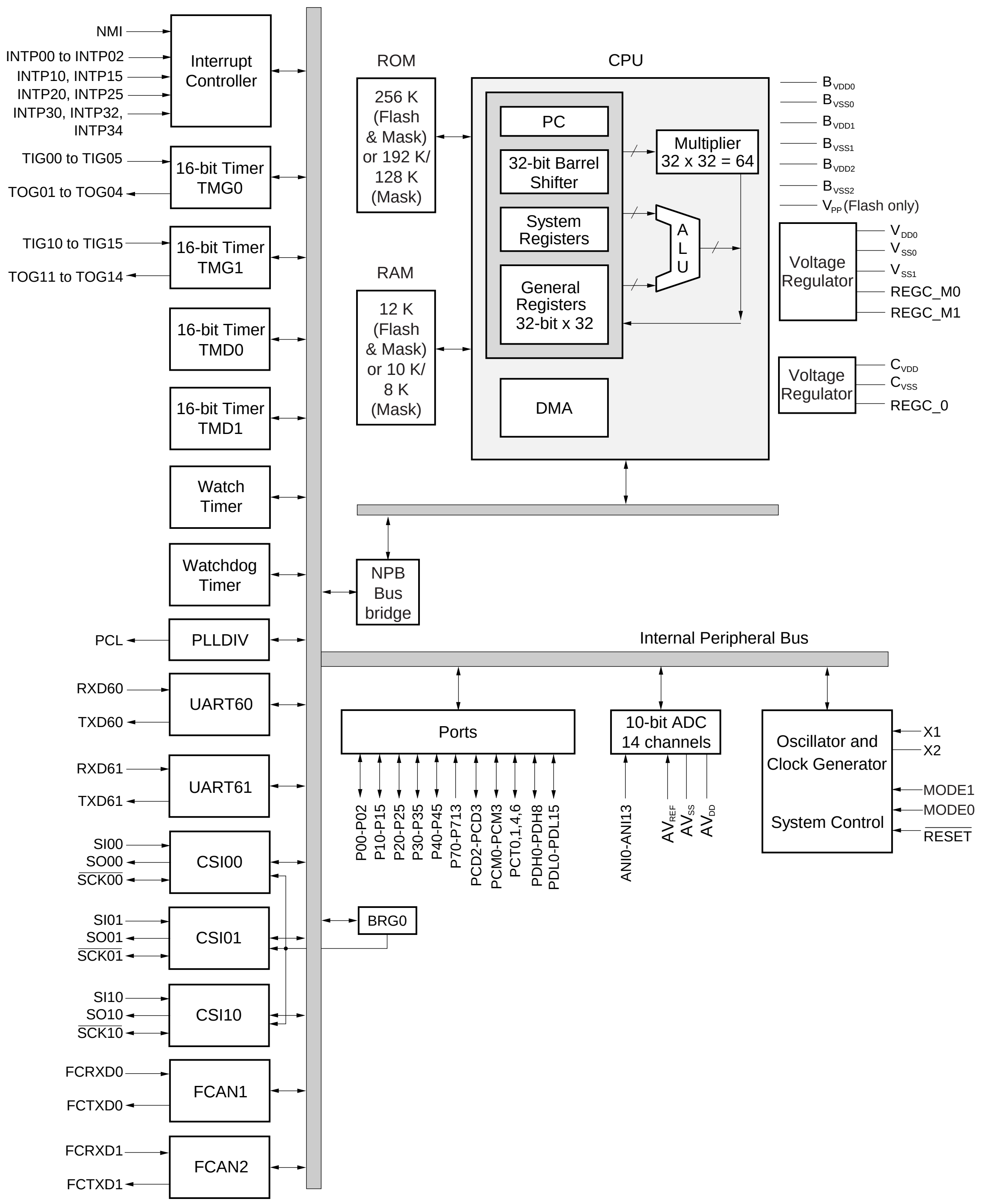
Pin Identification

ANI0 to ANI13	Analog Inputs	TOG01 to TOG04	Timer G0 Compare Output
AV_{DD}	Analog Power Supply	TOG11 to TOG14	Timer G1 Compare Output
AV_{SS}	Analog Ground	TIG00 to TIG05	Timer G0 Capture Input
AV_{REF}	Analog reference Voltage supply	TIG10 to TIG15	Timer G1 Capture Input
PCL	Processor Clock Output	REGC_M0, REGC_M1	Main Regulator Output
FCRXD0, FCRXD1	CAN Receive Data for channel 0 and 1	REGC_O	Osc and PLL Regulator Output
FCTXD0, FCTXD1	CAN Transmit Data for channel 0 and 1	MODE0, MODE1	Operation mode select
INTP00, INTP01, INTP02, INTP10, INTP15, INPT20, INTP25, INTP30, INTP32, INTP34	External Interrupt Input	X1, X2	Main System Clock
NMI	Non-Maskable Interrupt Input	$\overline{\text{RESET}}$	Reset Input
P00 to P02	Port 0	CV_{DD}, CV_{SS}	Oscillator and PLL power supply
P10 to P15	Port 1	V_{DD0}	Digital power supply for Flash, CPU and I/O buffer
P20 to P25	Port 2	V_{SS0}, V_{SS1}	Digital Ground for Flash, CPU and I/O buffer
P30 to P35	Port 3	BV_{SS0} to BV_{SS2}	I/O buffers Ground
P40 to P45	Port 4	BV_{DD0} to BV_{DD2}	I/O buffers supply
P70 to P713	Port 7	V_{PP}	Programming Voltage
PCM0 to PCM3	Port CM		
PCT0, PCT1, PCT4, PCT6	Port CT		
PCD2 to PCD3	Port CD		
PDL0 to PDL15	Port DL		
PDH0 to PDH8	Port DH		
RXD60, RXD61	UART Receive Data		
TXD60, TXD61	UART Transmission Data		
$\overline{\text{SCK00}}$, $\overline{\text{SCK01}}$, $\overline{\text{SCK10}}$	Synchronous Interface Clock		
SI00, SI01, SI10	Synchronous Interface Input		
SO00, SO01, SO10	Synchronous Interface Output		

1.6 Configuration of Function Block

1.6.1 Block Diagram of V850E/CA4

Figure 1-2: Block Diagram of the V850E/CA4 Microcontroller



1.6.2 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 $\times$ 16 bits $\rightarrow$ 32 bits or 32 bits $\times$ 32 bits $\rightarrow$ 64 bits) and the barrel shifter (32 bits), help accelerate processing of complex instructions.

(2) ROM

The µPD70F3175 has an on-chip flash memory of 256 Kbytes.
The µPD703175 has an on-chip mask ROM memory of 256 Kbytes.
The µPD703176 has an on-chip mask ROM memory of 192 Kbytes.
The µPD703174 has an on-chip mask ROM memory of 128Kbytes.
During instruction fetch, flash memory can be accessed from the CPU in 1-clock cycles.
The memory mapping is done from address 00000000H.

(3) RAM

The RAM area is mapped from address FFFFC000H for the µPD70F3175 and µPD703175.
The RAM area is mapped from address FFFFC800H for the µPD703176.
The RAM area is mapped from address FFFFD000H for the µPD703174.
Data in RAM can be accessed from the CPU in 1-clock cycles.

(4) Interrupt controller (INTC)

This controller handles hardware interrupt requests (NMI, INTPxy) 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.

(5) Clock generator (CG)

This clock generator supplies frequencies for the CPU and the built-in peripherals. As the input clock, an external crystal is connected to pins X1 and X2.

(6) Serial interface (SIO)

The 2-channels asynchronous serial interface (UART6), 3-channels clocked serial interface (CSI0 and CSI1) and 2-channels FCAN are provided as serial interface.

UART6 transfers data by using the TXD6n and RXD6n pins. (n = 0, 1).
CSI0 transfers data by using the SO0n, SI0n, and $\overline{\text{SCK0n}}$ pins. (n = 0 to 1).
CSI1 transfers data by using the SO10, SI10, and $\overline{\text{SCK10}}$ pins.
FCAN performs data transfer using FCTXDn and FCRXDn pins. (n = 0 to 1).

(7) Real time pulse unit

The timers provided with HELIOS have compare and capture functions with 16 bits capability.
Two dual-timebase timers G are provided, each with 6 16-bits compare registers and 4 16-bits capture registers.
Two general purpose timers D are also provided, each with one 16-bits compare register.

(8) DMA

The DMA controller handles 4 independent prioritized channels.

(9) A/D converter (ADC)

One high-resolution 10-bit A/D converter, it includes 14 analog input pins. Conversion uses the successive approximation method.

(10) Ports

Refer to Chapter 13 ”Port Feature” on page 437 for detailed description of each port and alternate function.

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)

Pin Name	I/O	Function	Driver Type	Alternate
P00	I/O	Port 0: 3-bit input/output port	5-K	FCRXD1/INTP00
P01				FCTXD1/INTP01
P02				PCL/INTP02
P10	I/O	Port 1: 6-bit input/output port	5-K	TIG00/INTP10
P11				TIG01/TOG01
P12				TIG02/TOG02
P13				TIG03/TOG03/SI10
P14				TIG04/TOG04/SO10
P15				TIG05/INTP15/$\overline{\text{SCK10}}$
P20	I/O	Port 2: 6-bit input/output port	5-K	TIG10/INTP20
P21				TIG11/T0G11
P22				TIG12/T0G12
P23				TIG13/T0G13
P24				TIG14/T0G14
P25				TIG15/INTP25
P30	I/O	Port 3: 6-bit input/output port	5-K	RXD60/INTP30
P31				TXD60
P32				RXD61/INTP32
P33				TXD61
P34				FCRXD0/INTP34
P35				FCTXD0
P40	I/O	Port 4: 6-bit input/output port	5-K	SI00
P41				SO00
P42				SCK00
P43				SI01
P44				SO01
P45				SCK01

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

Pin Name	I/O	Function	Driver Type	Alternate
P70	I	Port 7: 14-bit input port	9	ANI0
P71				ANI1
P72				ANI2
P73				ANI3
P74				ANI4
P75				ANI5
P76				ANI6
P77				ANI7
P78				ANI8
P79				ANI9
P710				ANI10
P711				ANI11
P712				ANI12
P713				ANI13
PCT0	I/O	Port CT: 4-bit input/output port	5	
PCT1				
PCT4				
PCT6				
PDH0	I/O	Port DH: 9-bit output port	5	
PDH1				
PDH2				
PDH3				
PDH4				
PDH5				
PDH6				
PDH7				
PDH8				
PCM0	I/O	Port CM: 4-bit output port	5	
PCM1				
PCM2				
PCM3				
PCD2	I/O	Port CD: 2-bit output port	5	
PCD3				

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

Pin Name	I/O	Function	Driver Type	Alternate
PDL0	I/O	Port DL: 16-bit input/output port	5	
PDL1				
PDL2				
PDL3				
PDL4				
PDL5				
PDL6				
PDL7				
PDL8				
PDL9				
PDL10				
PDL11				
PDL12				
PDL13				
PDL14				
PDL15				

(2) Non-port pins

Table 2-2: Non-Port Pins

Pin Number	Pin name	Connection for normal operation	I/O
1	AV_{REF}	Analog voltage reference for A/D converter	-
2	AV_{DD}	Analog Power Supply	-
3	AV_{SS}	Analog Ground	-
10	REGC_O	connect to CV_{SS} via a capacitor Note 1	Output
11	CV_{DD}	Power supply pin for oscillator and PLL	-
12	CV_{SS}	Ground potential pin for oscillator and PLL	-
13	X1	Refer to page 33 for recommended circuit	-
14	X2		-
15	$\overline{\text{RESET}}$	External system reset input	Input
16	MODE0	Connect to V_{SSn} via a resistor	Input
34	REGC_M1	Connect to V_{SS1} via a capacitor Note 2	Output
35	V_{SS1}	Ground potential pin for Flash, CPU and I/O buffers	-
36	BV_{SS0}	Ground potential pin for I/O buffers	-
37	BV_{DD0}	Power supply pin for I/O buffers	-
NMI	NMI	NMI	Input
61	MODE1	connect to V_{SSn} via a resistor	Input
62	V_{PP} Note 3	On Flash devices connect V_{PP} to ground via a resistor.	Input
63	BV_{SS1}	Ground potential pin for I/O buffers	
64	BV_{DD1}	Power supply pin for I/O buffers	
81	BV_{DD2}	Power supply pin for I/O buffers	-
82	BV_{SS2}	Ground potential pin for I/O buffers	-
83	V_{SS0}	Ground potential pin for Flash, CPU and I/O buffers	-
84	V_{DD0}	Power supply pin for Flash, CPU and I/O buffers	-
85	REGC_M0	Connect to REGC_M1 pin with the shortest way (lowest impedance)	Output

Notes: 1. NEC specifies to connect a minimum 330 nF Capacitor.

2. NEC specifies to connect a minimum 1 µF Capacitor.

3. Only for µPD70F3175 (Flash product)

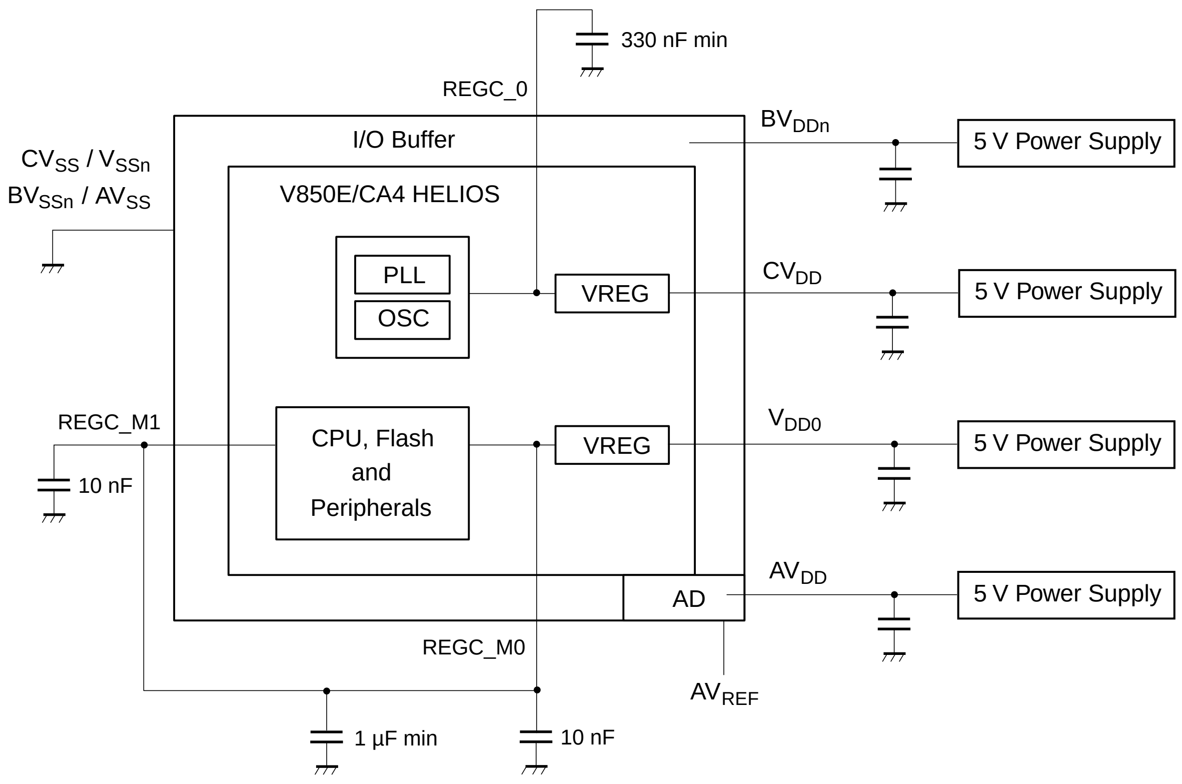
Cautions: 1. On REGC-pin and each pin of V_{DDn}, a capacitor has to be attached as tight as possible to the pin.

2. The capacitors used should have only very low serial impedance.

3. All ground pin have to be connected together.

4. For EMI optimization, NEC recommend to separate power supply for V_{DDn}, CV_{DDn} and BV_{DD} (refer to Figure 2-1, “Power Supply Connection,” on page 33).

Figure 2-1: Power Supply Connection



(3) Pin related to V850E/CA4 status

Table 2-3: Pin Related to V850E/CA4 Status

Pin name	HALT mode	WATCH mode	STOP mode	In Reset	After Reset
P00 to P05	Status Hold	Status Hold	Status Hold	Hi-Z	Hi-Z
P10 to P15					
P20 to P25					
P30 to P34					
P40 to 47					
P50 to P57					
P70 to P711					
PCT0, PCT1, PCT4, PCT6					
PDH0 to PDH5					
PCM0 to PCM3					
PDL0 to PDL15					

Table 2-4: Pin Functions (1/4)

Pin		Function		I/O	Buffer Type
No.	Name	Default	Alternate		
1	AV_{REF}	Reference-Voltage supply pin for A/D converter	-	-	-
2	AV_{DD}	Power supply pin for A/D converter	-	-	-
3	AV_{SS}	Ground potential for A/D converter	-	-	-
4	P10/TIG00/INTP10	Port 1: 6-bit input/output port	TimerG0 Capture Trigger 0 External interrupt input INTP10	I/O	5-K
5	P11/TIG01/TOG01		TimerG0 Capture Trigger 0 TimerG0 Compare Output 0	I/O	
6	P12/TIG02/TOG02		TimerG0 Capture Trigger 0 TimerG0 Compare Output 0	I/O	
7	P13/TIG03/TOG03/SI10		TimerG0 Capture Trigger 0 TimerG0 Compare Output 0 CSI1 channel 0 serial data input	I/O	
8	P14/TIG04/TOG04/SO10		TimerG0 Capture Trigger 0 TimerG0 Compare Output 0 CSI1 channel 0 serial data output	I/O	
9	P15/TIG05/INTP15/$\overline{\text{SCK10}}$		TimerG0 Capture Trigger 0 External interrupt input INTP15 CSI1 channel 0 serial clock input	I/O	
10	REGC_O	Pin for external 3.3 V Regulating Capacitor	-	-	-
11	CV_{DD}	Power supply pin for oscillator and PLL	-	-	-
12	CV_{SS}	Ground potential pin for oscillator and PLL	-	-	-
13	X1	Resonator connection for clock	-	-	-
14	X2	Resonator connection for clock	-	-	-
15	$\overline{\text{RESET}}$	External System reset input	-	-	2
16	MODE0	MODE Definition Input pins	-	I	2

Table 2-4: Pin Functions (2/4)

Pin		Function		I/O	Buffer Type
No.	Name	Default	Alternate		
17	P20/TIG10/INTP20	Port 2: 6-bit input/output port	TimerG1 Capture Trigger 0 External interrupt input INTP20	I/O	5-K
18	P21/TIG11/TOG11		TimerG1 Capture Trigger 0 TimerG1 Compare Output 0	I/O	
19	P22/TIG12/TOG12		TimerG1 Capture Trigger 0 TimerG1 Compare Output 0	I/O	
20	P23/TIG13/TOG13		TimerG1 Capture Trigger 0 TimerG1 Compare Output 0	I/O	
21	P24/TIG14/TOG14		TimerG1 Capture Trigger 0 TimerG1 Compare Output 0	I/O	
22	P25/TIG15/INTP25		TimerG1 Capture Trigger 0 External interrupt input INTP25	I/O	
23	P00/FCRXD1/INTP00	Port 0: 3-bit input/output port	FCAN channel 1 serial data input External interrupt input INTP00	I/O	
24	P01/FCTXD1/INTP01		FCAN channel 1 serial data output External interrupt input INTP01	I/O	
25	P30/RXD60/INTP30	Port 3: 6-bit input/output port	UART60 asynchronous data input External interrupt input INTP30	I/O	
26	P31/TXD60		UART60 asynchronous data out-put	I/O	
27	P40/SI00	Port 4: 6-bit input/output port	CSI0 channel 0 serial data input	I/O	
28	P41/SO00		CSI0 channel 0 serial data output	I/O	
29	P42/$\overline{\text{SCK00}}$		CSI0 channel 0 serial clock input	I/O	
30	P43/SI01		CSI0 channel 1 serial data input	I/O	
31	P44/SO01		CSI0 channel 1 serial data output	I/O	
32	P45/$\overline{\text{SCK01}}$		CSI0 channel 1 serial clock input	I/O	
33	P02/PCL/INTP02	Port 0: 3-bit input/output port	Processor clock output External interrupt input INTP02	I/O	
34	REGC_M1	pin for external 3.3 V Regulat-ing Capacitor	-	-	-
35	V_{SS1}	Ground potential pin for Flash, CPU and I/O buffers	-	-	-
36	BV_{SS0}	Ground potential pin for I/O buffers	-	-	-
37	BV_{DD0}	Power supply pin for I/O buff-ers	-	-	-
38	NMI	Non-maskable interrupt input pin	-	I	2
39	P32/RXD61/INTP32	Port 3: 6-bit input/output port	UART61 asynchronous data input External interrupt input INTP32	I/O	5-K
40	P33/TXD61		UART61 asynchronous data out-put	I/O	
41	P34/FCRXD0/INTP34		FCAN channel 0 serial data input External interrupt input INTP34	I/O	
42	P35/FCTXD0		FCAN channel 0 serial data output	I/O	

Table 2-4: Pin Functions (3/4)

Pin		Function		I/O	Buffer Type
No.	Name	Default	Alternate		
43	PDL0	Port DL: 16-bit input/output port	-	I/O	5
44	PDL1		-	I/O	
45	PDL2		-	I/O	
46	PDL3		-	I/O	
47	PDL4		-	I/O	
48	PDL5		-	I/O	
49	PDL6		-	I/O	
50	PDL7		-	I/O	
51	PDL8		-	I/O	
52	PDL9		-	I/O	
53	PDL10		-	I/O	
54	PDL11		-	I/O	
55	PDL12		-	I/O	
56	PDL13		-	I/O	
57	PDL14		-	I/O	
58	PDL15		-	I/O	
59	PDH0	Port DH: 9-bit output port	-	I/O	
60	PDH1		-	I/O	
61	MODE1	MODE Definition Input pins	-	I/O	2
62	V_{PP} **Note**	High Voltage apply pin to program the device	-	-	-
63	BV_{SS1}	Ground potential pin for I/O buffers		-	-
64	BV_{DD1}	Power supply pin for I/O buffers		-	-
65	PDH2	Port DH: 9-bit output port	-	I/O	5
66	PDH3		-	I/O	
67	PDH4		-	I/O	
68	PDH5		-	I/O	
69	PDH6		-	I/O	
70	PDH7		-	I/O	
71	PDH8		-	I/O	
72	PCT0	Port CT: 4-bit input/output port	-	I/O	
73	PCT1		-	I/O	
74	PCT4		-	I/O	
75	PCT6		-	I/O	
76	PCD2	Port CD: 2-bit output port	-	I/O	
77	PCD3		-	I/O	
78	PCM0	Port CM: 4-bit output port	-	I/O	
79	PCM1		-	I/O	
80	PCM2		-	I/O	

Table 2-4:   Pin Functions (4/4)

Pin		Function		I/O	Buffer Type
No.	Name	Default	Alternate		
81	BV_{DD2}	Power supply pin for I/O buffers	-	-	-
82	BV_{SS2}	Ground potential pin for I/O buffers	-	-	-
83	V_{SS0}	Ground potential pin for Flash, CPU and I/O buffers	-	-	-
84	V_{DD0}	Power supply pin for Flash, CPU and I/O buffers	-	-	-
85	REGC_M0	pin for external 3.3 V Regulating Capacitor	-	-	-
86	PCM3	Port CM: 4-bit output port	-	I/O	5
87	P713/ANI13	Port 7: 14-bit input port	ANI13	I/O	9
88	P712/ANI12		ANI12	I/O	
89	P711/ANI11		ANI11	I/O	
90	P710/ANI10		ANI10	I/O	
91	P79/ANI9		ANI9	I/O	
92	P78/ANI8		ANI8	I/O	
93	P77/ANI7		ANI7	I/O	
94	P76/ANI6		ANI6	I/O	
95	P75/ANI5		ANI5	I/O	
96	P74/ANI4		ANI4	I/O	
97	P73/ANI3		ANI3	I/O	
98	P72/ANI2		ANI2	I/O	
99	P71/ANI1		ANI1	I/O	
100	P70/ANI0		ANI0	I/O	

Note: Only for µPD70F3175 (Flash product)

2.2 Description of Pin Functions

(1) P00 to P02 (Port 0) … Input/output

Port 0 is an 3-bit input/output port in which input or output can be set in 1-bit units.
Besides functioning as an input/output port, P00 and P01 operate as CAN channel 1 data reception and transmission, and P02 operates as processor clock output.
In addition, each P0 bit P00 to P02 operates as external interrupt request input pin.

An operation mode of port or control mode can be selected for each bit and specified by the port 0 mode control register (PMC0).

(a) Port mode

P00 to P02 can be set to input or output in 1-bit units using the port 0 mode register (PM0).

(b) Control mode

P00 to P02 can be set to port or control mode in 1-bit units using PMC0.

(c) FCTXD1 (Transmit data for controller area network channel 1) … Output

This pin outputs FCAN channel 1 serial transmit data.

(d) FCRXD1 (Receive data for controller area network channel 1) … Input

This pin inputs FCAN channel 1 serial receive data.

(e) PCL (Processor Clock Output)

This is the internal system clock output pin.

(f) INTP00 to INTP02 (Interrupt request from peripherals) … Input

These are maskable external interrupt request input pins.

(2) P10 to P15 (Port 1) … Input/output

Port 1 is an 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, P10 to P15 operate as Timer G0 input/output and P13 to P15 operate as input/output for the clocked serial link CSI10.
Furthermore P10 and P15 operate as maskable external interrupt request input pins.

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 P15 can be set to input or output in 1-bit units using the port 1 mode register (PM1).

(b) Control mode

P10 to P15 can be set to port or control mode in 1-bit units using PMC1.

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

These pins output a Timer G0 PWM pulse signal.

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

These pins input a Timer G0 external capture input pin.

(e) SO10 (Serial output) … Output

This pin outputs CSI10 serial transmit data.

(f) SI10 (Serial input) … Input

This pin inputs CSI10 serial receive data.

(g) $\overline{\text{SCK10}}$ (Serial clock) … Input/output

This is CSI10 serial clock input/output pin.

(h) INTP10 and INTP15 (Interrupt request from peripherals) … Input

These are maskable external interrupt request input pins.

(3) P20 to P25 (Port 2) … Input/Output

Port 2 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, P20 to P25 operate as Timer G1 input/output.
Furthermore P20 and P25 operate as maskable external interrupt request input pins.

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 P25 can be set to input or output in 1-bit units using the port 2 mode register (PM2).

(b) Control mode

P20 to P25 can be set to port or control mode in 1-bit units using PMC2.

(c) TOG11 to TOG14 (Timer output) … Output

These pins output a Timer G1 PWM pulse signal.

(d) TIG10 to TIG15 (Timer input) … Input

These pins input a Timer G1 external capture input pin.

(e) INTP20 and INTP25 (Interrupt request from peripherals) … Input

These are maskable external interrupt request input pins.

(4) 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 and P31 operate as UART60 data emission/reception, P32 and P33 operate as UART61 data emission/reception, P34 and P35 operate as CAN channel 0 data reception and transmission.
Furthermore P30, P32 and P34 operate as maskable external interrupt request input pins.

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) TXD60, TXD61 (Transmit data) … Output

These pins output UART60 and UART61serial transmit data.

(d) RXD60, RXD61 (Receive data) … Input

These pins input UART60 and UART61 serial receive data.

(e) FCTXD0 (Transmit data for controller area network channel 0) … Output

This pin outputs FCAN channel 0 serial transmit data.

(f) FCRXD0 (Receive data for controller area network channel 0) … Input

This pin inputs FCAN channel 0 serial receive data.

(g) INTP30, INTP32 and INTP34 (Interrupt request from peripherals) … Input

These are maskable external interrupt request input pins.

(5) 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 P42 operate as clocked serial link CSI00 input/output, and P43 to P45 operate as clocked serial link CSI01 input/output.

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 P47 can be set to input or output in 1-bit units using the port 4 mode register (PM4).

(b) Control mode

P40 to P47 can be set to port or control mode in 1-bit units using PMC4.

(c) SO00 and SO01 (Serial output) … Output

These pins output CSI00 and CSI01 serial transmit data.

(d) SI00 and SI01 (Serial input) … Input

These pins input CSI00 and CSI01 serial receive data.

(e) $\overline{\text{SCK00}}$ and $\overline{\text{SCK01}}$ (Serial clock) … Input/output

These are CSI00 and CSI01serial clock input/output pins.

(6) P70 to P713 (Port 7) … Input

Port 7 is a 14-bit input-only port.
Besides functioning as an input port, in control mode, P70 to P713 operate as analog input pins to the A/D converter.

An operation mode of port or control mode can be selected in 8-bit units and specified by the port 7 mode control register (PMC7).

(a) Port mode

P70 to P713 are input-only pins.

(b) Control mode

P70 to P713 can be set to port or control mode in 8-bit units using PMC7.

(c) ANI0 to ANI13 (Analog input) … Input

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

(7) PDL0 to PDL15 (Port DL) … Input/output

Port DL is a 16-bit input/output port in which input or output can be set in 1-bit units.

(a) Port mode

PDL0 to PDL15 can be set to input or output in 1-bit units using the port DL mode register (PMDL).

(8) PDH0 to PDH8 (Port DH) … Input/output

Port DH is a 9-bit input/output port in which input or output can be set in 1-bit units.

(a) Port mode

PDH0 to PDH8 can be set to input or output in 1-bit units using the port DH mode register (PMDH).

(9) PCT00, PCT01, PCT04 and PCT06 (Port CT) … Input/output

Port CT is a 4-bit input/output port in which input or output can be set in 1-bit units.

(a) Port mode

PCT00, PCT01, PCT04 and PCT06 can be set to input or output in 1-bit units using the port CT mode register (PMCT).

(10) PCD2 to PCD3 (Port CD) … Input/output

Port CD is a 2-bit input/output port in which input or output can be set in 1-bit units.

(a) Port mode

PCD2 to PCD3 can be set to input or output in 1-bit units using the port CD mode register (PMCD).

(11) PCM0 to PCM3 (Port CM) … Input/output

Port CM is a 4-bit input/output port in which input or output can be set in 1-bit units.

(a) Port mode

PCM0 to PCM3 can be set to input or output in 1-bit units using the port CM mode register (PMCM).

(12) V_{PP} (Flash Memory Programming Voltage)

High voltage apply pin for FLASH programming mode setting. Connect to BV_{SS1} in normal operating mode. To use this pin for on board flash programming connect this pin with a pull down resistor to BV_{SS1}.

(13) $\overline{\text{RESET}}$ (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, WATCH, IDLE or STOP).

(14) NMI (Non-Maskable Interrupt Request)... input

This is the non-maskable interrupt request input pin.

(15) X1, X2 (Crystal)

These pins connect a resonator or crystal for system clock generation.
They also can input external clocks. For external clock input, connect to the X1 pin and leave the X2 pin open.

(16) REGC_O, REGC_M0 and REGC_M1 (Capacitor)

These pins connect an external Capacitor for the internal voltage system. This capacitor is used as a regulating capacitor for the internal voltage generator.

(17) BV_{DD0} to BV_{DD2} (Power supply)

This is the positive power supply pin for the I/O buffers.

(18) BV_{SS0} to BV_{SS2} (Ground)

This is the ground pin for the I/O buffers.

(19) V_{DD0} (Power supply)

This is the positive power supply pin for the flash, CPU and I/O buffer.

(20) V_{SS0} and V_{SS1} (Ground)

These are the ground pins for the flash, CPU and I/O buffer.

(21) CV_{DD} (Clock power supply)

This is the positive power supply pin for the oscillator and PLL.

(22) CV$_{SS}$ (Clock ground)

This is the ground pin for the oscillator and PLL.

(23) AV$_{DD}$ (Analog power supply)

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

(24) AV$_{SS}$ (Analog ground)

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

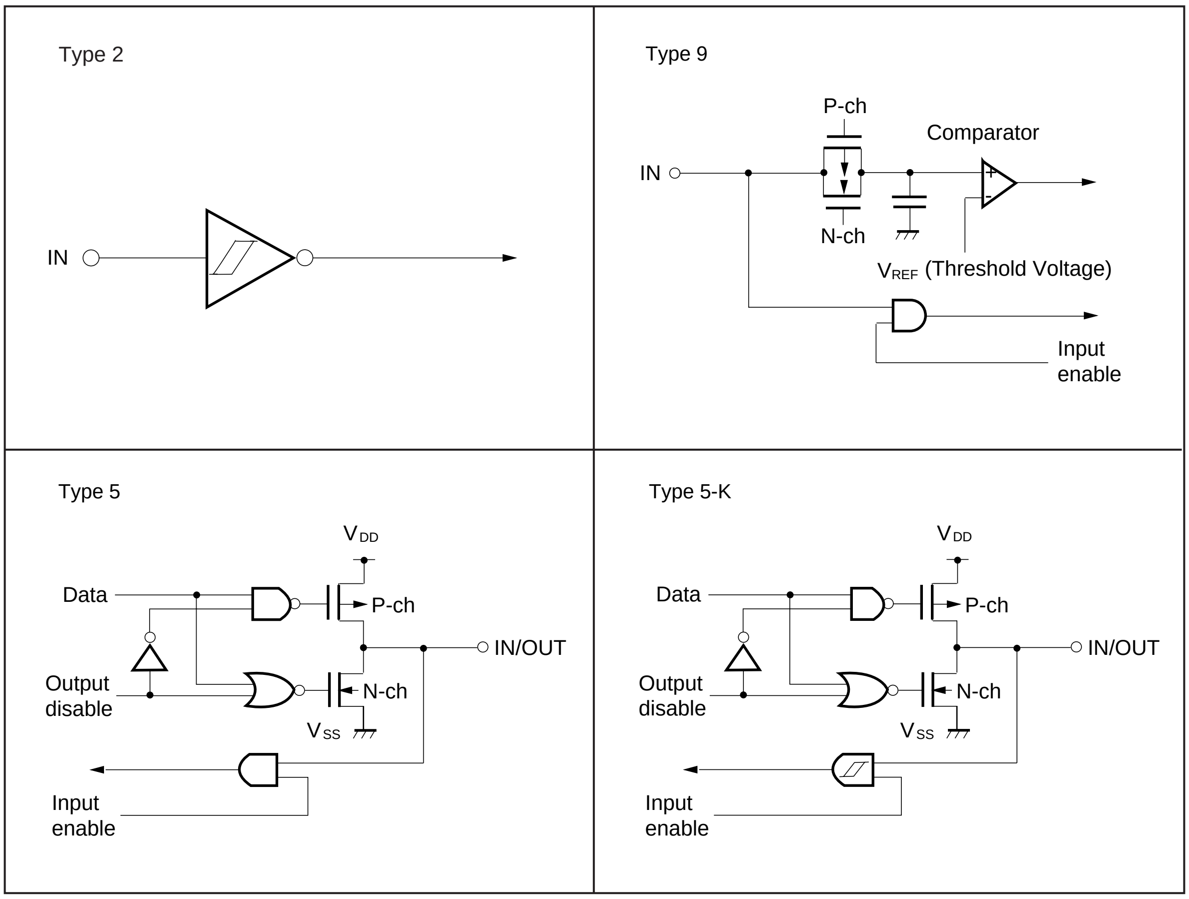
(25) AV$_{REF}$ (Analog reference voltage) … Input

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

(26) MODE0 and MODE1 (Operating modes)... Input

These are the mode definition pins and must be connected to ground.

Figure 2-2: Pin I/O Circuits



[MEMO]

Chapter 3 CPU Functions

The CPU of the V850E/CA4, which is based on RISC architecture, executes almost all instructions in one clock cycle due to its five-stage pipeline control.

3.1 Features

• Number of instructions:	81
• Minimum instruction execution time:	31.2 ns (@ 32 MHz operation, 4.5 V to 5.5 V)

- Memory space
    - Program area: 256 MB linear address space
    - Data area: 4 GB linear address space
    - Memory bank division function: 2, 4, or 8 MB/bank
- General-purpose registers: 32 bits × 32 registers
- Internal 32-bit architecture
- 5-stage pipeline control
- Instruction set
    - Upwardly compatible with V850 CPU
    - Saturated calculation instructions (with overflow/underflow detection function)
    - 32-bit shift instructions: 1 clock
    - Load/store instructions with long/short format
    - Signed load instructions
    - Multiplication can be performed in 1 or 2 clocks due to on-chip hardware multiplier
        - 16 bits × 16 bits → 32 bits
        - 32 bits × 32 bits → 32 bits or 64 bits in one clock cycle.
    - Four type of bit manipulation instructions
        - Set
        - Clear
        - Not
        - Test

3.2 CPU Register Set

The CPU registers of the V850E/CA4 can be classified into general purpose register set, which are used by programs, and system register set, which are used to control the execution environment. This chapter describe also specific registers which can be read or written using the LDSR and STSR instructions. All the registers have 32-bit width.

For details, refer to V850E1 User's Manual Architecture.
(Document No. U14559EJ2V0UM00 (2nd edition))

Figure 3-1: CPU Register Set

(1) Program register set

31	0
r0	(Zero register)
r1	(Assembler-reserved register)
r2	
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	(Interrupt status saving register)
EIPSW	(Interrupt status saving register)
FEPC	(NMI status saving register)
FEPSW	(NMI status saving register)
ECR	(Interrupt source register)
PSW	(Program status word)
CTPC	(CALLT execution status saving register)
CTPSW	(CALLT execution status saving register)
DBPC	(Exception/debug trap status saving register)
DBPSW	(Exception/debug trap status saving register)
CTBP	(CALLT base pointer)

3.2.1 Program register set

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

(1) General-purpose registers (r0 to r31)

Thirty-two general-purpose registers, r0 to r31, are available. All 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 always holds 0 and is used for operations that use 0 or offset 0 addressing. r30 is used as a base pointer when performing memory access with the SLD and SST short instructions.
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, and they must be restored to the registers after use. There are cases when r2 is used by the real-time OS. If r2 is not used by the real-time OS, r2 can be used as a variable register.

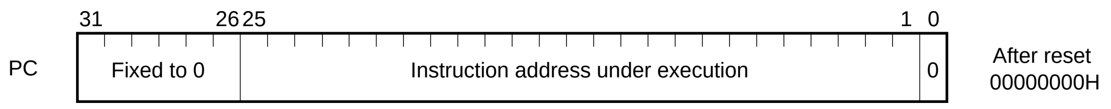
Table 3-1: Program Registers

Name	Usage	Operation
r0	Zero register	Always holds 0
r1	Assembler-reserved register	Working register for generating 32-bit immediate
r2	Address/data variable register (when r2 is not used by the real-time OS to be used)	
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 (area for placing program code)
r6 to r29	Address/data variable register	
r30	Element pointer	Base pointer when memory is accessed
r31	Link pointer	Used by compiler when calling function

(2) Program counter (PC)

This register holds the instruction address of the next instruction to be executed. 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 bit 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) Format



3.2.2 System register set

System registers control the status of the CPU and hold interrupt information.
Read from and write to system registers are performed by setting the system register numbers shown below with the system register load/store instructions (LDSR, STSR instructions).

Table 3-2: System Register Numbers

System Register			Operand Specification Enabled for instruction	
No.	Name	Function	LDSR	STSR
0	EIPC	PC value at Interrupt handler entry Note 1	Yes	Yes
1	EIPSW	PSW value at Interrupt handler entry Note 1	Yes	Yes
2	FEPC	PC value at NMI handler entry	Yes	Yes
3	FEPSW	PSW value at NMI handler entry	Yes	Yes
4	ECR	Exception Cause Register	No	Yes
5	PSW	Program status word	Yes	Yes
6 to 15	-	Reserved numbers for future function expansion (The operation is not guaranteed if accessed.)	No	No
16	CTPC	PC value at CALLT subroutine entry Note 2	Yes	Yes
17	CTPSW	PSW value at CALLT subroutine entry Note 2	Yes	Yes
18	DBPC	PC value at exception/debug trap entry	Yes	Yes
19	DBPSW	PSW value at exception/debug trap entry	Yes	Yes
20	CTBP	CALLT base pointer	Yes	Yes
21 to 31	-	Reserved numbers for future function expansion (The operation is not guaranteed if accessed.)	No	No

Notes: 1. Since only one set of these registers is available, the program must save the contents of these registers when multiple interrupt servicing is permitted.

2. Since only one set of these registers is available, the program must save the contents of these registers when CALLT instructions nesting are used.

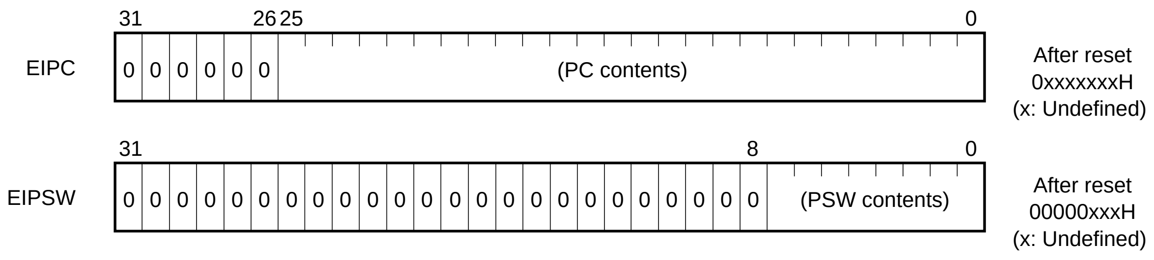
Caution: Even if bit 0 of EIPC, FEPC, or CTPC is set to (1) by the LDSR instruction, bit 0 is ignored during return with the RETI instruction following interrupt servicing (because bit 0 of PC is fixed to 0). When setting a value to EIPC, FEPC, and CTPC, set an even number (bit 0 = 0).

(1) Interrupt context saving registers (EIPC, EIPSW)

There are two context saving registers, EIPC and EIPSW.
Upon occurrence of a software exception or a maskable interrupt, the content of the program counter (PC) is saved to EIPC and the content of the program status word (PSW) is saved to EIPSW (upon occurrence of a non-maskable interrupt (NMI), the contents are saved to the NMI context saving registers (FEPC, FEPSW)).
The address of the next instruction following the instruction executed when a software exception or maskable interrupt occurs is saved to EIPC, except for the DIVH instruction (please see Chapter 5 ”Interrupt/Exception Processing Function” on page 109).

The current PSW contents are saved to EIPSW.
Since there is only one set of interrupt context saving registers, the contents of these registers must be saved by the program when multiple interrupt servicing is enabled.
Bits 31 to 26 of EIPC and bits 31 to 8 of EIPSW are reserved (fixed to 0) for future function expansion.

Figure 3-3: Interrupt Context Saving Registers (EIPC, EIPSW) Format

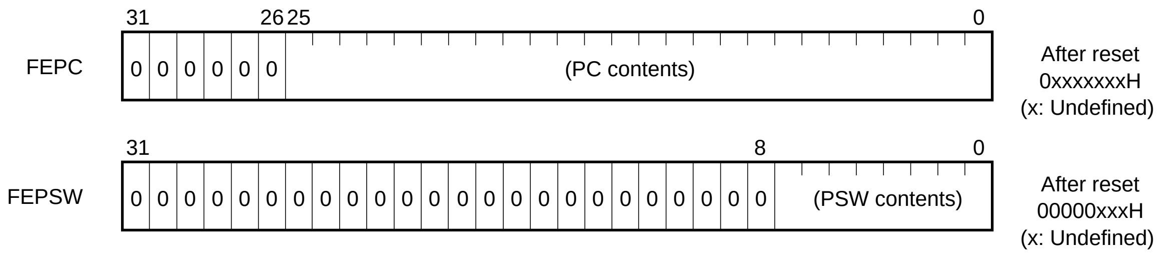


The values of EIPC and EIPSW are restored to PC and PSW during execution of a RETI instruction.

(2) NMI context saving registers (FEPC, FEPSW)

There are two NMI context saving registers, FEPC and FEPSW.
Upon occurrence of a non-maskable interrupt (NMI), the content of the program counter (PC) is saved to FEPC and the content of the program status word (PSW) is saved to FEPSW.
The address of the next instruction following the instruction executed when a non-maskable interrupt occurs is saved to FEPC, except for the DIVH instruction.
The current PSW contents are saved to FEPSW.
Bits 31 to 26 of FEPC and bits 31 to 8 of FEPSW are reserved (fixed to 0) for future function expansion.

Figure 3-4: NMI Context Saving Registers (FEPC, FEPSW) Format

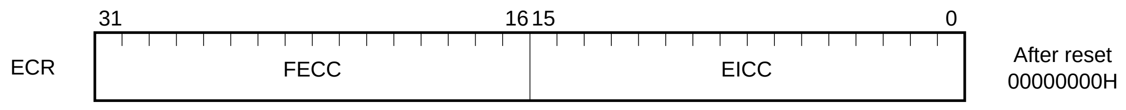


The values of FEPC and FEPSW are restored to PC and PSW during execution of a RETI instruction.

(3) Exception Cause Register (ECR)

Upon occurrence of an interrupt or an exception, the Exception Cause Register (ECR) holds the source of an interrupt or an exception. The value held by ECR is the exception code coded for each interrupt source. This register is a read-only register, and thus data cannot be written to it using the LDSR instruction.

Figure 3-5: Interrupt Source Register (ECR) Format



Bit position	Bit name	Description
31 to 16	FECC	Exception code of non-maskable interrupt (NMI)
15 to 0	EICC	Exception code of exception or maskable interrupt

The list of exception codes is tabulated in Table 5-1, “Interrupt Source List,” on page 109.

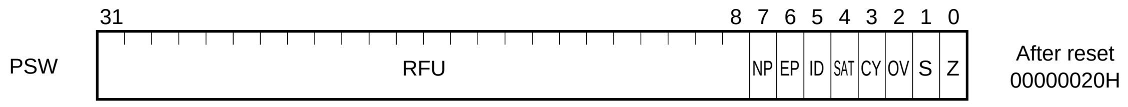
(4) Program status word (PSW)

A program status word (PSW) is a collection of flags that indicate the program status (instruction execution result) and the CPU status.

When the contents of this register are changed using the LDSR instruction, the new contents become valid immediately following completion of the LDSR instruction execution. However, if the ID flag is set to 1, interrupt request acknowledgement during LDSR instruction execution is prohibited.

Bits 31 to 8 are reserved (fixed to 0) for future function expansion.

Figure 3-6: Program Status Word (PSW) Format



Bit position	Bit name	Description
31 to 8	RFU	Reserved field. Fixed to 0.
7	NP	Indicates that non-maskable interrupt (NMI) servicing is in progress. This flag is set to 1 when an NMI request is acknowledged, and disables maskable interrupts. 0: NMI servicing not in progress 1: NMI servicing in progress
6	EP	Indicates that exception processing is in progress. This flag is set to 1 when an exception occurs. Moreover, interrupt requests can be acknowledged even when this bit is set. 0: Exception processing not in progress 1: Exception processing in progress
5	ID	Indicates whether maskable interrupt request acknowledgment is enabled. 0: Interrupt enabled 1: Interrupt disabled
4	SAT**Note**	Indicates that the result of executing a saturated operation instruction has overflowed and that the calculation result is saturated. Since this is a cumulative flag, it is set to 1 when the result of a saturated operation instruction becomes saturated, and it is not cleared to 0 even if the operation results of successive instructions do not become saturated. This flag is neither set nor cleared when arithmetic operation instructions are executed. 0: Not saturated 1: Saturated
3	CY	Indicates whether carry or borrow occurred as the result of an operation. 0: No carry or borrow occurred 1: Carry or borrow occurred
2	OV**Note**	Indicates whether overflow occurred during an operation. 0: No overflow occurred 1: Overflow occurred.
1	S**Note**	Indicates whether the result of an operation is negative. 0: Operation result is positive or 0. 1: Operation result is negative.
0	Z	Indicates whether operation result is 0. 0: Operation result is not 0. 1: Operation result is 0.

Note: During saturated operation, the saturated operation results are determined by the contents of the OV flag and S flag. The SAT flag is set to 1 only when the OV flag is set to 1 during saturated operation. This is explained on the following table.

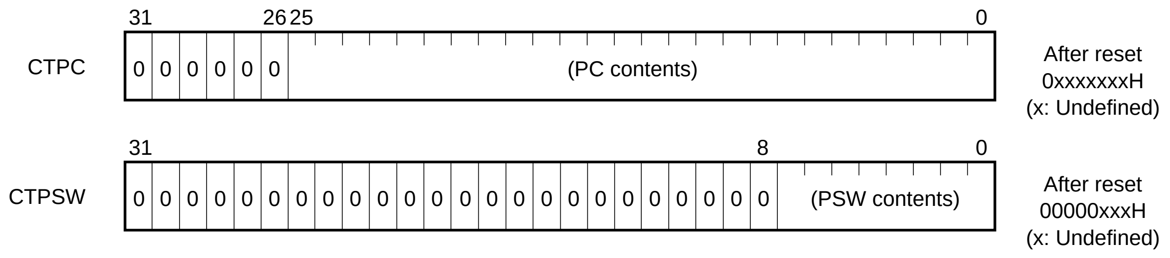
Table 3-3: Saturated Operation Results

Operation result status	Flag status			Saturated operation result
	SAT	OV	S	
Maximum positive value exceeded	1	1	0	7FFFFFFFH
Maximum negative value exceeded	1	1	1	80000000H
Positive (maximum value not exceeded)	Holds value before operation	0	0	Actual operation result
Negative (maximum value not exceeded)			1	

(5) CALLT execution context saving registers (CTPC, CTPSW)

There are two CALLT execution context saving registers, CTPC and CTPSW.
When the CALLT instruction is executed, the contents of the program counter (PC) are saved to CTPC, and the program status word (PSW) contents are saved to CTPSW.
The contents saved to CTPC consist of the address of the next instruction after the CALLT instruction.
The current PSW contents are saved to CTPSW.
Bits 31 to 26 CTPC and bits 31 to 8 of CTPSW are reserved (fixed to 0) for future function expansion.

Figure 3-7: CALLT Execution Context Saving Registers (CTPC, CTPSW) Format

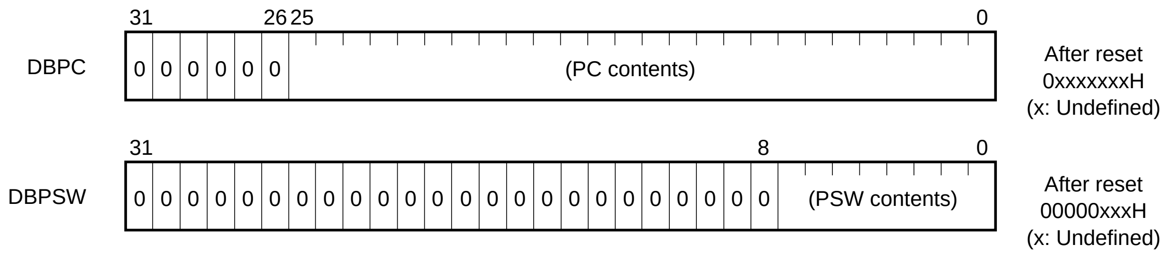


The values of CTPC and CTPSW are restored to PC and PSW during execution of the CTRET instruction.

(6) Exception/debug trap context saving registers (DBPC, DBPSW)

There are two exception/debug trap context saving registers, DBPC and DBPSW.
Upon occurrence of an exception trap or debug trap, the contents of the program counter (PC) are saved to DBPC, and the program status word (PSW) contents are saved to DBPSW.
The contents saved to DBPC consist of the address of the next instruction after the instruction executed when an exception trap or debug trap occurs.
The current PSW contents are saved to DBPSW.
Bits 31 to 26 of DBPC and bits 31 to 8 of DBPSW are reserved (fixed to 0) for future function expansion.

Figure 3-8: Exception/Debug Trap Context Saving Registers (DBPC, DBPSW) Format

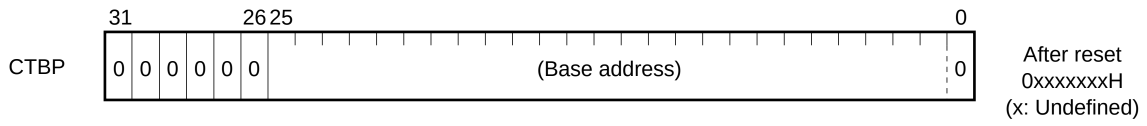


The values of DBPC and DBPSW are restored to PC and PSW during execution of the DBRET instruction.

(7) CALLT base pointer (CTBP)

The CALLT base pointer (CTBP) is used to specify CALLT table start address and generate target addresses (bit 0 is fixed to 0).
Bits 31 to 26 are reserved (fixed to 0) for future function expansion.

Figure 3-9: CALLT Base Pointer (CTBP) Format



3.2.3 Special registers

Special registers are registers that prevent invalid data from being written when an inadvertent program behaviour occurs.
The V850E/CA4 has the following two special registers.

- Power save control register (PSC)
- Processor clock control register (CKC)

Moreover, they are also two command registers (PRCMD and PHCMD), which are protection registers for write operations to the special registers. Write access to the special registers is performed with a special sequence and illegal store operations are notified to the system status register (PHS).

Writing to PSC is controlled by the PRCMD register, and writing to CKC is controlled by the PHCMD register. Occurrences of illegal writes to PSC and CKC can be checked in the PHS register.

(1) Setting data to special registers

Setting data to a special registers is done with the sequence detailed below.

(2) Command register (PRCMD, PHCMD)

The PRCMD and PHCMD registers are 8-bit registers used to prevent data from being written to registers that may have a large influence on the system, possibly causing the application system to unexpectedly stop. Only the first write operation to the special register (PSC or CKC) following the execution of a previously executed write operation to the PRCMD or PHCMD register, is valid. As a result, register values can be overwritten only using a preset sequence, preventing invalid write operations.

PRCMD and PHCMD protect special registers from an inadvertent write access. PRCMD and PHCMD registers must be written with store instruction execution by CPU only (not with DMA transfer). If an illegal store operation to a special register takes place, it can be checked by the PERR flag of the system status register (PHS).

This register can be written in 8-bit units only. Undefined data is read from this register.

Figure 3-10: Processor Command Register Format

	7	6	5	4	3	2	1	0	Address	After RESET
PRCMD	8 bits Registration Code								FFFF F1FCH	xxH
	W	W	W	W	W	W	W	W		

	7	6	5	4	3	2	1	0	Address	After RESET
PHCMD	8 bits Registration Code								FFFF F800H	xxH
	W	W	W	W	W	W	W	W		

Remark: Registration Code is any 8-bit data.

Note: PRCMD, PHCMD, PSC, CKC registers must be written with store instruction execution by CPU only. If an illegal store operation to a special register takes place, it can be checked by the PERR flag of the system status register (PHS).

(3) System status register (PHS)

The PHS register is an 8-bit register to which the PERR flag showing the generation of protection errors is assigned.

If a write operation to a special register has not been executed in the correct sequence including the access to the command register (PRCMD or PHCMD), the write operation to the planned register is not executed, a protection error is generated and the PERR flag is set to 1. The value of this register becomes “00H” by $\overline{\text{RESET}}$ input.

This register can be read/written in 8-bit and 1-bit units.

Figure 3-11: System Status Register Format

	7	6	5	4	3	2	1	0	Address	After RESET
PHS	0	0	0	0	0	0	0	PERR	FFFF F802H	00H
	R	R	R	R	R	R	R	R/W		

Operation conditions of PERR flag

- Set condition:

  Attempted write operation to a special register without previously write access to PRCMD or PHCMD register.
  If the first store instruction operation after data has been written to the PRCMD or PHCMD register is to memory or peripheral I/Os other than a special register.

- Reset condition:

  When “0” is written to the PERR flag of the PHS register.
  On system reset.

Cautions: 1. If 0 is written to the PERR bit of the PHS register (that is not a special register) immediately following write to the PRCMD or PHCMD register, the PERR bit becomes 0 (write priority).

2. If data is written to the PRCMD or PHCMD register (that are not special registers) immediately following write to the PRCMD or PHCMD register, the PERR bit becomes 1.

(4) Examples for setting data in a special register

Data setting in the PSC register follows the following sequence:

<1> Prepare data that shall be set in the PSC register in any optional general register
<2> The contents of the general register prepared in <1> is written to the command register (PRCMD) using the store instruction (ST/SST).
<3> The contents of the general register prepared in <1> is written to the PSC register with the next instruction (has to be executed immediately after write to PRCMD register).
- Store Instruction (ST/SST instruction)
- Bit Operation Instruction (SET1/CLR1/NOT1 instruction)

<4> In case of a move to software STOP mode, at least 5 NOP instructions should be inserted.

Example 1

```
<1> mov     0x02,r11
    movea   base_address, r0, r20;     base_address = FFFF000H
<2> st.b    r11, PRCMD[r20];           PRCMD = 01FCH
<3> st.b    r11, PSC[r20];             PSC = 01FEH
<4> nop
    nop
    nop
    nop
    nop
```

Example 2

```
<1> mov     0x02,r11
    movea   0xF1FCH,r0,r20
    movea   0xF1FEH,r0,r21
<2> st.b    r11,0x0[r20];          r20 = FFFFF1FCH (= PRCMD)
<3> st.b    r11,0x0[r21];          r21 = FFFFF1FEH (= PSC)
<4> nop
    nop
    nop
    nop
    nop
```

Caution: Interrupts are not acknowledged when executing the store instruction to the PRCMD or PHCMD register.
If another instruction is placed between step <2> and <3>, the correct sequence may not be realized when an interrupt is acknowledged for that instruction, resulting in the writing to the protected register to be not done, and an error to be stored in the PERR bit of the PHS register.

3.3 Operation Modes

The V850E/CA4 has the following operating modes.

(1) Normal operating mode

After the system has been released from the reset state, the pins related to the bus interface are set to the port mode, execution branches to the reset entry address of the internal ROM, and instruction processing is started.

(2) Flash memory programming mode

The internal flash memory can be written or erased when 10 V ± 0.3 V is applied to the V_{PP} pin.

V_{PP}	Operating Mode
0 V	Normal operation mode
7,8 V	Flash memory programming mode
Other than above	Setting prohibited

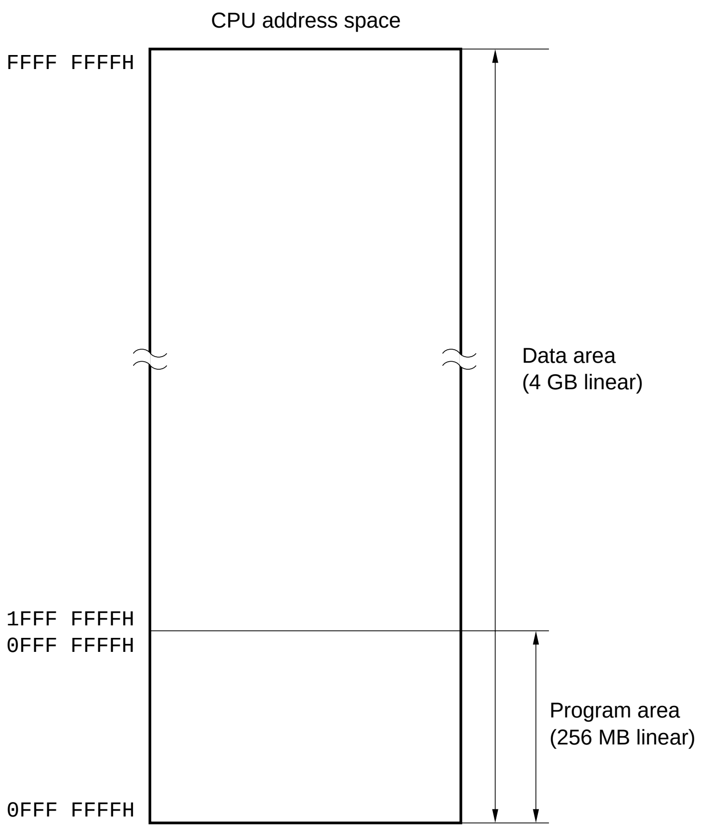
3.4 Address Space

3.4.1 CPU address space

The CPU of the V850E/CA4 uses a 32-bit architecture and supports up to 4 GB of linear address space (data space) during operand addressing (data access). When addressing instruction, a linear address space (program space) of up to 256 MB is supported. However, both the program and data spaces include areas whose use is prohibited.
For details, refer to Figure 3-13, "Address Space Image," on page 67.

Figure 3-12 shows the CPU address space.

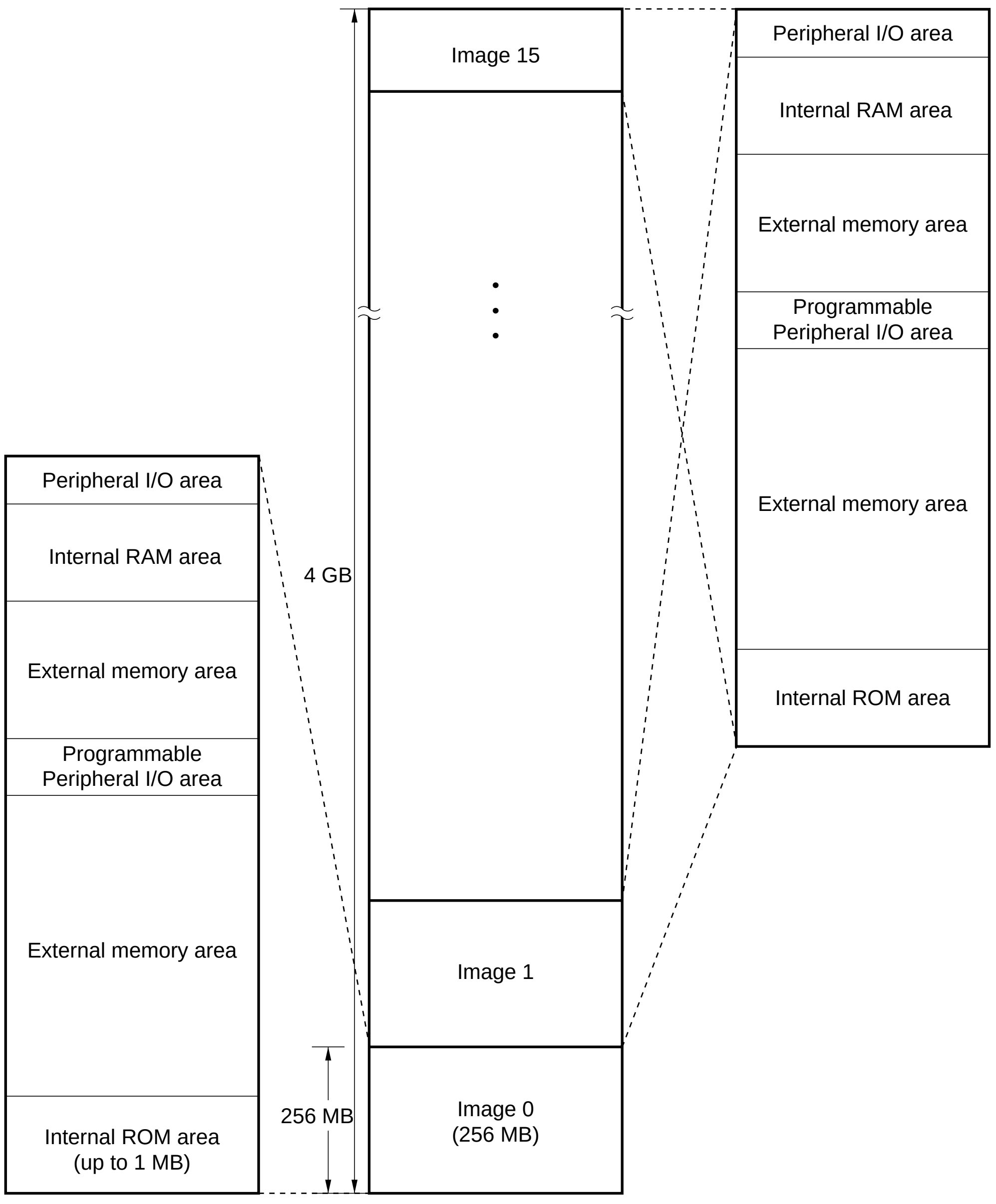
Figure 3-12: CPU Address Space



3.4.2 Image

When addressing an instruction address, up to 64 MB of linear address space (program space) and Internal RAM area are supported.
For operand addressing (data access), up to 4 GB of linear address space (data area) is supported. On this 4 GB address space, however, 256 MB physical address spaces can be seen as an image.
Therefore, whatever the values of bits 31 to 29 of an address may be, a physical address space of the same 256 MB is accessed.

Figure 3-13: Address Space Image



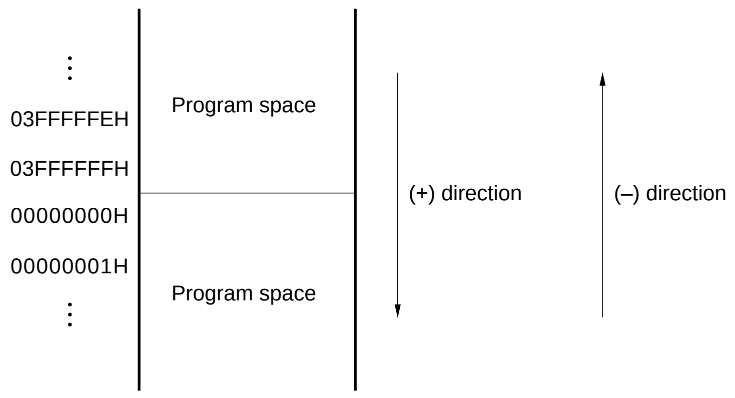
3.4.3 Wrap-around of CPU address space

(1) Program space

Of the 32 bits of the program counter (PC), the higher 6 bits are fixed to 0 and only the lower 26 bits are valid. Even if a carry or borrow occurs from bit 25 to bit 26 as a result of branch address calculation, the higher 6 bits ignore this and remain 0.
Therefore, the lower-limit address of the program space, 00000000H, and the upper-limit address, 03FFFFFFH, are contiguous addresses, and the program space is wrapped around at the boundary of these addresses.

Caution: No instructions can be fetched from the 4 KB area of 03FFF000H to 03FFFFFFH because this area is a peripheral I/O area. Therefore, do not execute any branch operation instructions in which the destination address will reside in any part of this area.

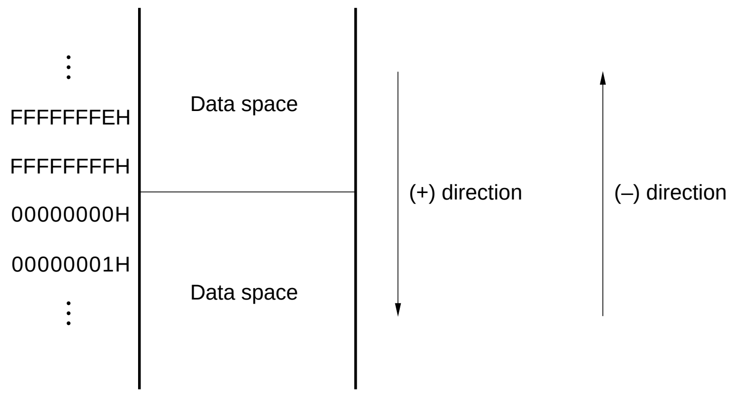
Figure 3-14: Program Space



(2) Data space

The result of an operand address calculation that exceeds 32 bits is truncated to 32 bits.
Therefore, the lower-limit address of the data space, address 00000000H, and the upper-limit address, FFFFFFFFH, are contiguous addresses, and the data space is wrapped around at the boundary of these addresses.

Figure 3-15: Data Space



3.4.4 Memory map

Memory map for 256 KB flash and mask version (µPD70F3175, µPD703175) is shown in Figure 3-16.
Memory map for 192 KB ROM version (µPD703176) is shown in Figure 3-17.
Memory map for 128 KB ROM version (µPD703174) is shown in Figure 3-18.

Figure 3-16: Memory Map for 256 KB Flash Version (µPD70F3175, µPD703175)

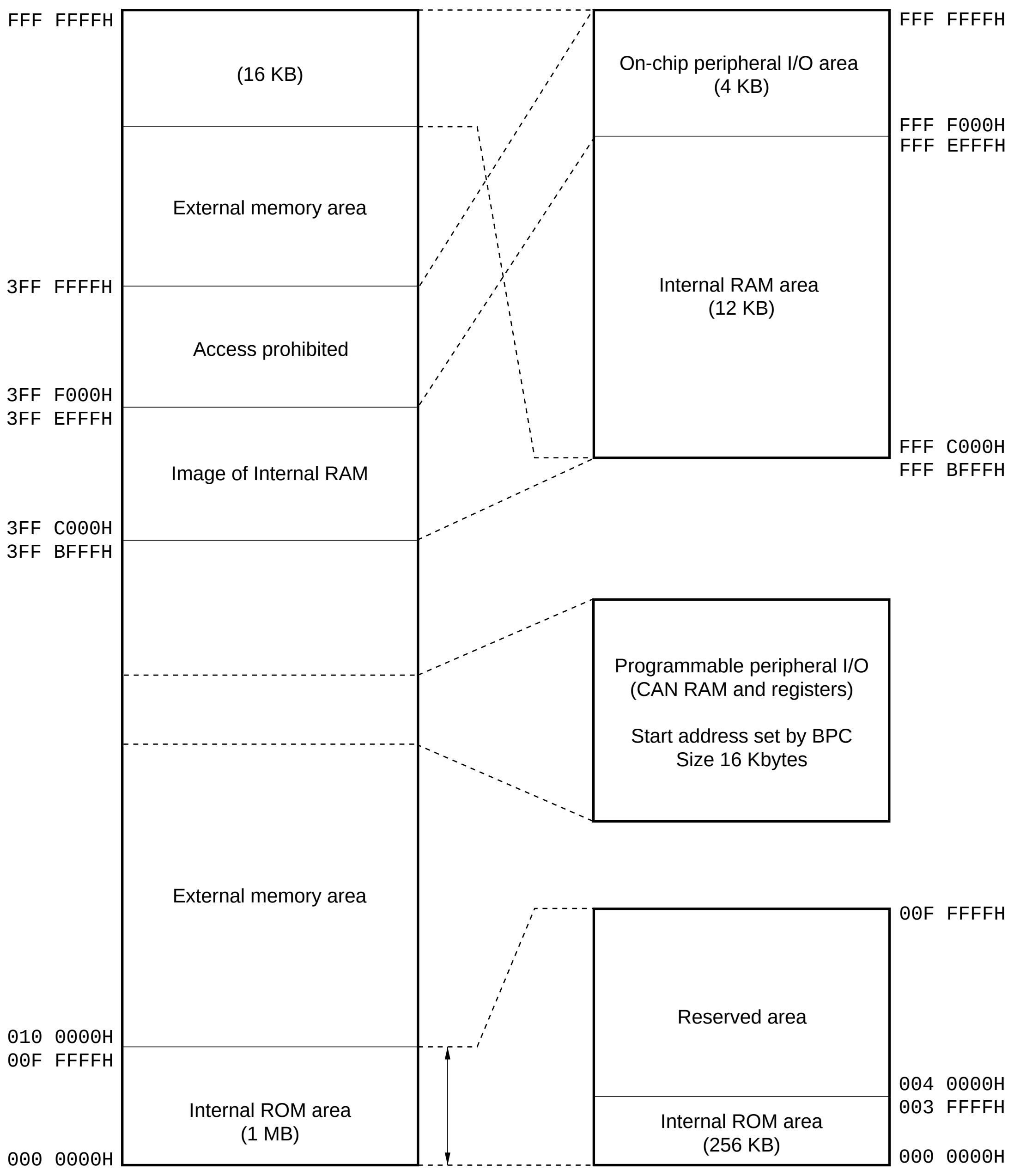


Figure 3-17: Memory Map for 192 KB ROM Version (µPD703176)

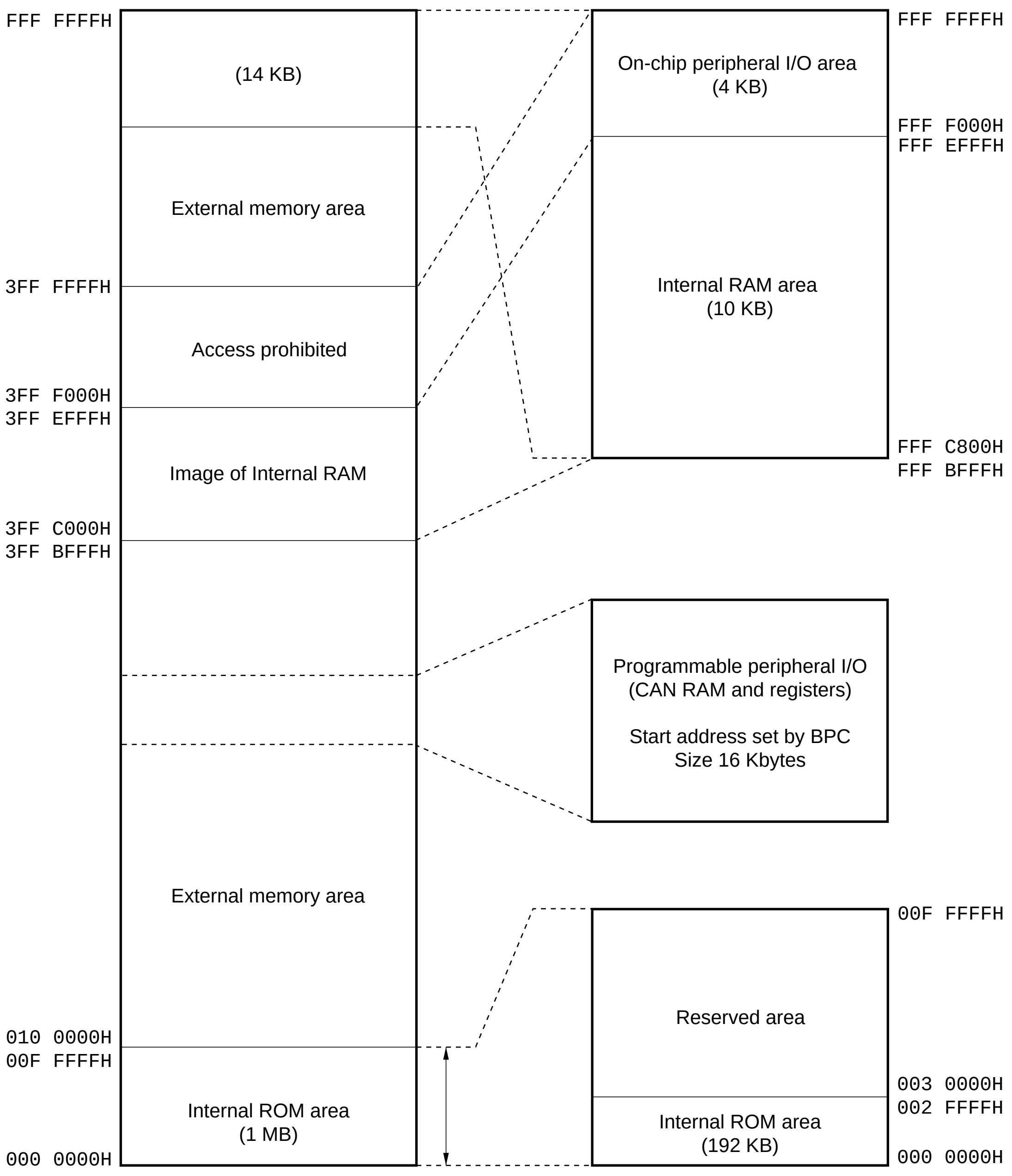
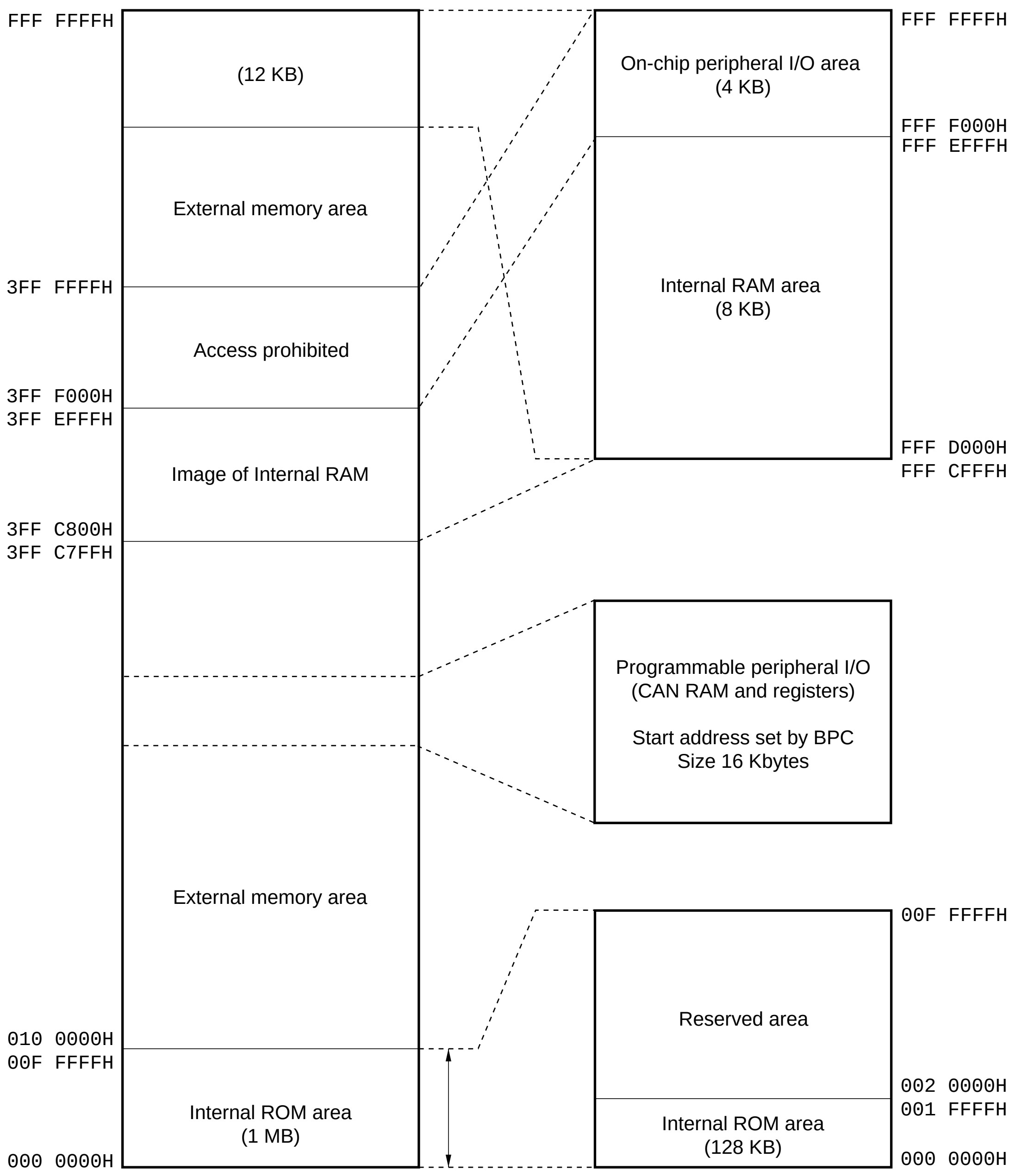


Figure 3-18: Memory Map for 128 KB ROM Version (µPD703174)


FFF FFFFH
(12 KB)
External memory area
3FF FFFFH
Access prohibited
3FF F000H
3FF EFFFH
Image of Internal RAM
3FF C800H
3FF C7FFH
External memory area
010 0000H
00F FFFFH
Internal ROM area
(1 MB)
000 0000H
FFF FFFFH
On-chip peripheral I/O area
(4 KB)
FFF F000H
FFF EFFFH
Internal RAM area
(8 KB)
FFF D000H
FFF CFFFH
Programmable peripheral I/O
(CAN RAM and registers)
Start address set by BPC
Size 16 Kbytes
00F FFFFH
Reserved area
002 0000H
001 FFFFH
Internal ROM area
(128 KB)
000 0000H

3.4.5 Areas

(1) Internal ROM area

An area of 1 MB from 0000000H to 00FFFFFH is reserved for the internal ROM area.

Interrupt/exception table

The V850E/CA4 increases the interrupt response speed by assigning handler addresses corresponding to each interrupt/exception.
This group of handler addresses is called an interrupt/exception table. This table is located in the internal ROM area. When an interrupt/exception request is acknowledged, execution jumps to the handler address and the program written in that memory is executed.
For detailed list of the interrupt/exception sources and the corresponding handler addresses, please refer to Chapter 5 ”Interrupt/Exception Processing Function” on page 109,
Table 5-1: ”Interrupt Source List” on page 109.

(2) Internal RAM area

An area of 12 KB maximum from FFF C000H to FFF EFFFH is reserved for the internal RAM area.

The same contents are seen at address ranges FFF C000H to FFF EFFFH and 3FF C000H to 3FF EFFFH in the µPD70F3175 (12 KB).

The same contents are seen at address ranges FFF C800H to FFF EFFFH and 3FF C800H to 3FF EFFFH in the µPD703176 (10 KB).

The same contents are seen at address ranges FFF D000H to FFF EFFFH and 3FF D000H to 3FF EFFFH in the µPD703174 (8 KB).

(3) On-chip peripheral I/O area (SFR area)

A 4 KB area from FFF F000H to FFF FFFFH is provided as the on-chip peripheral I/O area.

Peripheral I/O registers assigned with functions such as on-chip peripheral I/O operation mode specification and state monitoring are mapped to the on-chip peripheral I/O area. Program fetches are not allowed in this area.

Cautions:
1. **If word access of a register is attempted, half-word access to the word area is performed twice, first for the lower byte, then for the higher byte, ignoring the lower 2 address bits.**
2. **If a register that can be accessed in byte units is accessed in half-word units, the higher 8 bits become undefined if the access is a read operation. If a write access is performed, only the data in the lower 8 bits is written to the register.**
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) Programmable peripheral area

The programmable peripheral area is 16 KB wide and is used to map the FCAN registers and RAM.
The first address of this area is defined by the BPC register: PA0 to PA13 set address bits A14 to A27 of the peripheral area.

BPC is a read/write 16-bit accessible register.

Figure 3-19: Programmable Peripheral Area Control Register BPC

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	At Reset
BPC	PA15	0	PA13	PA12	PA11	PA10	PA9	PA8	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	FFFF F064H	00H

PA15	Access to programmable peripheral area	BPC read Value
0	Disabled	0000H
1	Enabled	8FFBH

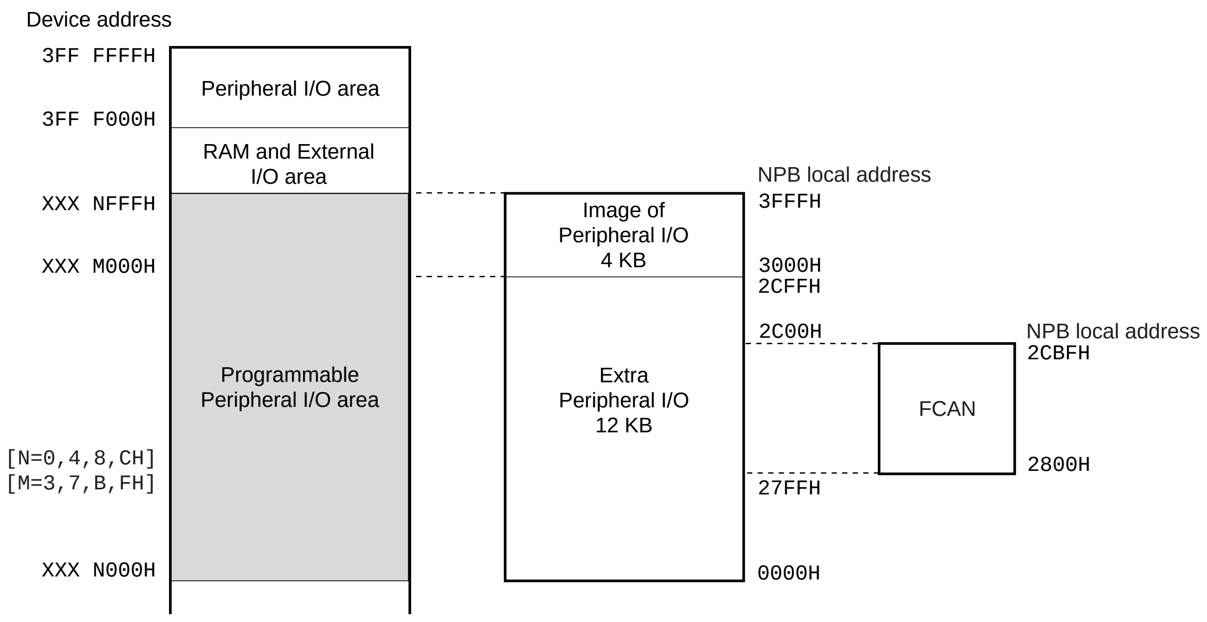
To use the CAN built-in interfaces, set PA15 to 1 by writing the BPC register with a 16-bit memory manipulation instruction.

To disable access to the CAN RAM and CAN registers, clear PA15 to 0 by writing 0000H to the BPC register with a 16-bit memory manipulation instruction.

The mapping of the CAN RAM and registers can be shown in section 3.4.8 ”FCAN RAM and registers” on page 83.

For example, if BPC = 0x8040, the programmable area is set to 0x100000.

Figure 3-20: Programmable Peripheral Area



Caution: it is recommended to locate the programmable peripheral area in the first 32 Mbytes of the physical memory.

3.4.6 Area access time

Be sure to set the VSWC register before using the V850E/CA4.

(1) Number of access clocks

The number of basic clocks necessary for accessing each resources is as follow:

Bus Cycle Type	Resource (bus width)		
	Internal ROM (32 bits)	Internal RAM (32 bits)	Peripheral I/O (16 bits)
Instruction fetch (continuous/normal mode)	1	1 or 2	Disable
Instruction fetch (branch)	2	1 0r 2	Disable
Operand data access	3	1	3 + VSWC setting

Remark: Unit: Clock / access

(2) System wait control register (VSWC)

The system wait control register (VSWC) controls the bus access wait time for the on-chip peripheral I/O registers.

This register can be read or written in 1-bit and 8-bit units.

Figure 3-21: System Wait Control Register (VSWC) Format

	7	6	5	4	3	2	1	0	Address	After RESET
VSWC	0	SUWL2	SUWL1	SUWL0	0	VSWL2	VSWL1	VSWL0	FFFF F06EH	77H
	R	R	R	R	R	R	R	R/W		

After reset, VSWC holds 77H.
For internal operation at 32 MHz, set the value 12H to VSWC.
For operation slower than 32 MHz, set the value 11H to VSWC for faster operation.

System clock	Setup wait	Strobe wait	VSWC value
FCPU = 32 MHZ	1	2	12H
FCPU < 32 MHz	1	1	11H

3.4.7 Peripheral I/O registers

Table 3-4: Peripheral I/O Registers (1/7)

Address	Special Function Register Name	Symbol	R/W	Accessibility			After Reset
				1-bit	8-bit	16-bit	
FFFFF004	Port DL register	PDL	R/W	×		×	undefined
FFFFF004	Port DL register L	PDLL	R/W	×	×		undefined
FFFFF005	Port DL register H	PDLH	R/W	×	×		undefined
FFFFF006	Port DH register	PDH	R/W	×		×	undefined
FFFFF006	Port DH register L	PDHL	R/W	×	×		undefined
FFFFF007	Port DH register H	PDHH	R/W	×	×		undefined
FFFFF00A	Port CT register	PCT	R/W	×	×		undefined
FFFFF00C	Port CM register	PCM	R/W	×	×		undefined
FFFFF00E	Port CD register	PCD	R/W	×	×		undefined
FFFFF024	Port DL mode register	PMDL	R/W	×		×	FFFFH
FFFFF024	Port DL mode register L	PMDLL	R/W	×	×		FFH
FFFFF025	Port DL mode register H	PMDLH	R/W	×	×		FFH
FFFFF026	Port DH mode register	PMDH	R/W	×		×	FFFFH
FFFFF026	Port DH mode register L	PMDHL	R/W	×	×		FFH
FFFFF027	Port DH mode register H	PMDHH	R/W	×	×		FFH
FFFFF02A	Port CT mode register	PMCT	R/W	×	×		53H
FFFFF02C	Port CM mode register	PMCM	R/W	×	×		0FH
FFFFF02E	Port CD mode register	PMCD	R/W	×	×		0CH
FFFFF064	Programmable Peripheral Area control register	BPC	R/W			×	0000H
FFFFF066	Bus size configuration register	BSC	R/W			×	5555H
FFFFF06E	System wait control register	VSWC	R/W	×	×		77H
FFFFF080	DMA source address register 0L	DSA0L	R/W			×	undefined
FFFFF082	DMA source address register 0H	DSA0H	R/W			×	undefined
FFFFF084	DMA destination address register 0L	DDA0L	R/W			×	undefined
FFFFF086	DMA destination address register 0H	DDA0H	R/W			×	undefined
FFFFF088	DMA source address register 1L	DSA1L	R/W			×	undefined
FFFFF08A	DMA source address register 1H	DSA1H	R/W			×	undefined
FFFFF08C	DMA destination address register 1L	DDA1L	R/W			×	undefined
FFFFF08E	DMA destination address register 1H	DDA1H	R/W			×	undefined
FFFFF090	DMA source address register 2L	DSA2L	R/W			×	undefined
FFFFF092	DMA source address register 2H	DSA2H	R/W			×	undefined
FFFFF094	DMA destination address register 2L	DDA2L	R/W			×	undefined
FFFFF096	DMA destination address register 2H	DDA2H	R/W			×	undefined
FFFFF098	DMA source address register 3L	DSA3L	R/W			×	undefined
FFFFF09A	DMA source address register 3H	DSA3H	R/W			×	undefined
FFFFF09C	DMA destination address register 3L	DDA3L	R/W			×	undefined
FFFFF09E	DMA destination address register 3H	DDA3H	R/W			×	undefined
FFFFF0C0	DMA transfer count register 0	DBC0	R/W			×	undefined

Table 3-4: Peripheral I/O Registers (2/7)

Address	Special Function Register Name	Symbol	R/W	Accessibility			After Reset
				1-bit	8-bit	16-bit	
FFFFF0C2	DMA transfer count register 1	DBC1	R/W			×	undefined
FFFFF0C4	DMA transfer count register 2	DBC2	R/W			×	undefined
FFFFF0C6	DMA transfer count register 3	DBC3	R/W			×	undefined
FFFFF0D0	DMA addressing control register 0	DADC0	R/W			×	undefined
FFFFF0D2	DMA addressing control register 1	DADC1	R/W			×	undefined
FFFFF0D4	DMA addressing control register 2	DADC2	R/W			×	undefined
FFFFF0D6	DMA addressing control register 3	DADC3	R/W			×	undefined
FFFFF0E0	DMA channel control register 0	DCHC0	R/W	×	×		00H
FFFFF0E2	DMA channel control register 1	DCHC1	R/W	×	×		00H
FFFFF0E4	DMA channel control register 2	DCHC2	R/W	×	×		00H
FFFFF0E6	DMA channel control register 3	DCHC3	R/W	×	×		00H
FFFFF0F0	DMA disable status register	DDIS	R/W	×	×		00H
FFFFF0F2	DMA restart register	DRST	R/W	×	×		00H
FFFFF100	Interrupt mask register 0L	IMR0	R/W			×	FFFFH
FFFFF100	Interrupt mask register 0L	IMR0L	R/W	×	×		FFH
FFFFF101	Interrupt mask register 0H	IMR0H	R/W	×	×		FFH
FFFFF102	Interrupt mask register 1	IMR1	R/W			×	FFFFH
FFFFF102	Interrupt mask register 1L	IMR1L	R/W	×	×		FFH
FFFFF103	Interrupt mask register 1H	IMR1H	R/W	×	×		FFH
FFFFF104	Interrupt mask register 2	IMR2	R/W			×	FFFFH
FFFFF104	Interrupt mask register 2L	IMR2L	R/W	×	×		FFH
FFFFF105	Interrupt mask register 2H	IMR2H	R/W	×	×		FFH
FFFFF106	Interrupt mask register 3	IMR3	R/W			×	FFFFH
FFFFF106	Interrupt mask register 3L	IMR3L	R/W	×	×		FFH
FFFFF107	Interrupt mask register 3H	IMR3H	R/W	×	×		FFH
FFFFF110	Interrupt control register	P00IC	R/W	×	×		47H
FFFFF112	Interrupt control register	P01IC	R/W	×	×		47H
FFFFF114	Interrupt control register	P02IC	R/W	×	×		47H
FFFFF116	Interrupt control register	DETIC	R/W	×	×		47H
FFFFF118	Interrupt control register	WTIC	R/W	×	×		47H
FFFFF11A	Interrupt control register	TMG00IC	R/W	×	×		47H
FFFFF11C	Interrupt control register	TMG01IC	R/W	×	×		47H
FFFFF11E	Interrupt control register	CCG00IC	R/W	×	×		47H
FFFFF120	Interrupt control register	CCG01IC	R/W	×	×		47H
FFFFF122	Interrupt control register	CCG02IC	R/W	×	×		47H
FFFFF124	Interrupt control register	CCG03IC	R/W	×	×		47H
FFFFF126	Interrupt control register	CCG04IC	R/W	×	×		47H
FFFFF128	Interrupt control register	CCG05IC	R/W	×	×		47H
FFFFF12A	Interrupt control register	TMG10IC	R/W	×	×		47H
FFFFF12C	Interrupt control register	TMG11IC	R/W	×	×		47H

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

Address	Special Function Register Name	Symbol	R/W	Accessibility			After Reset
				1-bit	8-bit	16-bit	
FFFFF12E	Interrupt control register	CCG10IC	R/W	×	×		47H
FFFFF130	Interrupt control register	CCG11IC	R/W	×	×		47H
FFFFF132	Interrupt control register	CCG12IC	R/W	×	×		47H
FFFFF134	Interrupt control register	CCG13IC	R/W	×	×		47H
FFFFF136	Interrupt control register	CCG14IC	R/W	×	×		47H
FFFFF138	Interrupt control register	CCG15IC	R/W	×	×		47H
FFFFF13A	Interrupt control register	CMD0IC	R/W	×	×		47H
FFFFF13C	Interrupt control register	CMD01IC	R/W	×	×		47H
FFFFF13E	Interrupt control register	WDTMIC	R/W	×	×		47H
FFFFF140	Interrupt control register	DMA0IC	R/W	×	×		47H
FFFFF142	Interrupt control register	DMA1IC	R/W	×	×		47H
FFFFF144	Interrupt control register	DMA2IC	R/W	×	×		47H
FFFFF146	Interrupt control register	DMA3IC	R/W	×	×		47H
FFFFF148	Interrupt control register	FC1RXIC	R/W	×	×		47H
FFFFF14A	Interrupt control register	FC1TXIC	R/W	×	×		47H
FFFFF14C	Interrupt control register	FC1ERIC	R/W	×	×		47H
FFFFF14E	Interrupt control register	MACIC	R/W	×	×		47H
FFFFF150	Interrupt control register	CSI00IC	R/W	×	×		47H
FFFFF152	Interrupt control register	CSI01IC	R/W	×	×		47H
FFFFF154	Interrupt control register	SR60IC	R/W	×	×		47H
FFFFF156	Interrupt control register	ST60IC	R/W	×	×		47H
FFFFF158	Interrupt control register	SRE60IC	R/W	×	×		47H
FFFFF15A	Interrupt control register	SR61IC	R/W	×	×		47H
FFFFF15C	Interrupt control register	ST61IC	R/W	×	×		47H
FFFFF15E	Interrupt control register	SRE61IC	R/W	×	×		47H
FFFFF160	Interrupt control register	ADIC	R/W	×	×		47H
FFFFF162	Interrupt control register	FC2RXIC	R/W	×	×		47H
FFFFF164	Interrupt control register	FC2TXIC	R/W	×	×		47H
FFFFF166	Interrupt control register	FC2ERIC	R/W	×	×		47H
FFFFF168	Interrupt control register	P10IC	R/W	×	×		47H
FFFFF16A	Interrupt control register	P15IC	R/W	×	×		47H
FFFFF16C	Interrupt control register	P20IC	R/W	×	×		47H
FFFFF16E	Interrupt control register	P25IC	R/W	×	×		47H
FFFFF170	Interrupt control register	P30IC	R/W	×	×		47H
FFFFF172	Interrupt control register	P32IC	R/W	×	×		47H
FFFFF174	Interrupt control register	P34IC	R/W	×	×		47H
FFFFF176	Interrupt control register	WTIIC	R/W	×	×		47H
FFFFF178	Interrupt control register	CSI10IC	R/W	×	×		47H
FFFFF17A	Interrupt control register	DOVFIC	R/W	×	×		47H
FFFFF1FA	In service priority register	ISPR	R	×	×		00H

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

Address	Special Function Register Name	Symbol	R/W	Accessibility			After Reset
				1-bit	8-bit	16-bit	
FFFFF1FC	Command register	PRCMD	W		×		undefined
FFFFF1FE	Power save control register	PSC	R/W	×	×		00H
FFFFF200	A/D scan mode register 0	ADSCM0	R/W			×	0000H
FFFFF200	A/D scan mode register 0L	ADSCM0L	R/W	×	×		00H
FFFFF201	A/D scan mode register 0H	ADSCM0H	R/W	×	×		00H
FFFFF202	A/D scan mode register 1	ADSCM1	R/W	×	×		00H
FFFFF204	A/D Voltage detect mode register	ADETM	R/W			×	0000H
FFFFF204	A/D Voltage detect mode register L	ADETML	R/W	×	×		00H
FFFFF205	A/D Voltage detect mode register H	ADETMH	R/W	×	×		00H
FFFFF210	A/D conversion result register 0	ADCR0	R			×	undefined
FFFFF212	A/D conversion result register 1	ADCR1	R			×	undefined
FFFFF214	A/D conversion result register 2	ADCR2	R			×	undefined
FFFFF216	A/D conversion result register 3	ADCR3	R			×	undefined
FFFFF218	A/D conversion result register 4	ADCR4	R			×	undefined
FFFFF21A	A/D conversion result register 5	ADCR5	R			×	undefined
FFFFF21C	A/D conversion result register 6	ADCR6	R			×	undefined
FFFFF21E	A/D conversion result register 7	ADCR7	R			×	undefined
FFFFF220	A/D conversion result register 8	ADCR8	R			×	undefined
FFFFF222	A/D conversion result register 9	ADCR9	R			×	undefined
FFFFF224	A/D conversion result register 10	ADCR10	R			×	undefined
FFFFF226	A/D conversion result register 11	ADCR11	R			×	undefined
FFFFF228	A/D conversion result register 12	ADCR12	R			×	undefined
FFFFF22A	A/D conversion result register 13	ADCR13	R			×	undefined
FFFFF400	Port 0 register	P0	R/W	×	×		undefined
FFFFF402	Port 1 register	P1	R/W	×	×		undefined
FFFFF404	Port 2 register	P2	R/W	×	×		undefined
FFFFF406	Port 3 register	P3	R/W	×	×		undefined
FFFFF408	Port 4 register	P4	R/W	×	×		undefined
FFFFF40E	Port 7 register	P7	R			×	undefined
FFFFF40E	Port 7 register L	P7L	R	×	×		undefined
FFFFF40F	Port 7 register H	P7H	R	×	×		undefined
FFFFF420	Port 0 mode register	PM0	R/W	×	×		07H
FFFFF422	Port 1 mode register	PM1	R/W	×	×		3FH
FFFFF424	Port 2 mode register	PM2	R/W	×	×		3FH
FFFFF426	Port 3 mode register	PM3	R/W	×	×		3FH
FFFFF428	Port 4 mode register	PM4	R/W	×	×		3FH
FFFFF440	Port 0 mode control register	PMC0	R/W	×	×		00H
FFFFF442	Port 1 mode control register	PMC1	R/W	×	×		00H
FFFFF444	Port 2 mode control register	PMC2	R/W	×	×		00H
FFFFF446	Port 3 mode control register	PMC3	R/W	×	×		00H

Table 3-4: Peripheral I/O Registers (5/7)

Address	Special Function Register Name	Symbol	R/W	Accessibility			After Reset
				1-bit	8-bit	16-bit	
FFFFF448	Port 4 mode control register	PMC4	R/W	×	×		00H
FFFFF44E	Port 7 mode control register	PMC7	R/W	×	×		00H
FFFFF540	Timer counter D0	TMD0	R			×	0000H
FFFFF542	Compare register D0	CMD0	R/W			×	0000H
FFFFF544	Timer D control register 0	TMCD0	R/W	×	×		00H
FFFFF550	Timer counter D1	TMD1	R			×	0000H
FFFFF552	Compare register D1	CMD1	R/W			×	0000H
FFFFF554	Timer D control register 1	TMCD1	R/W	×	×		00H
FFFFF580	Timer G0 mode register	TMGM0	R/W			×	0000H
FFFFF580	Timer G0 mode register L	TMGM0L	R/W	×	×		00H
FFFFF581	Timer G0 mode register H	TMGM0H	R/W	×	×		00H
FFFFF582	Timer G0 channel mode register	TMGCM0	R/W			×	0000H
FFFFF582	Timer G0 channel mode register L	TMGCM0L	R/W	×	×		00H
FFFFF583	Timer G0 channel mode register H	TMGCM0H	R/W	×	×		00H
FFFFF584	Timer G0 output control register	OCTLG0	R/W			×	4444H
FFFFF584	Timer G0 output control register L	OCTLG0L	R/W	×	×		44H
FFFFF585	Timer G0 output control register H	OCTLG0H	R/W	×	×		44H
FFFFF586	Timer G0 status register	TMGST0	R	×	×		00H
FFFFF588	Timer G0 Count Register 0	TMG00	R			×	0000H
FFFFF58A	Timer G0 Count Register 1	TMG01	R			×	0000H
FFFFF58C	Timer G0 Capture/Compare Register 0	GCC00	R/W			×	0000H
FFFFF58E	Timer G0 Capture/Compare Register 1	GCC01	R/W			×	0000H
FFFFF590	Timer G0 Capture/Compare Register 2	GCC02	R/W			×	0000H
FFFFF592	Timer G0 Capture/Compare Register 3	GCC03	R/W			×	0000H
FFFFF594	Timer G0 Capture/Compare Register 4	GCC04	R/W			×	0000H
FFFFF596	Timer G0 Capture/Compare Register 5	GCC05	R/W			×	0000H
FFFFF600	Timer G1 mode register	TMGM1	R/W			×	0000H
FFFFF600	Timer G1 mode register L	TMGM1L	R/W	×	×		00H
FFFFF601	Timer G1 mode register H	TMGM1H	R/W	×	×		00H
FFFFF602	Timer G1 channel mode register	TMGCM1	R/W			×	0000H
FFFFF602	Timer G1 channel mode register L	TMGCM1L	R/W	×	×		00H
FFFFF603	Timer G1 channel mode register H	TMGCM1H	R/W	×	×		00H
FFFFF604	Timer G1 output control register	OCTLG1	R/W			×	4444H
FFFFF604	Timer G1 output control register L	OCTLG1L	R/W	×	×		44H
FFFFF605	Timer G1 output control register H	OCTLG1H	R/W	×	×		44H
FFFFF606	Timer G1 status register	TMGST1	R	×	×		00H
FFFFF608	Timer G1 Count Register 0	TMG10	R			×	0000H
FFFFF60A	Timer G1 Count Register 1	TMG11	R			×	0000H
FFFFF60C	Timer G1 Capture/Compare Register 0	GCC10	R/W			×	0000H
FFFFF60E	Timer G1 Capture/Compare Register 1	GCC11	R/W			×	0000H

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

Address	Special Function Register Name	Symbol	R/W	Accessibility			After Reset
				1-bit	8-bit	16-bit	
FFFFF610	Timer G1 Capture/Compare Register 2	GCC12	R/W			×	0000H
FFFFF612	Timer G1 Capture/Compare Register 3	GCC13	R/W			×	0000H
FFFFF614	Timer G1 Capture/Compare Register 4	GCC14	R/W			×	0000H
FFFFF616	Timer G1 Capture/Compare Register 5	GCC15	R/W			×	0000H
FFFFF680	Watchdog Timer Mode Register	WDTM	R/W	×	×		67H
FFFFF681	Watchdog Timer Enable Register	WDTE	R/W		×		9AH
FFFFF6E0	Watch Timer Mode Register	WTM	R/W	×	×		00H
FFFFF6F0	Timer Input Select Register	TIS	R/W	×	×		00H
FFFFF800	Peripheral command Register	PHCMD	W		×		undefined
FFFFF802	Peripheral Status Register	PHS	R/W	×	×		00H
FFFFF810	DMA trigger source select 0	DTFR0	R/W	×	×		00H
FFFFF812	DMA trigger source select 1	DTFR1	R/W	×	×		00H
FFFFF814	DMA trigger source select 2	DTFR2	R/W	×	×		00H
FFFFF816	DMA trigger source select 3	DTFR3	R/W	×	×		00H
FFFFF820	Power Save Mode Register	PSM	R/W	×	×		00H
FFFFF822	Processor Clock Control Register	CKC	R/W	×	×		00H
FFFFF824	PLL status register	PSTAT	R	×	×		00H
FFFFF830	Reset source monitor register	RSM	R		×		00H
FFFFF840	PCL control register	PCLCNT	R/W	×	×		00H
FFFFF842	PLL divider factor register	PDIV	R/W	×	×		00H
FFFFFA00	UART mode register 0	ASIM0	R/W	×	×		01H
FFFFFA02	UART Reception Register 0	RXB_ASIS0	R			×	FF00H
FFFFFA02	UART Receive buffer register 0	RXB0	R		×		FFH
FFFFFA03	UART status register 0	ASIS0	R		×		00H
FFFFFA04	UART Transmit buffer register 0	TXB0	R/W		×		FFH
FFFFFA05	UART transmission status register 0	ASIF0	R		×		00H
FFFFFA06	UART Clock Control register 0	CKSR_BRGC0	R/W	×		×	00FFH
FFFFFA06	UART Clock selection register 0	CKSR0	R/W		×		00H
FFFFFA07	UART Baud rate generator control register 0	BRGC0	R/W	×	×		FFH
FFFFFA08	LIN Control Register 0	ASICL0	R/W	×	×		16H
FFFFFA10	UART mode register 1	ASIM1	R/W	×	×		01H
FFFFFA12	UART Reception Register 1	RXB_ASIS1	R			×	FF00H
FFFFFA12	UART Receive buffer register 1	RXB1	R		×		FFH
FFFFFA13	UART status register 1	ASIS1	R		×		00H
FFFFFA14	UART Transmit buffer register 1	TXB1	R/W		×		FFH
FFFFFA15	UART transmission status register 1	ASIF1	R		×		00H
FFFFFA16	UART Clock Control register 1	CKSR_BRGC1	R/W	×		×	00FFH
FFFFFA16	UART Clock selection register 1	CKSR1	R/W		×		00H
FFFFFA17	UART Baud rate generator control register 1	BRGC1	R/W	×	×		FFH
FFFFFA18	LIN Control Register 1	ASICL1	R/W	×	×		16H

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

Address	Special Function Register Name	Symbol	R/W	Accessibility			After Reset
				1-bit	8-bit	16-bit	
FFFFFC00	External Interrupt Mode Register 0	INTM0	R/W	×	×		00H
FFFFFC02	External Interrupt Mode Register 1	INTM1	R/W	×	×		00H
FFFFFC04	External Interrupt Mode Register 2	INTM2	R/W	×	×		00H
FFFFFC06	External Interrupt Mode Register 3	INTM3	R/W	×	×		00H
FFFFFC08	External NMI Mode Register	NMIM	R/W		×		03H
FFFFFCA0	Baud rate generator prescaler mode register 0	PRSM0	R/W	×	×		00H
FFFFFCA1	Baud rate generator divider register 0	PRSCM0	R/W	×	×		00H
FFFFFD00	CSI 00 control register	CSIM_CSIC0	R/W			×	0000H
FFFFFD00	CSI 00 operation mode control register	CSIM0	R/W	×	×		00H
FFFFFD01	CSI 00 clock selection register	CSIC0	R/W	×	×		00H
FFFFFD02	CSI 00 receive buffer register	SIRB0	R			×	0000H
FFFFFD02	CSI 00 receive buffer register L	SIRB0L	R		×		00H
FFFFFD04	CSI 00 transmit buffer register	SOTB0	R/W			×	0000H
FFFFFD04	CSI 00 transmit buffer register L	SOTB0L	R/W		×		00H
FFFFFD08	CSI 00 first transmit buffer register	SOTBF0	R/W			×	0000H
FFFFFD08	CSI 00 first transmit buffer register L	SOTBF0L	R/W		×		00H
FFFFFD0A	CSI 00 shift register	SIO0	R			×	0000H
FFFFFD0A	CSI 00 shift register L	SIO0L	R		×		00H
FFFFFD10	CSI 01 control register	CSIM_CSIC1	R/W			×	0000H
FFFFFD10	CSI 01 operation mode control register	CSIM1	R/W	×	×		00H
FFFFFD11	CSI 01 clock selection register	CSIC1	R/W	×	×		00H
FFFFFD12	CSI 01 receive buffer register	SIRB1	R			×	0000H
FFFFFD12	CSI 01 receive buffer register L	SIRB1L	R		×		00H
FFFFFD14	CSI 01 transmit buffer register	SOTB1	R/W			×	0000H
FFFFFD14	CSI 01 transmit buffer register L	SOTB1L	R/W		×		00H
FFFFFD18	CSI 01 first transmit buffer register	SOTBF1	R/W			×	0000H
FFFFFD18	CSI 01 first transmit buffer register L	SOTBF1L	R/W		×		00H
FFFFFD1A	CSI 01 shift register	SIO1	R			×	0000H
FFFFFD1A	CSI 01 shift register L	SIO1L	R		×		00H
FFFFFD20	CSI 10 control register	CSIM10	R/W	×	×		00H
FFFFFD21	CSI 10 clock selection register	CSIC10	R/W	×	×		00H
FFFFFD22	CSI 10 shift register	SIO10	R		×		00H
FFFFFD24	CSI 10 transmit buffer register	SOTB10	R/W		×		00H

3.4.8 FCAN RAM and registers

In this table the addresses are offsets in the FCAN area, which begins at offset 2800H from the address of the programmable peripheral area defined with the BPC register (Figure 3-20, “Programmable Peripheral Area,” on page 74).

Table 3-5: CAN RAM Mapping (1/3)

Address	CAN Buffer Name	R/W	Accessibility			Reset Value
			1-bit	8-bit	16-bit	
000H	M_EVT00	R/W		×		undefined
001H	M_EVT01	R/W		×		undefined
002H	M_EVT02	-				undefined
003H	M_EVT03	R/W		×		undefined
004H	M_DLC0	R/W		×		undefined
005H	M_CTRL0	R/W		×		undefined
006H	M_TIME0	R/W			×	undefined
008H	M_DATA00	R/W	×	×		undefined
009H	M_DATA01	R/W		×		undefined
00AH	M_DATA02	R/W		×		undefined
00BH	M_DATA03	R/W		×		undefined
00CH	M_DATA04	R/W		×		undefined
00DH	M_DATA05	R/W		×		undefined
00EH	M_DATA06	R/W		×		undefined
00FH	M_DATA07	R/W		×		undefined
010H	M_IDL0	R/W			×	undefined
012H	M_IDH0	R/W			×	undefined
014H	M_CONF0	R/W		×		undefined
015H	M_STAT0	R/W		×		undefined
016H	SC_STAT0	R			×	undefined
018H	Reserved	-				undefined
019H		-				undefined
01AH		-				undefined
01BH		-				undefined
01CH		-				undefined
01DH		-				undefined
01EH		-				undefined
01FH		-				undefined
020H to 03FH	Message buffer 1 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
040H to 05FH	Message buffer 2 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
060H to 07FH	Message buffer 3 (same structure as buffer 0)	Same access rights as for buffer 0				undefined

Table 3-5: CAN RAM Mapping (2/3)

Address	CAN Buffer Name	R/W	Accessibility			Reset Value
			1-bit	8-bit	16-bit	
080H to 09FH	Message buffer 4 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
0A0H to 0BFH	Message buffer 5 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
0C0H to 0DFH	Message buffer 6 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
0E0H to 0FFH	Message buffer 7 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
100H to 11FH	Message buffer 8 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
120H to 13FH	Message buffer 9 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
140H to 15FH	Message buffer 10 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
160H to 17FH	Message buffer 11 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
180H to 19FH	Message buffer 12 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
1A0H to 1BFH	Message buffer 13 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
1C0H to 1DFH	Message buffer 14 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
1E0H to 1FFH	Message buffer 15 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
200H to 21FH	Message buffer 16 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
220H to 23FH	Message buffer 17 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
240H to 25FH	Message buffer 18 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
260H to 27FH	Message buffer 19 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
280H to 29FH	Message buffer 20 (same structure as buffer 0)	Same access rights as for buffer 0				undefined

Table 3-5: CAN RAM Mapping (3/3)

Address	CAN Buffer Name	R/W	Accessibility			Reset Value
			1-bit	8-bit	16-bit	
2A0H to 2BFH	Message buffer 21 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
2C0H to 2DFH	Message buffer 22 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
2E0H to 2FFH	Message buffer 23 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
300H to 31FH	Message buffer 24 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
320H to 33FH	Message buffer 25 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
340H to 35FH	Message buffer 26 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
360H to 37FH	Message buffer 27 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
380H to 39FH	Message buffer 28 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
3A0H to 3BFH	Message buffer 29 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
3C0H to 3DFH	Message buffer 30 (same structure as buffer 0)	Same access rights as for buffer 0				undefined
3E0H to 3FFH	Message buffer 31 (same structure as buffer 0)	Same access rights as for buffer 0				undefined

Table 3-6: CAN Registers Mapping (1/2)

Address	CAN register name	Symbol	R/W	Accessibility			Reset Value
				1-bit	8-bit	16-bit	
400H	CSTOP	CAN stop register	R/W		×	×	0000H
404H	CCINTP	CAN interrupt pending register	R		×	×	0000H
410H	CGST	CAN global status register	R	×	×	×	0000H
			W			×	
412H	CGIE	CAN global interrupt enable register	R	×	×	×	0000H
			W			×	
414H	CGCS	CAN main clock select register	R	×	×	×	7F05H
			W	×	×	×	
416H	CGTEN	CAN timer event enable register	R/W	×	×	×	0000H
418H	CGTSC	CAN global time system counter	R	×	×	×	0000H
			W			×	
41AH	CGMSS	CAN message search start register	W			×	undefined
	CGMSR	CAN message search result register	R	×	×	×	
41CH	CTBR	CAN test bus register	R/W		×	×	0000H
420H	CGINTP	CAN global interrupt pending register	R		×	×	0000H
			W			×	
422H	C1INTP	CAN1 interrupt pending register	R		×	×	0000H
			W		×	×	
424H	C2INTP	CAN2 interrupt pending register	R		×	×	0000H
			W		×	×	
440H	C1MASKL0	CAN1 mask 0 register L	R/W		×	×	undefined
442H	C1MASKH0	CAN1 mask 0 register H	R/W		×	×	undefined
444H	C1MASKL1	CAN1 mask 1 register L	R/W		×	×	undefined
446H	C1MASKH1	CAN1 mask 1 register H	R/W		×	×	undefined
448H	C1MASKL2	CAN1 mask 2 register L	R/W		×	×	undefined
44AH	C1MASKH2	CAN1 mask 2 register H	R/W		×	×	undefined
44CH	C1MASKL3	CAN1 mask 3 register L	R/W		×	×	undefined
44EH	C1MASKH3	CAN1 mask 3 register H	R/W		×	×	undefined
450H	C1CTRL	CAN1 control register	R		×	×	0101H
			W			×	
452H	C1DEF	CAN1 definition register	R		×	×	0000H
			W			×	
454H	C1LAST	CAN1 information register	R		×	×	00FFH
456H	C1ERC	CAN1 error counter register	R		×	×	0000H
458H	C1IE	CAN1 interrupt enable register	R		×	×	0000H
			W			×	
45AH	C1BA	CAN1 bus activity register	R		×	×	00FFH
			W			×	

Table 3-6: CAN Registers Mapping (2/2)

Address	CAN register name	Symbol	R/W	Accessibility			Reset Value
				1-bit	8-bit	16-bit	
45CH	C1BRP	CAN1 bit rate prescaler register	R		×	×	0000H
			W		×	×	
	C1DINF	CAN1 bus diagnostic information register	R		×	×	0000H
45EH	C1SYNC	CAN1 synchronization control register	R		×	×	0218H
			W		×	×	
480H	C2MASKL0	CAN2 mask 0 register L	R/W		×	×	
482H	C2MASKH0	CAN2 mask 0 register H	R/W		×	×	undefined
484H	C2MASKL1	CAN2 mask 1 register L	R/W		×	×	undefined
486H	C2MASKH1	CAN2 mask 1 register H	R/W		×	×	undefined
488H	C2MASKL2	CAN2 mask 2 register L	R/W		×	×	undefined
48AH	C2MASKH2	CAN2 mask 2 register H	R/W		×	×	undefined
48CH	C2MASKL3	CAN2 mask 3 register L	R/W		×	×	undefined
48EH	C2MASKH3	CAN2 mask 3 register H	R/W		×	×	undefined
490H	C2CTRL	CAN2 control register	R		×	×	undefined
			W			×	0101H
492H	C2DEF	CAN2 definition register	R		×	×	
			W			×	0000H
494H	C2LAST	CAN2 information register	R		×	×	
496H	C2ERC	CAN2 error counter register	R		×	×	00FFH
498H	C2IE	CAN2 interrupt enable register	R		×	×	0000H
			W			×	0000H
49AH	C2BA	CAN2 bus activity register	R		×	×	
			W			×	00FFH
49CH	C2BRP	CAN2 bit rate prescaler register	R		×	×	
			W		×	×	0000H
	C2DINF	CAN2 bus diagnostic information register	R		×	×	
49EH	C2SYNC	CAN2 synchronization control register	R		×	×	0000H
			W		×	×	0218H

[MEMO]

Chapter 4 DMA Functions (DMA Controller)

The V850E/CA4 includes a direct memory access (DMA) controller (DMAC) that executes and controls DMA transfer.
The DMAC controls data transfer between internal RAM memory and I/Os, based on DMA requests issued by the on-chip peripheral I/O, or software triggers.

4.1 Features

- 4 independent DMA channels
- Transfer units: 8, 16 and 32 bits
- Maximum transfer count: 65,536 (2^{16})
- One type of transfer
  - two cycle transfer
- One transfer mode
  - Single transfer mode
- Transfer requests
  - Request by interrupts from on-chip peripheral I/O
  - Requests by software trigger
- Transfer objects
  - Internal RAM $\leftrightarrow$ I/O
  - Internal RAM $\leftrightarrow$ Internal RAM
- Next address setting function

4.2 Control Registers

4.2.1 DMA source address registers high-word (DSA0H to DSA3H)

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, DSAnH and DSAnL.
Since these registers are configured as 2-stage FIFO buffer registers, a new source address for DMA transfer can be specified during DMA transfer.

(1) DMA source address registers high-word (DSA0H to DSA3H)

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 4-1: DMA Source Address Registers DSA0H to DSA3H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSA0H	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
DSA1H	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
DSA2H	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
DSA3H	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: 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 low-word (DSA0L to DSA3L)

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

Figure 4-2: DMA Source Address Registers DSA0L to DSA3L

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DSA0L	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
DSA1L	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
DSA2L	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
DSA3L	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.

4.2.2 DMA destination address registers high-word (DDA0H to DDA3H)

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, DDAnH and DDAnL.
Since these registers are configured as 2-stage FIFO buffer registers, a new destination address for DMA transfer can be specified during DMA transfer.

(1) DMA destination address registers DDA0H to DDA3H

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 4-3: DMA Destination Address Registers high-word (DDA0H to DDA3H)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDA0H	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
DDA1H	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
DDA2H	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
DDA3H	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: 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 low-word (DDA0L to DDA3L)

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

Figure 4-4: DMA Destination Address Registers low-word (DDA0L to DDA3L)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DDA0L	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
DDA1L	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
DDA2L	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
DDA3L	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.

4.2.3 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 (n = 0 to 3). 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.
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 4-5: 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.

Bit Position	Bit Name	Function
15 to 0	BC15 to BC0	Sets the transfer count. It stores the remaining transfer count during DMA transfer.

DBCn	States
0000H	Transfer count 1 or remaining transfer count
0001H	Transfer count 2 or remaining transfer count
:	:
FFFFH	Transfer count 65,536 (2^{16}) or remaining transfer count

4.2.4 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 (n = 0 to 3). These registers cannot be accessed during DMA operation.
They can be read/written in 16-bit units.

Figure 4-6: DMA Addressing Control Registers 0 to 3 (DADC0 to DADC3)

	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	0	0	0	0	FFFFF0D0H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC1	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	0	0	0	0	FFFFF0D2H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC2	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	0	0	0	0	FFFFF0D4H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
DADC3	DS1	DS0	0	0	0	0	0	0	SAD1	SAD0	DAD1	DAD0	0	0	0	0	FFFFF0D6H	0000H

Bit Position	Bit Name	Function
15, 14	DS1, DS0	Sets the transfer data size for DMA transfer. DS1 / DS0 / Transfer Data Size: 0 / 0 / 8 bits 0 / 1 / 16 bits 1 / 0 / 32 bits 1 / 1 / Setting prohibited For the peripheral I/O and programmable peripheral I/O registers, ensure the transfer size matches the access size.
7, 6	SAD1, SAD0	Sets the count direction of the source address for DMA channel n (n = 0 to 3). SAD1 / SAD0 / Count Direction: 0 / 0 / Increment 0 / 1 / Decrement 1 / 0 / Fixed 1 / 1 / Setting prohibited
5, 4	DAD1, DAD0	Sets the count direction of the destination address for DMA channel n (n = 0 to 3). DAD1 / DAD0 / Count Direction: 0 / 0 / Increment 0 / 1 / Decrement 1 / 0 / Fixed 1 / 1 / Setting prohibited

Caution: all others bits in the DADCn registers must be kept to 0.

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

These 8-bit registers are used to control the DMA transfer start, end an interruption for DMA channel n (n = 0 to 3).
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 4-7: 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. This bit is always read as 0.
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. This bit is always read as 0.
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

4.2.6 DMA disable status register (DDIS)

This register holds the contents of the ENn bit of the DCHCn register during NMI input (n = 0 to 3).
This register is read-only in 8-bit or 1-bit units.

Figure 4-8: 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.

4.2.7 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 (n = 0 to 3). 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 4-9: DMA Restart Register (DRST)

	7	6	5	4	3	2	1	0	Address	Initial value
DRST	0	0	0	0	DRSTEN3	DRSTEN2	DRSTEN1	DRSTEN0	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

4.2.8 DMA trigger factor register 0 (DTFR0)

This 8-bit registers is used to control the DMA transfer 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 trigger factors.
This register can be read/written in 8-bit/1-bit units.

Figure 4-10: DMA Trigger Factor Registers 0 (DTFR0)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR0	DRQ0	DOFL0	1	0	0	IFC02	IFC01	IFC00	FFFFF810H	00H

Bit Position	Bit Name	Function
7	DRQ0[Note]	DMA Request: 0: no DMA transfer is pending for channel 0 1: no DMA transfer is pending for channel 0
6	DOFL0[Note]	DMA request Overflow: 0: no DMA request overflow occurred 1: DMA request overflow detected
3 to 0	IFC03 to IFC00	Sets the interrupt source that serves as the DMA trigger factor.

IFC02	IFC01	IFC00	Peripheral Source
0	0	0	CSI00
0	0	1	CSI01
0	1	0	CSI10
0	1	1	UART60, Reception
1	0	0	UART61, Reception
1	0	1	ADC end of conversion
1	1	0	Timer TMG0 capture/compare 1
1	1	1	Timer TMG1 capture/compare 1

Note: DRQ0 and DOFL0 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 DRQ0 or DOFL0.

Cautions:
1. **Be sure to stop DMA operation before making changes to DTFR0 register settings.**
2. **An interrupt request input in standby mode (IDLE, WATCH or STOP mode) cannot be used as a DMA transfer start factor.**
3. **Always write 1 in bit 5 of DTFR0.**

4.2.9 DMA trigger factor register 1 (DTFR1)

This 8-bit registers is used to control the DMA transfer 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 trigger factors.
This register can be read/written in 8-bit/1-bit units.

Figure 4-11: DMA Trigger Factor Registers 1 (DTFR1)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR1	DRQ1	DOFL1	1	0	0	IFC12	IFC11	IFC10	FFFFF812H	00H

Bit Position	Bit Name	Function
7	DRQ1[Note]	DMA Request: 0: no DMA transfer is pending for channel 1 1: no DMA transfer is pending for channel 1
6	DOFL1[Note]	DMA request Overflow: 0: no DMA request overflow occurred 1: DMA request overflow detected
3 to 0	IFC13 to IFC10	Sets the interrupt source that serves as the DMA trigger factor.

IFC12	IFC11	IFC10	Peripheral Source
0	0	0	CSI00
0	0	1	CSI01
0	1	0	CSI10
0	1	1	UART60, Reception
1	0	0	UART61, Reception
1	0	1	ADC end of conversion
1	1	0	Timer TMG0 capture/compare 2
1	1	1	Timer TMD0 compare match

Note: DRQ1 and DOFL1 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 DRQ1 or DOFL1.

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

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

3. Always write 1 in bit 5 of DTFR1.

4.2.10 DMA trigger factor register 2 (DTFR2)

This 8-bit registers is used to control the DMA transfer 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 trigger factors.
This register can be read/written in 8-bit/1-bit units.

Figure 4-12: DMA Trigger Factor Registers 2 (DTFR2)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR2	DRQ2	DOFL2	1	0	0	IFC22	IFC21	IFC20	FFFFF814H	00H

Bit Position	Bit Name	Function
7	DRQ2[Note]	DMA Request: 0: no DMA transfer is pending for channel 2 1: no DMA transfer is pending for channel 2
6	DOFL2[Note]	DMA request Overflow: 0: no DMA request overflow occurred 1: DMA request overflow detected
3 to 0	IFC23 to IFC20	Sets the interrupt source that serves as the DMA trigger factor.

IFC22	IFC21	IFC20	Peripheral Source
0	0	0	CSI00
0	0	1	CSI01
0	1	0	CSI10
0	1	1	UART60, Transmission
1	0	0	UART61, Transmission
1	0	1	ADC end of conversion
1	1	0	Timer TMG0 capture/compare 3
1	1	1	Timer TMG1 capture/compare 3

Note: DRQ2 and DOFL2 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 DRQ2 or DOFL2.

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

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

3. Always write 1 in bit 5 of DTFR2.

4.2.11 DMA trigger factor register 3 (DTFR3)

This 8-bit registers is used to control the DMA transfer 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 trigger factors.
This register can be read/written in 8-bit/1-bit units.

Figure 4-13: DMA Trigger Factor Registers 3 (DTFR3)

	7	6	5	4	3	2	1	0	Address	Initial value
DTFR3	DRQ3	DOFL3	1	0	0	IFC32	IFC31	IFC30	FFFFF816H	00H

Bit Position	Bit Name	Function
7	DRQ3Note	DMA Request: 0: no DMA transfer is pending for channel 3 1: no DMA transfer is pending for channel 3
6	DOFL3Note	DMA request Overflow: 0: no DMA request overflow occurred 1: DMA request overflow detected
3 to 0	IFC33 to IFC30	Sets the interrupt source that serves as the DMA trigger factor.

IFC32	IFC31	IFC30	Peripheral Source
0	0	0	CSI00
0	0	1	CSI01
0	1	0	CSI10
0	1	1	UART60, Transmission
1	0	0	UART61, Transmission
1	0	1	ADC end of conversion
1	1	0	Timer TMG0 capture/compare 4
1	1	1	Timer TMD1 compare match

Note: DRQ3 and DOFL3 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 DRQ3 or DOFL3.

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

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

3. Always write 1 in bit 5 of DTFR3.

4.3 DMA Bus States

4.3.1 Types of bus states

The DMAC bus states consist of the following 8 states.

(1) TI state

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

(2) T0 state

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) 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.

(5) 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.

(6) 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.

(7) 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.

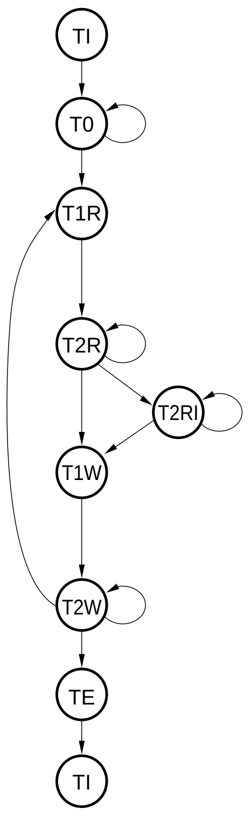
(8) TE state

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.

4.3.2 DMAC bus cycle state transition

Each time the processing for a DMA transfer is completed, the bus mastership is released.

Figure 4-14: DMAC Bus Cycle (Two-Cycle Transfer) State Transition



4.4 Transfer Mode

4.4.1 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.

4.5 Transfer Types

4.5.1 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 destination address is output and writing is performed from the DMAC to the destination.

4.6 Transfer Object

4.6.1 Transfer type and transfer object

Table 4-1 lists the relationships between transfer type and transfer object.

Table 4-1: Relationship Between Transfer Type and Transfer Object

		Destination	
		Two-Cycle Transfer	
		On-Chip Peripheral I/O	Internal RAM
Source	On-chip peripheral I/O	×	×
	Internal RAM	×	×

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.

4.7 DMA Channel Priorities

The DMA channel priorities are fixed as follows.

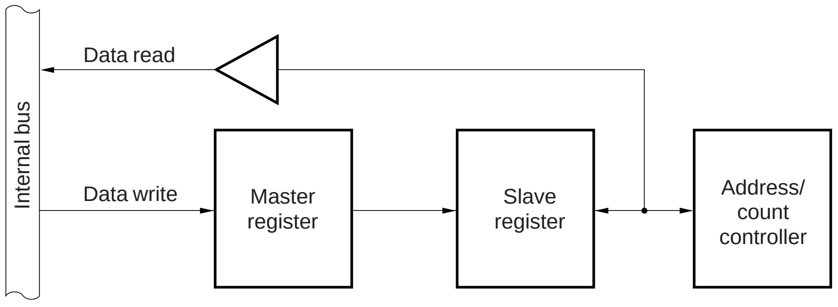
DMA channel 0 > DMA channel 1 > DMA channel 2 > DMA channel 3

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.

4.8 Next Address Setting Function

The DMA source address registers (DSAnH, DSAnL), DMA destination address registers (DDAnH, DDAnL), and DMA transfer count register (DBCn) are buffer registers with a 2-stage FIFO configuration (n = 0 to 3).
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 to 1 (however, the DMA transfer end interrupt may be issued even if DMA transfer is automatically started).
Figure 4-15 shows the configuration of the buffer register.

Figure 4-15: Buffer Register Configuration



4.9 DMA Transfer Start Factors

There are two types of DMA transfer start factors, as shown below.

(1) Request from software

If the STGn, ENn, and TCn bits of the DCHCn register are set as follows, DMA transfer starts (n = 0 to 3).

- STGn bit = 1
- ENn bit = 1
- TCn bit = 0

(2) Request from on-chip peripheral I/O

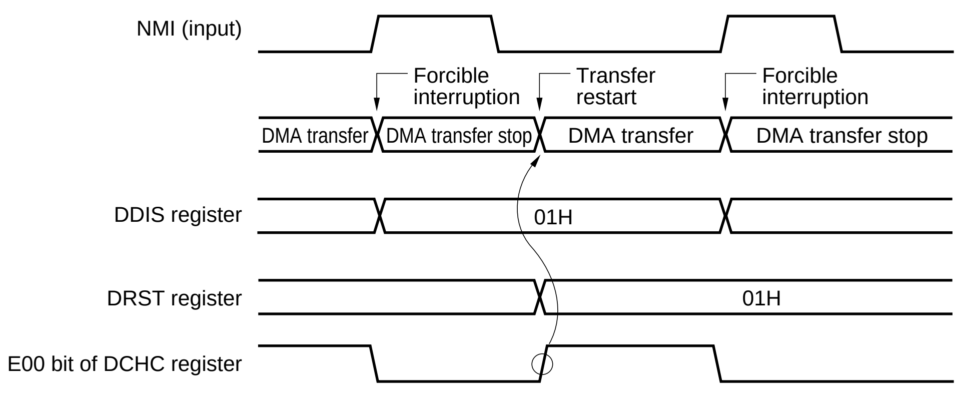
If, when the ENn and TCn bits of the DCHCn register are set as shown below, an interrupt request is issued from the on-chip peripheral I/O that is set in the DTFRn register, DMA transfer starts (n = 0 to 3).

- ENn bit = 1
- TCn bit = 0

4.10 Forcible Interruption

DMA transfer can be forcibly interrupted by NMI input during DMA transfer. At such a time, the DMAC resets the ENn bit of the DCHCn register of all channels to 0 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 (n = 0 to 3).
In the single-step transfer mode or block transfer mode, the DMA transfer request is held in the DMAC. If the ENn bit is set to 1, DMA transfer restarts from the point where it was interrupted.
In the single transfer mode, if the ENn bit is set to 1, the next DMA transfer request is acknowledged and DMA transfer starts.

Figure 4-16: Example of Forcible Interruption of DMA Transfer



4.11 DMA Transfer End

4.11.1 DMA transfer end interrupt

When DMA transfer ends and the TCn bit of the DCHCn register is set to 1, a DMA transfer end interrupt (INTDMAn) is issued to the interrupt controller (INTC) (n = 0 to 3).

4.11.2 Terminal count output upon DMA transfer end

The terminal count signal becomes active for one clock during the last DMA transfer cycle.

4.12 Forcible Termination

In addition to the forcible interruption operation by means of NMI input, DMA transfer can be forcibly terminated by the INITn bit of the DCHCn register (n = 0 to 3).

Remark: 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 **4.8 Next Address Setting Function** and **4.2.4 DMA addressing control registers 0 to 3 (DADC0 to DADC3)**).

4.13 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.

[MEMO]

Chapter 5 Interrupt/Exception Processing Function

5.1 Outline

The V850E/CA4 HELIOS is provided with a dedicated interrupt controller for interrupt servicing, which realizes a high-performance interrupt function that can service interrupt requests from a total of 59 sources.
An interrupt is an event that occurs asynchronously (independently of program execution), and an exception is an event that occurs synchronously (dependently on program execution). Generally, an exception takes precedence over an interrupt.
The V850E/CA4 HELIOS can process interrupt requests from the internal peripheral hardware and external sources. Moreover, exception processing can be started (exception trap) by the TRAP instruction (software exception) or by generation of an exception event (fetching of an illegal op code).

5.1.1 Features

- Interrupts
    - Non-maskable interrupt: 2 sources
    - Maskable interrupt: 54 sources
    - 8 levels programmable priorities
    - Mask specification for the interrupt request according to priority
    - Mask can be specified to each maskable interrupt request.
    - Valid edge for detection of external interrupt request signal can be specified.
- Exceptions
    - Software exceptions: 32 sources
    - Exception trap: 1 source (illegal op code exception)

Interrupt/exception sources are listed in Table 5-1.

Table 5-1: Interrupt Source List (1/3)

Type	Default Priority	Name	Trigger	Exception Code	Handler Address	Interrupt Control Register	Interrupt Request Flag Name
Reset	-	RESET	Reset input, Watchdog Reset	0000H	00000000H	-	
Non-maskable	-	NMIWDT	WatchDog Timer overflow	0010H	00000010H	-	
	-	NMI	NMI pin input	0030H	00000030H	-	
Software exception	-	TRAP0n **Note**	TRAP instruction (n=0-FH)	004nH **Note**	00000040H	-	
	-	TRAP1n **Note**	TRAP instruction (n=0-FH)	005nH **Note**	00000050H	-	
Exception trap	-	ILGOP	Illegal op code	0060H	00000060H	-	

Note: n= 0 to FH

Table 5-1: Interrupt Source List (2/3)

Type	Default Priority	Name	Trigger	Exception Code	Handler Address	Interrupt Control Register	Interrupt Request Flag Name
Maskable Interrupt	0	INTP00	Edge detection on INTP00 pin	0080H	00000080H	P00IC	P00IF
	1	INTP01	Edge detection on INTP01 pin	0090H	00000090H	P01IC	P01IF
	2	INTP02	Edge detection on INTP02 pin	00A0H	000000A0H	P02IC	P02IF
	3	INTDET	AD voltage drop detection	00B0H	000000B0H	DETIC	DETIF
	4	INTWT	Watch timer overflow	00C0H	000000C0H	WTIC	WTIF
	5	INTTMG00	16-bits timer G00 overflow	00D0H	000000D0H	TMG00IC	TMG00IF
	6	INTTMG01	16-bits timer G01 overflow	00E0H	000000E0H	TMG01IC	TMG01IF
	7	INTCCG00	TIG00 valid edge detection or GCC00 and TMG00 match	00F0H	000000F0H	CCG00IC	CCG00IF
	8	INTCCG01	TIG01 valid edge detection or GCC01 and TMG0n match	0100H	00000100H	CCG01IC	CCG01IF
	9	INTCCG02	TIG02 valid edge detection or GCC02 and TMG0n match	0110H	00000110H	CCG02IC	CCG02IF
	10	INTCCG03	TIG03 valid edge detection or GCC03 and TMG0n match	0120H	00000120H	CCG03IC	CCG03IF
	11	INTCCG04	TIG04 valid edge detection or GCC04 and TMG0n match	0130H	00000130H	CCG04IC	CCG04IF
	12	INTCCG05	TIG05 valid edge detection or GCC05 and TMG01 match	0140H	00000140H	CCG05IC	CCG05IF
	13	INTTMG10	16-bits timer G10 overflow	0150H	00000150H	TMG10IC	TMG10IF
	14	INTTMG11	16-bits timer G11 overflow	0160H	00000160H	TMG11IC	TMG11IF
	15	INTCCG10	TIG10 valid edge detection or GCC10 and TMG10 match	0170H	00000170H	CCG10IC	CCG10IF
	16	INTCCG11	TIG11 valid edge detection or GCC11 and TMG1n match	0180H	00000180H	CCG11IC	CCG11IF
	17	INTCCG12	TIG12 valid edge detection or GCC12 and TMG1n match	0190H	00000190H	CCG12IC	CCG12IF
	18	INTCCG13	TIG13 valid edge detection or GCC13 and TMG1n match	01A0H	000001A0H	CCG13IC	CCG13IF
	19	INTCCG14	TIG14 valid edge detection or GCC14 and TMG1n match	01B0H	000001B0H	CCG14IC	CCG14IF
	20	INTCCG15	TIG15 valid edge detection or GCC15 and TMG11 match	01C0H	000001C0H	CCG15IC	CCG15IF
	21	INTCMD0	16-bits timer D0 compare match	01D0H	000001D0H	CMD0IC	CMD0IF
	22	INTCMD1	16-bits timer D1 compare match	01E0H	000001E0H	CMD1IC	CMD1IF
	23	INTWDTM	WatchDog maskable interrupt	01F0H	000001F0H	WDTMIC	WDTMIF
	24	INTDMA0	DMA channel 0 end of transfer	0200H	00000200H	DMA0IC	DMA0IF
	25	INTDMA1	DMA channel 1 end of transfer	0210H	00000210H	DMA1IC	DMA1IF
	26	INTDMA2	DMA channel 2 end of transfer	0220H	00000220H	DMA2IC	DMA2IF
	27	INTDMA3	DMA channel 3 end of transfer	0230H	00000230H	DMA3IC	DMA3IF
	28	INTFC1RX	FCAN0 receive	0240H	00000240H	FC1RXIC	FC1RXIF
	29	INTFC1TX	FCAN0 transfer	0250H	00000250H	FC1TXIC	FC1TXIF
	30	INTFC1ER	FCAN0 communication error	0260H	00000260H	FC1ERIC	FC1ERIF
	31	INTMAC	FCAN MAC interrupt	0270H	00000270H	MACIC	MACIF
	32	INTCSI00	CSI00 end of transfer	0280H	00000280H	CSI00IC	CSI00IF

Table 5-1: Interrupt Source List (3/3)

Type	Default Priority	Name	Trigger	Exception Code	Handler Address	Interrupt Control Register	Interrupt Request Flag Name
Maskable Interrupt	33	INTCSI01	CSI01 end of transfer	0290H	00000290H	CSI01IC	CSI01IF
	34	INTSR60	UART60 end of reception	02A0H	000002A0H	SR60IC	SR60IF
	35	INTST60	UART60 end of transmission	02B0H	000002B0H	ST60IC	ST60IF
	36	INTSRE60	UART60 reception error	02C0H	000002C0H	SRE60IC	SER60IF
	37	INTSR61	UART61 end of reception	02D0H	000002D0H	SR61IC	SR61IF
	38	INTST61	UART61 end of transmission	02E0H	000002E0H	ST61IC	ST61IF
	39	INTSRE61	UART61 reception error	02F0H	000002F0H	SRE61IC	SER61IF
	40	INTAD	AD conversion done	0300H	00000300H	ADIC	ADIF
	41	INTFC2RX	FCAN1 receive	0310H	00000310H	FC2RXIC	FC2RXIF
	42	INTFC2TX	FCAN1 transfer	0320H	00000320H	FC2TXIC	FC2TXIF
	43	INTFC2ER	FCAN1 communication error	0330H	00000330H	FC2ERIC	FC2ERIF
	44	INTP10	Edge detection on INTP10 pin	0340H	00000340H	P10IC	P10IF
	45	INTP15	Edge detection on INTP15 pin	0350H	00000350H	P15IC	P15IF
	46	INTP20	Edge detection on INTP20 pin	0360H	00000360H	P20IC	P20IF
	47	INTP25	Edge detection on INTP25 pin	0370H	00000370H	P25IC	P25IF
	48	INTP30	Edge detection on INTP30 pin	0380H	00000380H	P30IC	P30IF
	49	INTP32	Edge detection on INTP32 pin	0390H	00000390H	P32IC	P32IF
	50	INTP34	Edge detection on INTP34 pin	03A0H	000003A0H	P34IC	P34IF
	51	INTWTI	Watch timer interval timer overflow	03B0H	000003B0H	WTIIC	WTIIF
	52	INTCSI10	CSI10 end of transfer	03C0H	000003C0H	CSI10IC	CSI10IF
	53	INTDOVF	DMA request overflow	03D0H	000003D0H	DOVFIC	DOVFIF
	54	INTBRG	Baud rate generator	03E0H	000003E0H	BRGIC	BRGIF

Remarks:

1. Default Priority: Priority that takes precedence when two or more maskable interrupt requests are present at the same time. The highest priority is 0.

2. The execution address of the illegal instruction when an illegal op code exception occurs is equal to (saved PC - 4).

3. The choice between a non-maskable interrupt (NMIWDT) and a maskable interrupt (INTWDTM) generation from the Watchdog Timer is made with the bits WDM1 and WDM0 of the watchdog timer mode register (WDTM).

The value of PC saved when an interrupt/exception (other than $\overline{\text{RESET}}$) occurs is the value of the current PC, which holds the address of the next instruction to be executed when returning from interrupt handling routine. However if the interrupt request occurs during execution of a DIVH instruction, the value saved is the address of the DIVH instruction itself (rather than the address of the instruction following the DIVH), because the DIVH is cancelled in this case, and restarted completely after interrupt servicing.

5.2 Non-Maskable Interrupt

The non-maskable interrupt is acknowledged unconditionally, even when interrupts are disabled (DI state). The NMI is not subject to priority control and takes precedence over all the other interrupts. Non-maskable interrupts of the V850E/CA4 HELIOS are available for the following two requests:

- NMI pin input (NMI)
- Non-maskable watchdog timer interrupt request (NMIWDT)

When the valid edge specified by the register NMIM is detected at the NMI pin, a non-maskable interrupt occurs.

The non-maskable interrupt from Watchdog Timer (NMIWDT) can occur only if the watchdog timer mode register (WDTM) is set with (WDM1,WDM0) = (0,1).

5.2.1 NP flag

The NP flag is a status flag that indicates that non-maskable interrupt (NMI) servicing is under execution. This flag is set when the NMI interrupt request has been acknowledged.
The NP flag is stored in the PSW register.

Figure 5-1: NP Flag (NP)

Symbol	31 … 8	7	6	5	4	3	2	1	0	After Reset
PSW	0	NP	EP	ID	SAT	CY	OV	S	Z	0000 0020H

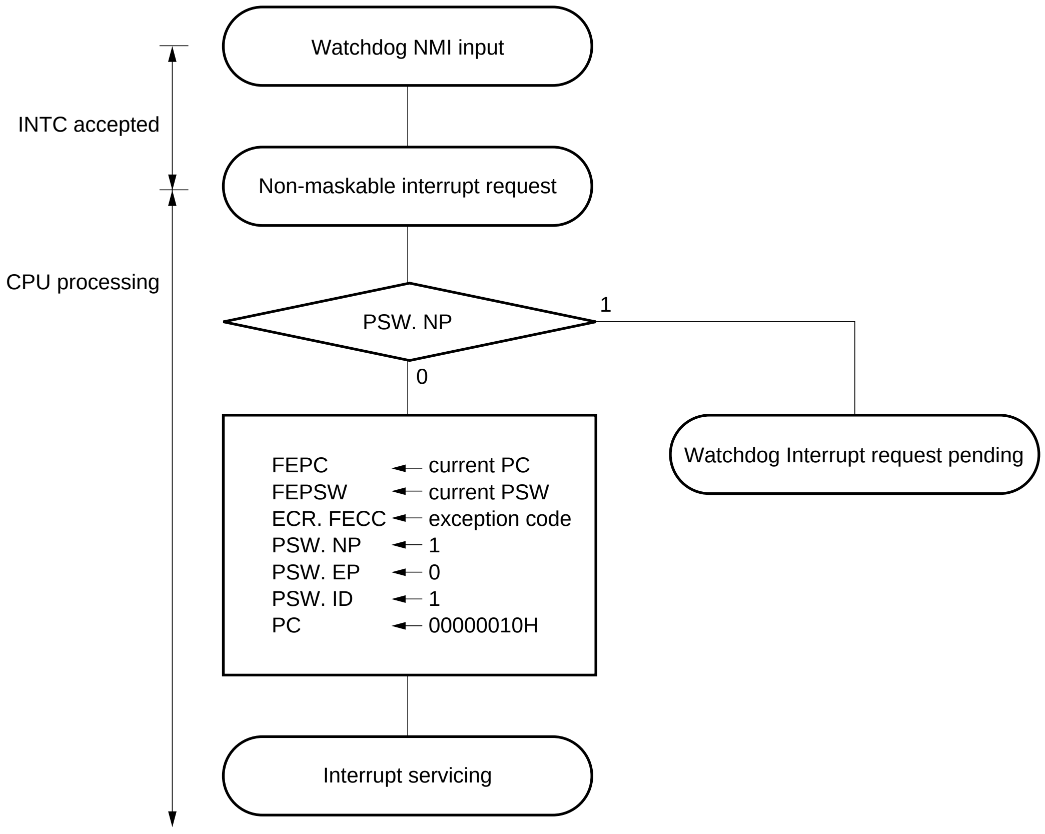
NP	NMI Servicing State
0	No NMI interrupt servicing
1	NMI interrupt currently servicing

5.2.2 Watchdog Timer NMI Operation

If the non-maskable interrupt is generated by the Watchdog Timer, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the current PC to FEPC.
(2) Saves the current PSW to FEPSW.
(3) Writes exception code 0010H to the higher half-word (FECC) of ECR.
(4) Sets the NP and ID bits of PSW and clears the EP bit.
(5) Loads the handler address (00000010H) of the non-maskable interrupt routine to the PC, and transfers control.

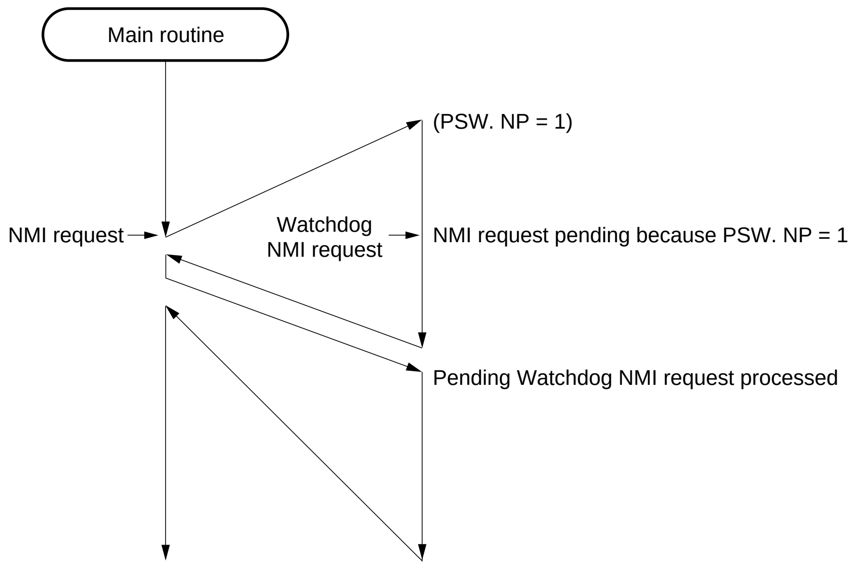
Figure 5-2: Watchdog Timer Non-Maskable Interrupt Servicing



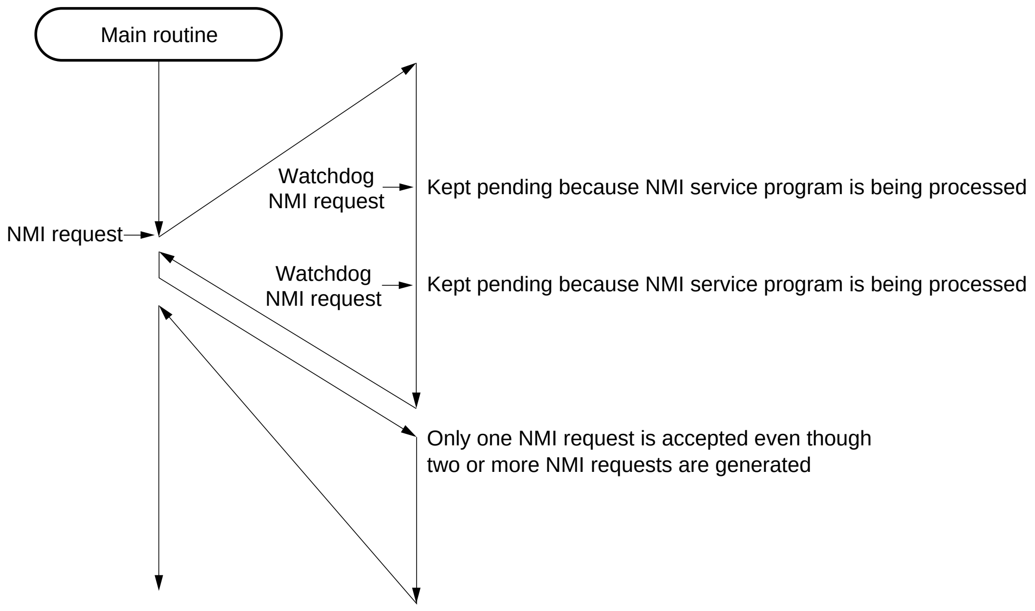
While the service routine of the non-maskable interrupt is being executed (PSW.NP = 1), the acknowledgement of another non-maskable interrupt request from the watchdog timer is kept pending. The pending NMI is acknowledged after the original service routine of the non-maskable interrupt under execution has been terminated (by the RETI instruction), or when PSW.NP is cleared to 0 by the LDSR instruction. Note that if two or more NMI requests are generated by the watchdog timer during the execution of the NMI service routine, the number of NMIs that will be acknowledged after PSW.NP goes to "0", is only one.

Figure 5-3: Acknowledging Watchdog Non-Maskable Interrupt Request

(a) If a new Watchdog NMI request is generated while an NMI service routine is executing:


Main routine
(PSW. NP = 1)
NMI request
Watchdog NMI request
NMI request pending because PSW. NP = 1
Pending Watchdog NMI request processed

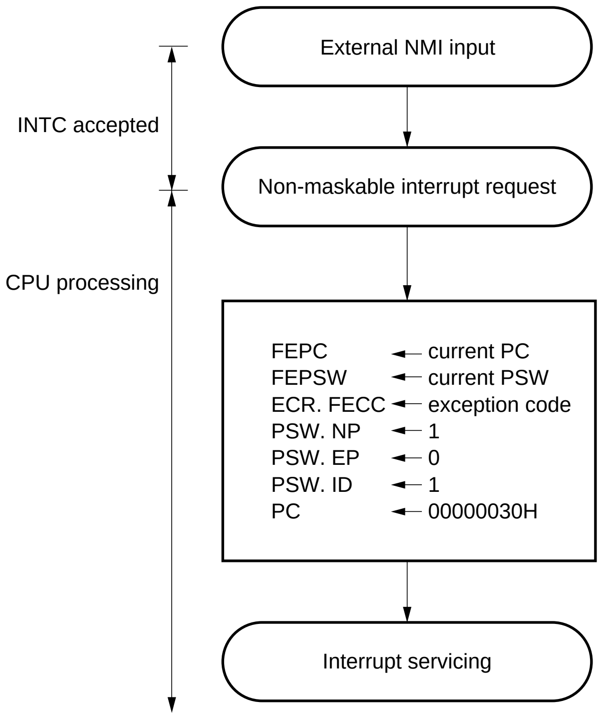
(b) If a new Watchdog NMI request is generated twice while an NMI service routine is executing:


Main routine
Watchdog NMI request
Kept pending because NMI service program is being processed
NMI request
Watchdog NMI request
Kept pending because NMI service program is being processed
Only one NMI request is accepted even though two or more NMI requests are generated

5.2.3 External NMI Operation

An external NMI request is always acknowledged, regardless of the state of the NP bit.

Figure 5-4: External Non-Maskable Interrupt Servicing



Note: The execution of the NMI handler can be interrupted by a new incoming external NMI request.

(a) Nesting of external NMI handlers

As the V850E/CA4 HELIOS provides just one context saving area for NMI (FEPC, FEPSW), the acknowledgement of the new external NMI request causes FEPC and FEPSW to be overwritten: the context saved by the first handler activation is lost.

So, like with maskable interrupts, the way to return correctly from nested NMI handlers is

- at handler prolog, save FEPC, FEPSW
- at handler epilog, restore FEPC, FEPSW

Caution: Even when using the above described method, if a new external NMI request arrives during handler prolog or epilog, correct return cannot be achieved: if such configuration may appear, nesting NMIs is not possible, and the only way to avoid program crash is to generate an internal reset with the watchdog timer at the end of the NMI handler. Note that it is possible to check if the new NMI occurred during prolog or epilog by comparing the value of FEPC with the boundaries of the NMI handler.

(b) Edge detection function of NMI pin

The NMI pin valid edge can be selected from falling edge, rising edge or both edges, and the level of the NMI pin can be read through the NMIM register.
The NMIM register can be read/written in 8-bit units.
Only the first write access to NMIM after reset release is granted.

Figure 5-5: NMI Edge Specification: NMIM Register Format

	7	6	5	4	3	2	1	0	Address	After reset
NMIM	PNMI	0	0	0	0	0	ESN1	ESN0	FFFF FC08H	03H
	R	R	R	R	R	R	R/W	R/W		

ESN1	ESN0	Edge Selection for NMI
0	0	Falling edge selected
0	1	Rising edge selected
1	0	Both rising and falling edges selected
1	1	Both rising and falling edges selected

PNMI	NMI pin level
0	Read level is 0
1	Read level is 1

(c) Noise elimination circuit of NMI pin

NMI pin noise is eliminated by the noise elimination circuit with analog delay. Therefore, a signal input to the NMI pin is not detected as an edge, unless it maintains its input level for a certain period. The edge is detected after a certain period has elapsed.
NMI pin can be used for cancelling the standby mode, even if the clock is stopped, because the noise elimination is based on analog delay.

5.2.4 Watchdog or external NMI Restore

Execution is restored from the non-maskable interrupt service by the RETI instruction.

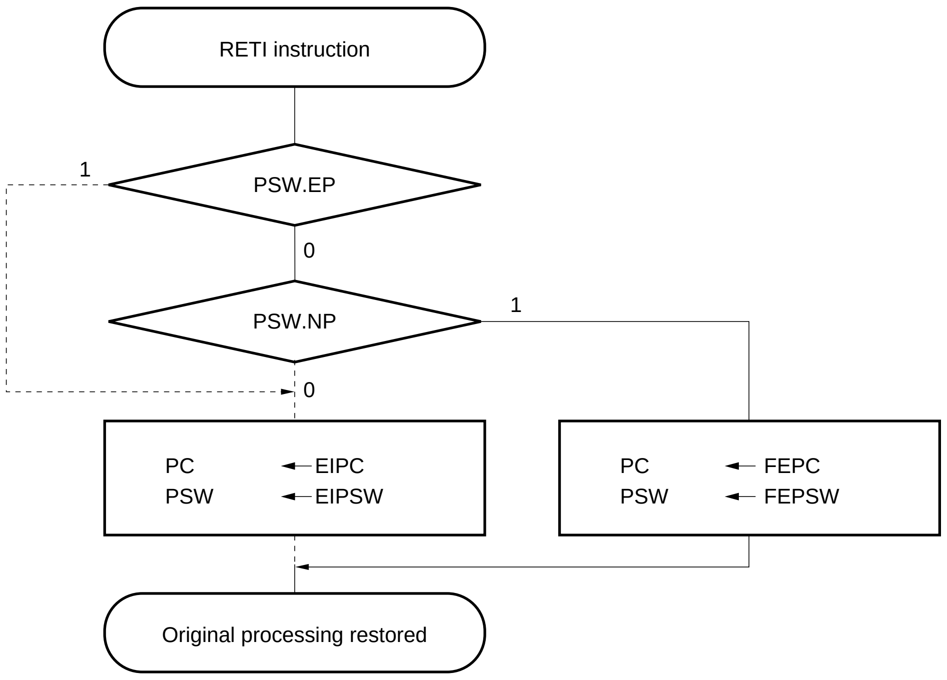
Operation of RETI instruction

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 PC and PSW from FEPC and FEPSW, respectively, because the EP bit of PSW is 0 and the NP bit of PSW is 1.
(2) Transfers control back to the address of the restored PC and PSW.

How the RETI instruction is processed is shown below.

Figure 5-6: RETI Instruction Processing



Caution: When the PSW.EP bit and PSW.NP bit are changed by the LDSR instruction during the non-maskable interrupt service, 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.

5.3 Maskable Interrupts

The V850E/CA4 HELIOS has 54 maskable interrupt sources.
Maskable interrupt requests can be masked by interrupt control registers.
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, allowing programmable priority control.
When an interrupt request has been acknowledged, the acknowledgement of other maskable interrupts is disabled and the interrupt disabled (DI) status is set.
When the EI instruction is executed in an interrupt servicing routine, the interrupt enabled (EI) status is set which enables interrupts having a higher priority to immediately interrupt the current service routine in progress. Note that only interrupts with a higher priority will have this capability; interrupts with the same priority level cannot be nested.
To use multiple interrupts, it is necessary to save EIPC and EIPSW to memory or a register before executing the EI instruction, and restore EIPC and EIPSW to the original values before the RETI instruction.
When the WDCS1 and WDCS0 bits of the watchdog timer mode register (WDTM) are set to 0, the watchdog timer overflow interrupt functions as a maskable interrupt (INTWDTM).

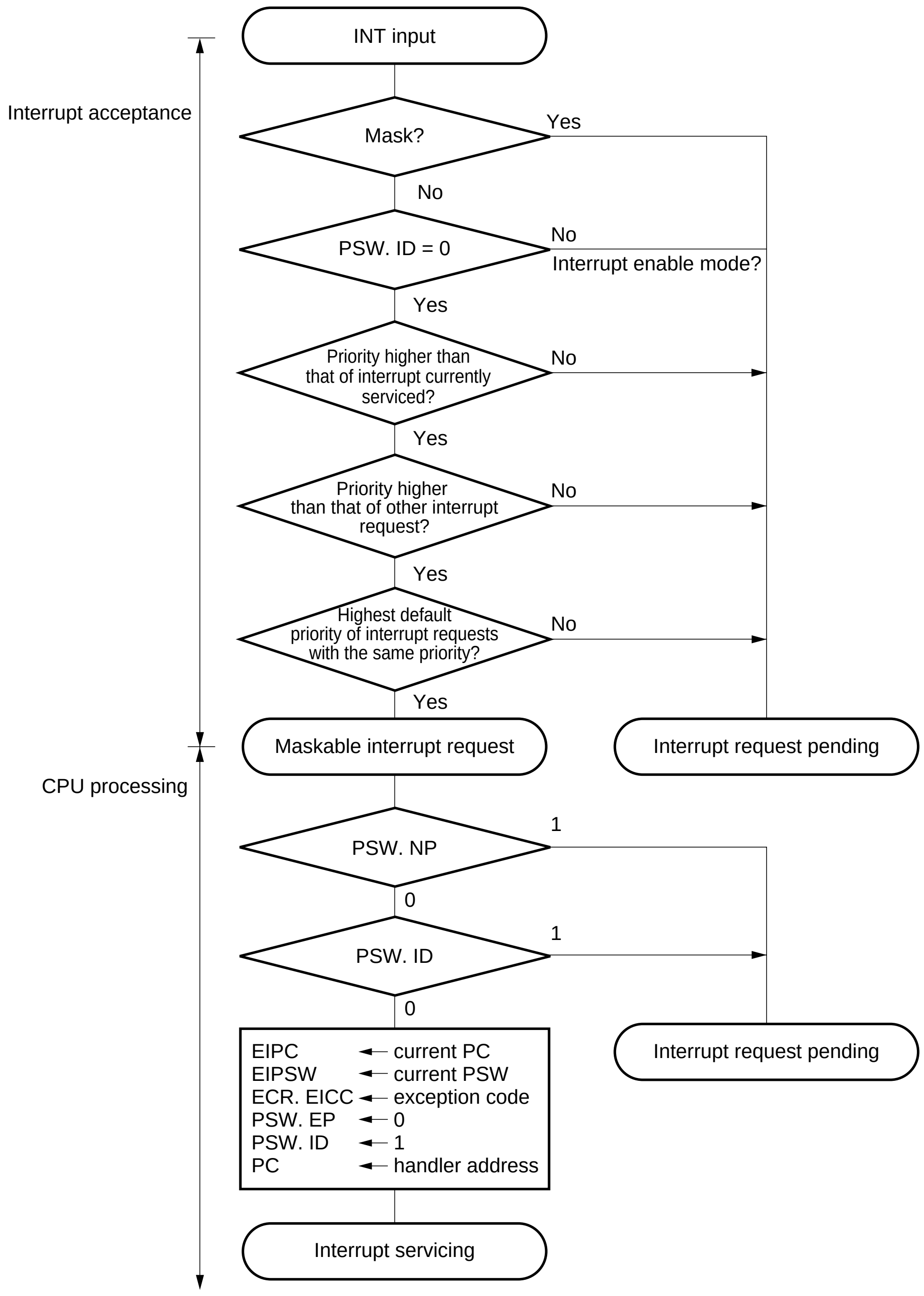
5.3.1 Operation

If a maskable interrupt occurs, the CPU performs the following processing, and transfers control to a handler routine:

(1) Saves the current PC to EIPC.
(2) Saves the current PSW to EIPSW.
(3) Writes an exception code to the lower half-word of ECR (EICC).
(4) Sets the ID bit of PSW and clears the EP bit.
(5) Loads the corresponding handler address to the PC, and transfers control.

While an interrupt is being serviced (when PSW.NP = 1 or PSW.ID = 1), a new interrupt request is internally kept pending. When the interrupts are unmasked, or when PSW.NP = 0 and PSW.ID = 0 by using the RETI or LDSR instructions, the pending interrupt is input to start the new maskable interrupt servicing.
How the maskable interrupts are serviced is shown below.

Figure 5-7: Maskable Interrupt Servicing


INT input
Interrupt acceptance
Mask?
Yes
No
PSW. ID = 0
No
Interrupt enable mode?
Yes
Priority higher than that of interrupt currently serviced?
No
Yes
Priority higher than that of other interrupt request?
No
Yes
Highest default priority of interrupt requests with the same priority?
No
Yes
Maskable interrupt request
Interrupt request pending
CPU processing
PSW. NP
1
0
PSW. ID
1
0
EIPC ← current PC
EIPSW ← current PSW
ECR. EICC ← exception code
PSW. EP ← 0
PSW. ID ← 1
PC ← handler address
Interrupt request pending
Interrupt servicing

5.3.2 Restore

To restore execution from the maskable interrupt servicing, the RETI instruction is used.

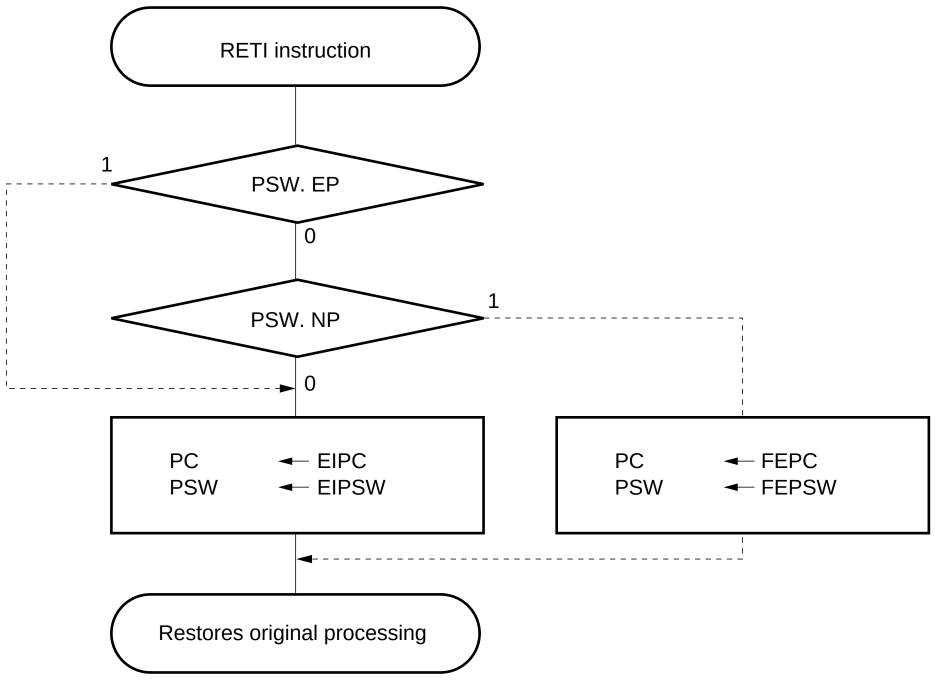
Operation of RETI instruction

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 PC and PSW from EIPC and EIPSW because the EP bit of PSW is 0 and the NP bit of PSW is 0.
(2) Transfers control to the address of the restored PC and PSW.

The processing of the RETI instruction is shown below.

Figure 5-8: RETI Instruction Processing

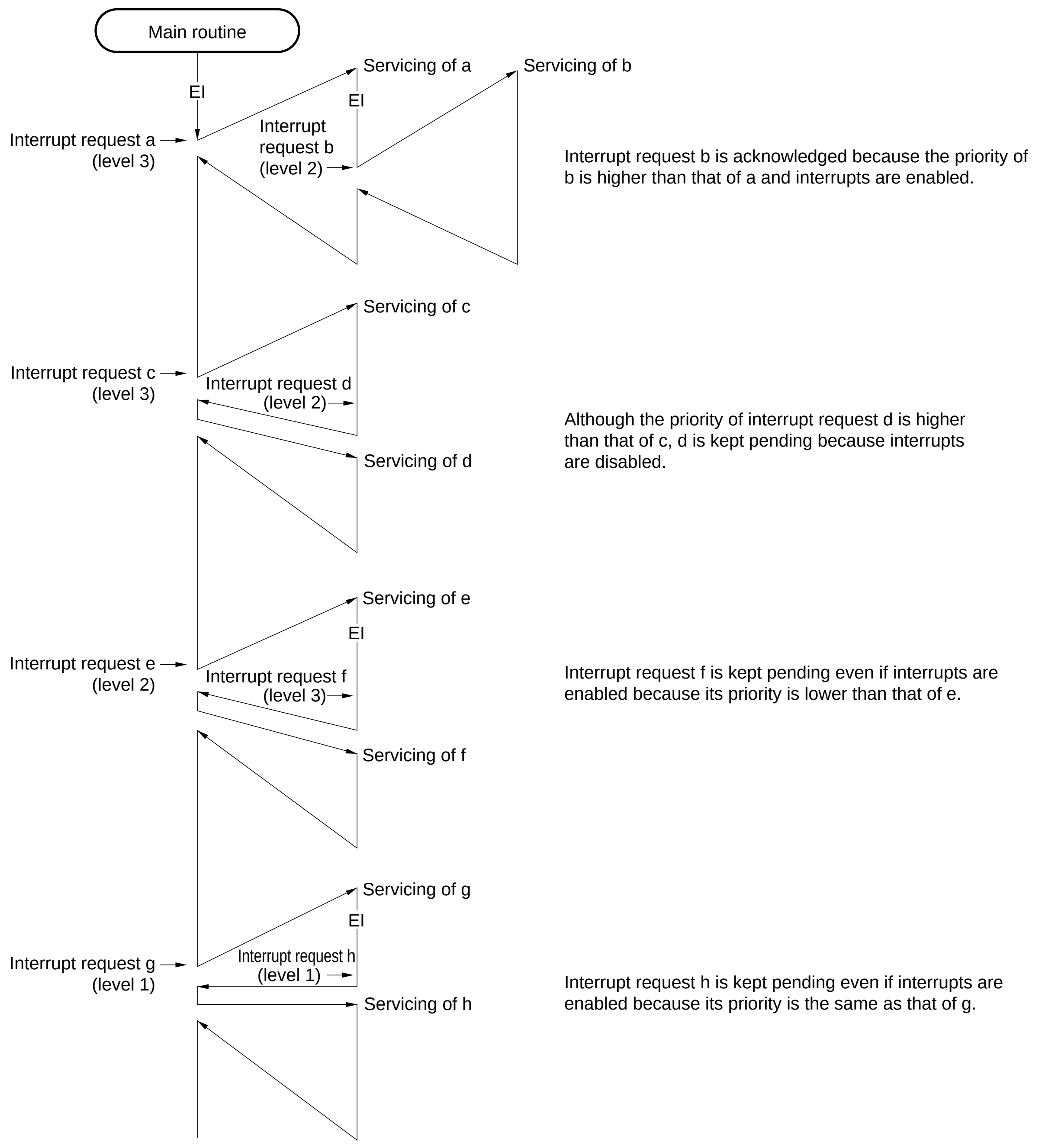


Caution: When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during the maskable interrupt service, in order to restore the PC and PSW correctly during recovery by the RETI instruction, it is necessary to set PSW.EP back to 0 and PSW.NP back to 0 using the LDSR instruction immediately before the RETI instruction.

5.3.3 Priorities of maskable interrupts

The V850E/CA4 HELIOS provides a multiple interrupt service that acknowledges an interrupt while servicing another interrupt. 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 which are specified by interrupt priority level specification bits (xxnPR bits in the Interrupt Control Register xxnIC). When two or more interrupts having the same priority level specified by xxnPR are generated at the same time, interrupts are serviced in order depending on the priority level allocated to each interrupt request types (default priority level) beforehand. For more information, refer to Table 5-1, “Interrupt Source List,” on page 109. The programmable priority control customizes interrupt requests into eight levels by setting the priority level specification bits.
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 into the interrupt service program) to set the interrupt enable mode.
Please see Figure 5-9, “Example of Interrupt Nesting Service (1/2),” on page 122 and Figure 5-10, “Example of Servicing Interrupt Requests Simultaneously Generated,” on page 124.

Figure 5-9: Example of Interrupt Nesting Service (1/2)

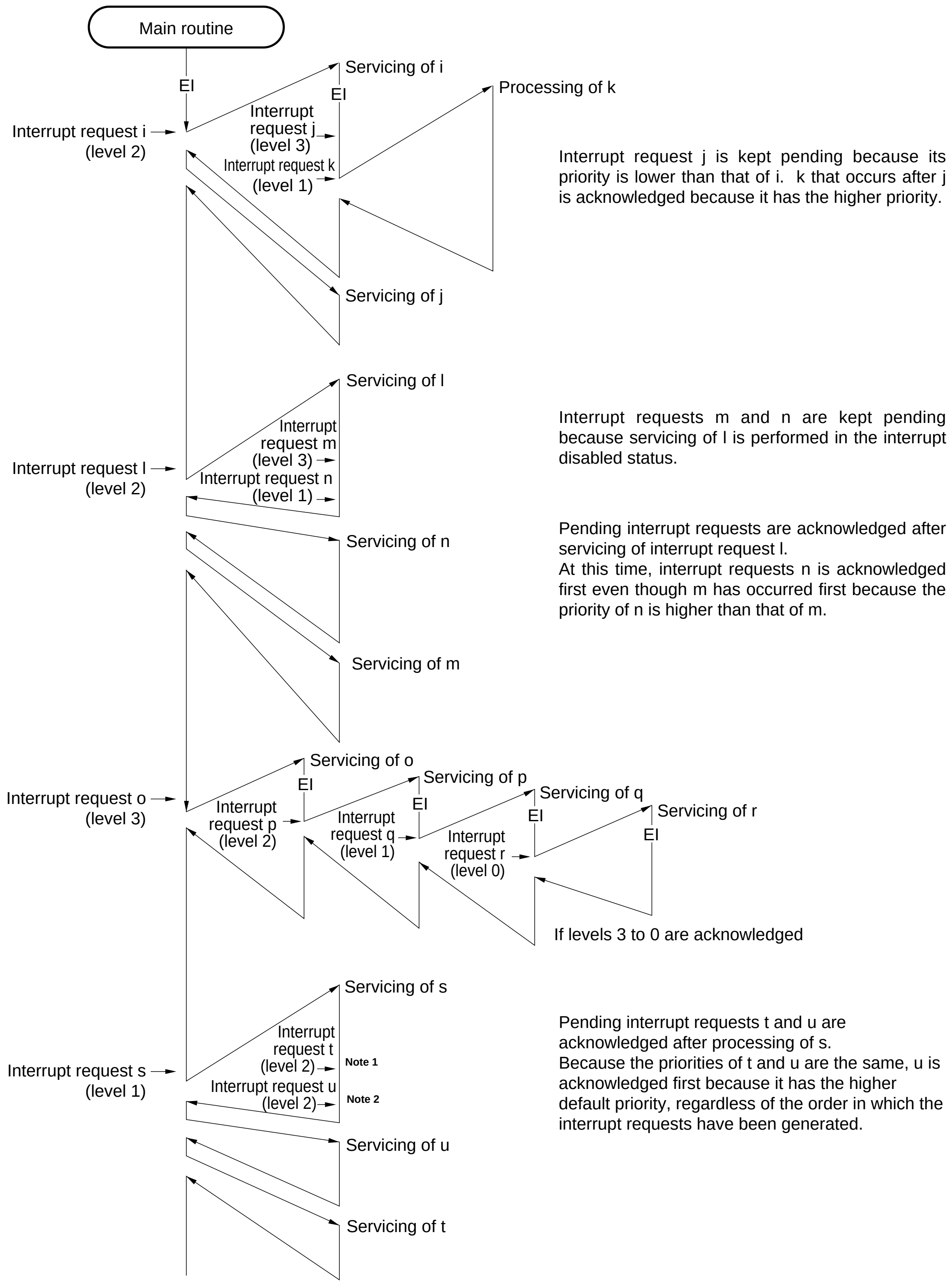

Main routine
EI
Servicing of a
Servicing of b
EI
Interrupt request a (level 3)
Interrupt request b (level 2)
Interrupt request b is acknowledged because the priority of b is higher than that of a and interrupts are enabled.
Servicing of c
Interrupt request c (level 3)
Interrupt request d (level 2)
Servicing of d
Although the priority of interrupt request d is higher than that of c, d is kept pending because interrupts are disabled.
Servicing of e
EI
Interrupt request e (level 2)
Interrupt request f (level 3)
Interrupt request f is kept pending even if interrupts are enabled because its priority is lower than that of e.
Servicing of f
Servicing of g
EI
Interrupt request g (level 1)
Interrupt request h (level 1)
Servicing of h
Interrupt request h is kept pending even if interrupts are enabled because its priority is the same as that of g.

Caution: The values of EIPC and EIPSW must be saved before executing multiple interrupts.

Remarks: **1.** “a” to “u” in the figure are the 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 5-9: Example of Interrupt Nesting Process (2/2)


Main routine
EI
Servicing of i
EI
Processing of k
Interrupt request i (level 2)
Interrupt request j (level 3)
Interrupt request k (level 1)
Servicing of j
Servicing of l
Interrupt request l (level 2)
Interrupt request m (level 3)
Interrupt request n (level 1)
Servicing of n
Servicing of m
Servicing of o
EI
Servicing of p
EI
Servicing of q
EI
Servicing of r
EI
Interrupt request o (level 3)
Interrupt request p (level 2)
Interrupt request q (level 1)
Interrupt request r (level 0)
If levels 3 to 0 are acknowledged
Servicing of s
Interrupt request s (level 1)
Interrupt request t (level 2)
Note 1
Interrupt request u (level 2)
Note 2
Servicing of u
Servicing of t

Interrupt request j is kept pending because its priority is lower than that of i. k that occurs after j is acknowledged because it has the higher priority.

Interrupt requests m and n are kept pending because servicing of l is performed in the interrupt disabled status.

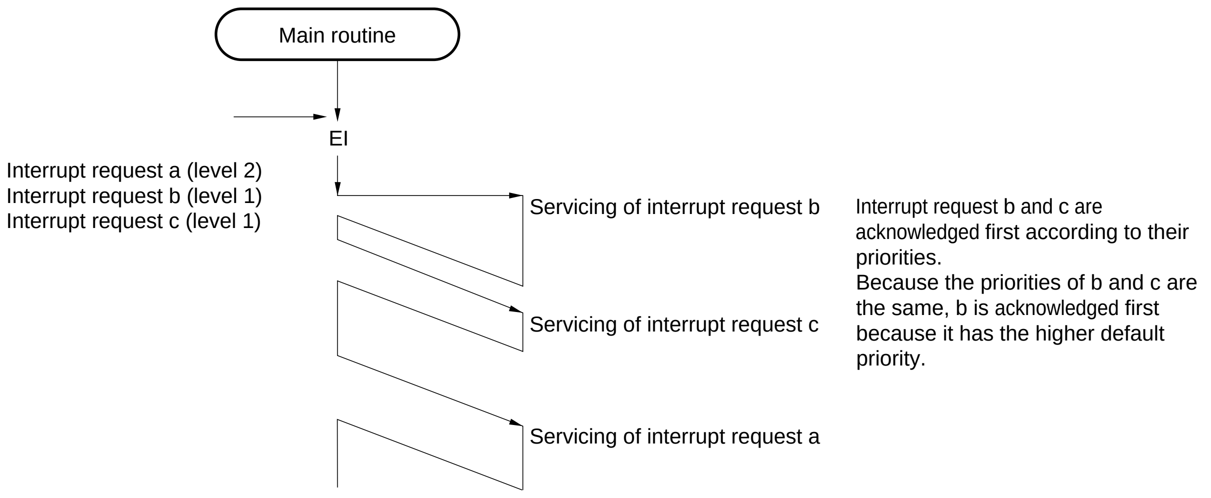
Pending interrupt requests are acknowledged after servicing of interrupt request l.
At this time, interrupt requests n is acknowledged first even though m has occurred first because the priority of n is higher than that of m.

Pending interrupt requests t and u are acknowledged after processing of s.
Because the priorities of t and u are the same, u is acknowledged first because it has the higher default priority, regardless of the order in which the interrupt requests have been generated.

Notes: 1. Lower default priority

2. Higher default priority

Figure 5-10: Example of Servicing Interrupt Requests Simultaneously Generated


Main routine
EI
Interrupt request a (level 2)
Interrupt request b (level 1)
Interrupt request c (level 1)
Servicing of interrupt request b
Servicing of interrupt request c
Servicing of interrupt request a
Interrupt request b and c are acknowledged first according to their priorities.
Because the priorities of b and c are the same, b is acknowledged first because it has the higher default priority.

5.3.4 Interrupt control register (xxnIC)

An interrupt control register is assigned to each maskable interrupt and sets the control conditions for each maskable interrupt request.
The interrupt control register can be read/written in 8-bit or 1-bit units.

Caution: Write access to xxnIC registers may cause spurious interrupt to be generated. So be sure to manipulate xxnIC (and xxnIF flags) while interrupts are disabled.

Figure 5-11: Interrupt Control Register (xxnIC) Format

	7	6	5	4	3	2	1	0	Address	After reset
xxnIC	xxnIF	xxnMK	0	0	0	xxnPR2	xxnPR1	xxnPR0	FFFF F110H to FFFF F17AH	47H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

xxnIF	Interrupt request flag [Note]
0	Interrupt request not generated
1	Interrupt pending from xxn source

xxnMK	Interrupt mask flag
0	Enables interrupt servicing (interrupt NOT masked)
1	Disables interrupt servicing (interrupt masked)

xxnPR2	xxnPR1	xxnPR0	Interrupt priority specification bit
0	0	0	Specifies level 0 (highest)
0	0	1	Specifies level 1
0	1	0	Specifies level 2
0	1	1	Specifies level 3
1	0	0	Specifies level 4
1	0	1	Specifies level 5
1	1	0	Specifies level 6
1	1	1	Specifies level 7 (lowest)

Note: Automatically reset by hardware when interrupt request is acknowledged.

Remark: xx:Identification name of each peripheral unit (P0, CSI, DMA...)
n:Peripheral unit number.

The detailed list of the interrupt control register names and interrupt request flags is shown in Table 5-1, “Interrupt Source List,” on page 109.

5.3.5 Interrupt Mask Registers (IMRn)

In addition to the individual bitwise manipulation capability offered by the xxnMK bits, the V850E/CA4 HELIOS allows the manipulation of the interrupt masks with the Interrupt Mask Registers IMRn (n = 0 to 3).
These registers can be manipulated with 16-bit, 8-bit or 1-bit memory manipulation instructions.

Figure 5-12: Interrupt Mask Register Format IMR0

(a) When using 16-bit access to IMR0, use IMR0

	7	6	5	4	3	2	1	0	Address	After reset
IMR0	CCG00MK	TMG01MK	TMG00MK	WTMK	DETMK	P02MK	P01MK	P00MK	FFFF F100H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		
	15	14	13	12	11	10	9	8		
	CCG10MK	TMG11MK	TMG10MK	CCG05MK	CCG04MK	CCG03MK	CCG02MK	CCG01MK		
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

(b) When using 8-bit access to IMR, use IMR0L for the LSB byte, and IMR0H for the MSB byte

	7	6	5	4	3	2	1	0	Address	After reset
IMR0L	CCG00MK	TMG01MK	TMG00MK	WTMK	DETMK	P02MK	P01MK	P00MK	FFFF F100H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
IMR0H	CCG10MK	TMG11MK	TMG10MK	CCG05MK	CCG04MK	CCG03MK	CCG02MK	CCG01MK	FFFF F101H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

(c) When using 1-bit access, use the xxnMK flag name.

Figure 5-13: Interrupt Mask Register Format IMR1

(a) When using 16-bit access to IMR1, use IMR1

	7	6	5	4	3	2	1	0	Address	After reset
IMR1	WDTMMK	CMD1MK	CMD0MK	CCG15MK	CCG14MK	CCG13MK	CCG12MK	CCG11MK	FFFF F102H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	15	14	13	12	11	10	9	8
	MACMK	FC1ERMK	FC1TXMK	FC1RXMK	DMA3MK	DMA2MK	DMA1MK	DMA0MK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(b) When using 8-bit access, use IMR1L for the LSB byte, and IMR1H for the MSB byte.

	7	6	5	4	3	2	1	0	Address	After reset
IMR1L	WDTMMK	CMD1MK	CMD0MK	CCG15MK	CCG14MK	CCG13MK	CCG12MK	CCG11MK	FFFF F102H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
IMR1H	MACMK	FC1ERMK	FC1TXMK	FC1RXMK	DMA3MK	DMA2MK	DMA1MK	DMA0MK	FFFF F103H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

(c) When using 1-bit access, use the xxnMK flag name.

Figure 5-14: Interrupt Mask Register Format IMR2

(a) When using 16-bit access to IMR2, use IMR2

	7	6	5	4	3	2	1	0	Address	After reset
IMR2	SER1MK	ST1MK	SR1MK	SER0MK	ST0MK	SR0MK	CSI01MK	CSI00MK	FFFF F104H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	15	14	13	12	11	10	9	8
	P25MK	P20MK	P15MK	P10MK	FC2ERMK	FC2TXMK	FC2RXMK	ADMK
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(b) When using 8-bit access, use IMR2L for the LSB byte and IMR2H for the MSB byte.

	7	6	5	4	3	2	1	0	Address	After reset
IMR2L	SER1MK	ST1MK	SR1MK	SER0MK	ST0MK	SR0MK	CSI01MK	CSI00MK	FFFF F104H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
IMR2H	P25MK	P20MK	P15MK	P10MK	FC2ERMK	FC2TXMK	FC2RXMK	ADMK	FFFF F105H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

(c) When using 1-bit access, use the xxnMK flag name.

Figure 5-15: Interrupt Mask Register Format IMR3

(a) When using 16-bit access to IMR3, use IMR3

	7	6	5	4	3	2	1	0	Address	After reset
IMR3	1	BRGMK	DOVFMK	CSI10MK	WTIMK	P34MK	P32MK	P30MK	FFFF F106H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	15	14	13	12	11	10	9	8
	1	1	1	1	1	1	1	1
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(b) When using 8-bit access, use IMR3L for the LSB byte.

	7	6	5	4	3	2	1	0	Address	After reset
IMR3L	1	1	DOVFMK	CSI10MK	WTIMK	P34MK	P32MK	P30MK	FFFF F106H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

(c) When using 1-bit access, use the xxnMK flag name.

5.3.6 In-service priority register (ISPR)

This register holds the priority levels of the maskable interrupts currently acknowledged. When an interrupt request is acknowledged, the bit of this register corresponding to the priority level of that interrupt is set to 1 and remains set while the end of the interrupt service.
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 when execution is returned from non-maskable processing or exception processing.
This register can be only read in 8-bit or 1-bit units.

Figure 5-16: In-Service Priority Register (ISPR) Format

	7	6	5	4	3	2	1	0	Address	After reset
ISPR	ISPR7	ISPR6	ISPR5	ISPR4	ISPR3	ISPR2	ISPR1	ISPR0	FFFF F1FAH	00H
	R	R	R	R	R	R	R	R		

ISPRn	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)

5.3.7 Global interrupt mask: ID (Interrupt Disable)

The global interrupt mask (ID) of the PSW controls the enabling and disabling of maskable interrupt requests. As a status flag, it also displays the current maskable interrupt acknowledgment condition.

Figure 5-17: Interrupt Disable Flag (ID)

Symbol	31	8	7	6	5	4	3	2	1	0	After Reset
PSW	0		NP	EP	ID	SAT	CY	OV	S	Z	0000 0020H

ID	Specifies Maskable Interrupt Servicing[Note]
0	Maskable interrupt acknowledgement enabled
1	Maskable interrupt acknowledgement disabled (pending)

Note: Interrupt disable flag (ID) function

- It 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 the PSW.
- Non-maskable interrupt 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) can be acknowledged when the xxnIF bit of xxnIC is set to 1, and the ID flag is reset to 0.

5.3.8 Edge detection function

Valid edges of the INTPn pins can be selected for each pin from the falling, rising, or both edges.
The choice between rising and falling edge is controlled by the INTMn register (n = 0 to 3).

After reset, the valid edge of each interrupt source is set to falling edge.

Figure 5-18: External Interrupts Edge Selection Register INTM0

	7	6	5	4	3	2	1	0	Address	After reset
INTM0	0	0	ES021	ES020	ES011	ES010	ES001	ES000	FFFF FC00H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

Figure 5-19: External Interrupts Edge Selection Register INTM1

	7	6	5	4	3	2	1	0	Address	After reset
INTM1	0	0	0	0	ES151	ES150	ES101	ES100	FFFF FC02H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

Figure 5-20: External Interrupts Edge Selection Register INTM2

	7	6	5	4	3	2	1	0	Address	After reset
INTM2	0	0	0	0	ES251	ES250	ES201	ES200	FFFF FC04H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

Figure 5-21: External Interrupts Edge Selection Register INTM3

	7	6	5	4	3	2	1	0	Address	After reset
INTM3	0	0	ES341	ES340	ES321	ES320	ES301	ES300	FFFF FC06H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

Input	ESxx1	ESxx0	Valid edge selected
INTPxx	0	0	Falling edge
	0	1	Rising edge
	1	0	(reserved)
	1	1	both edges

When using Pxy/INTPn as interrupt input, the port control registers PMCx and PMx should be set to enable the alternate function of the Pxy/INTPn pin.

5.4 Software Exception

A software exception is generated when the CPU executes the TRAP instruction, and is always accepted.

- TRAP instruction format: TRAP vector (where vector is 0 to 1FH)

For details of the instruction function, refer to the **V850E Family User's Manual Architecture.**

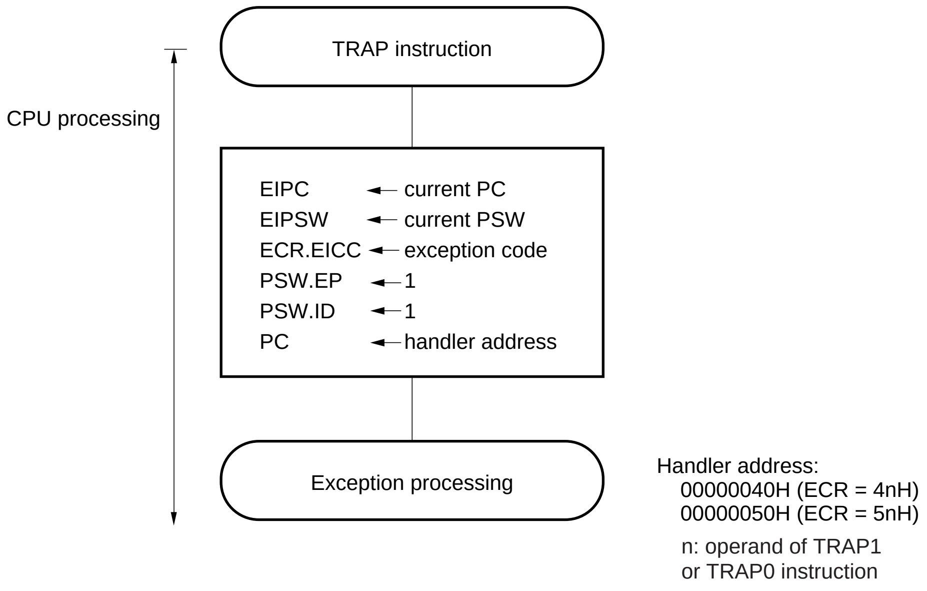
5.4.1 Operation

If a software exception occurs, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the current 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 PSW.
(5) Loads the handler address (00000040H or 00000050H) of the software exception routine in the PC, and transfers control.

How a software exception is processed is shown below.

Figure 5-22: Software Exception Processing



5.4.2 Restore

To restore or return execution from the software exception service routine, the RETI instruction is used.

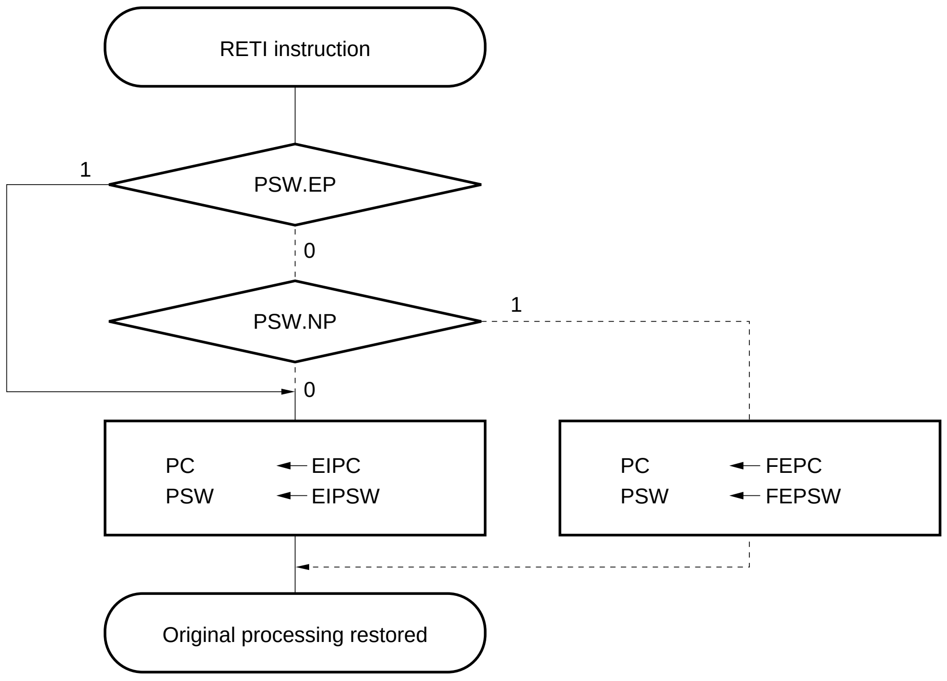
Operation of RETI instruction

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 PC and PSW from EIPC and EIPSW because the EP bit of PSW is 1.
(2) Transfers control to the address of the restored PC and PSW.

The processing of the RETI instruction is shown below.

Figure 5-23: RETI Instruction Processing



Caution: When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during the software exception process, 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.

5.4.3 EP flag

The EP flag in PSW is a status flag used to indicate that exception processing is in progress. It is set when on exception occurs, and the interrupt is disabled.

Figure 5-24: EP Flag (EP)

Symbol	31	8	7	6	5	4	3	2	1	0	After Reset
PSW	0		NP	EP	ID	SAT	CY	OV	S	Z	0000 0020H

EP	Exception Processing
0	Exception processing is not in progress
1	Exception processing is in progress

5.5 Exception Trap

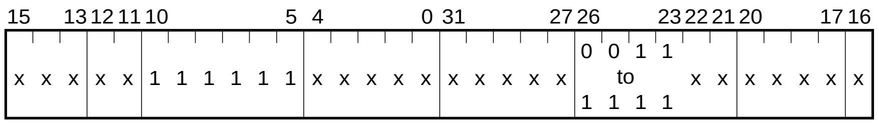
The exception trap is an interrupt that is requested when illegal execution of an instruction takes place. In the V850E/CA4 HELIOS, an illegal op code exception (ILGOP: ILeGal OPcode trap) is considered as an exception trap.

- Illegal op code exception: occurs if the sub op code field of an instruction to be executed next is not a valid op code.

5.5.1 Illegal op code definition

An illegal op code is defined to be a 32-bit word with bits 5 to 10 being 111111B and bits 23 to 26 being 0011B to 1111B.

Figure 5-25: Illegal op Code



Remark: x: don’t care

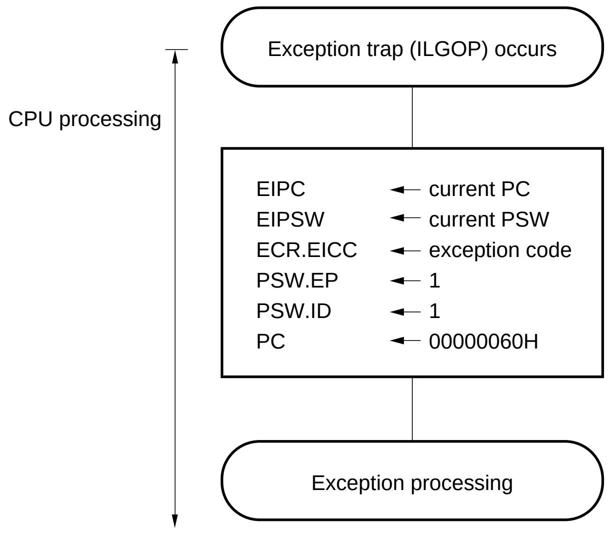
5.5.2 Operation

If an exception trap occurs, the CPU performs the following processing, and transfers control to the handler routine:

(1) Saves the current PC to EIPC.
(2) Saves the current PSW to EIPSW.
(3) Writes an exception code (0060H) to the lower 16 bits (EICC) of ECR.
(4) Sets the EP and ID bits of PSW.
(5) Loads the handler address (00000060H) for the exception trap routine to the PC, and transfers control.

How the exception trap is processed is shown below.

Figure 5-26: Exception Trap Processing



5.5.3 Restore

To restore or return execution from the exception trap, the RETI instruction is used.

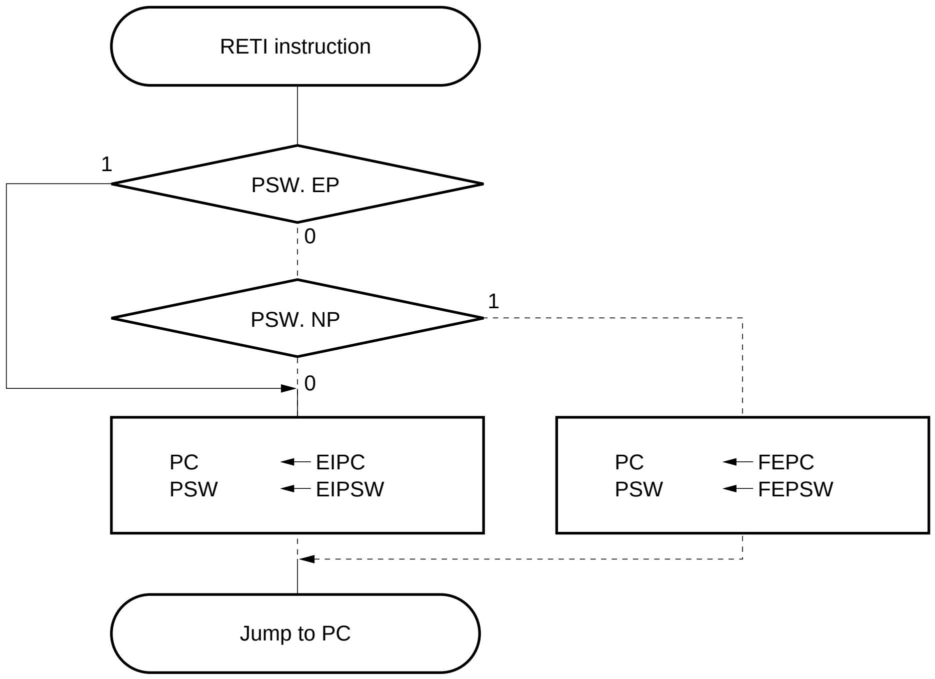
Operation of RETI instruction

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 PC and PSW from EIPC and EIPSW because the EP bit of PSW is 1.
(2) Transfers control to the address of the restored PC and PSW.

The processing of the RETI instruction is shown below.

Figure 5-27: RETI Instruction Processing



Caution: When the PSW.EP bit and the PSW.NP bit are changed by the LDSR instruction during the exception trap process, 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.

5.6 Priority Control

5.6.1 Priorities of interrupts and exceptions

Table 5-2: Priorities of Interrupts and Exceptions

	RESET	NMI	INT	TRAP	ILGOP
RESET		∗	∗	∗	∗
NMI	×		←	←	←
INT	×	↑		←	←
TRAP	×	↑	↑		←
ILGOP	×	↑	↑	↑	

RESET:	reset
NMI:	non-maskable interrupt
INT:	maskable interrupt
TRAP:	software exception
ILGOP:	illegal op code exception
∗:	Item on the left ignores the item above.
×:	Item on the left is ignored by the item above.
↑:	Item above is higher than the item on the left in priority.
←:	Item on the left is higher than the item above in priority.

5.6.2 Multiple interrupt processing

Multiple interrupt servicing is a function that allows the nesting of interrupts. If a higher priority interrupt is generated and acknowledged, it will be allowed to stop a current interrupt service routine in progress. Execution of the original routine will resume once the higher priority interrupt routine is completed.
If an interrupt with a lower or equal priority is generated and a service routine is currently in progress, the later interrupt will be kept pending.
Multiple interrupt servicing control is performed when the processor is in the interrupt enable state (ID = 0). Even in an interrupt servicing routine, this state must be entered by executing an EI instruction, which sets ID = 0. If multiple nested interrupts are allowed, EIPC and EIPSW must be saved at the entry of the interrupt handler.
The following example shows the procedure of interrupt nesting.

(1) To acknowledge maskable interrupts in service program

Service program of maskable interrupt or exception

...
...
- Saves EIPC to memory or register
- Saves EIPSW to memory or register
- EI instruction (enables interrupt acknowledgement)

...
...

← Acknowledges interrupt such as INTP input.

- DI instruction (disables interrupt acknowledgement)
- Restores saved value to EIPSW
- Restores saved value to EIPC
- RETI instruction

(2) To generate exception in service program

Service program of maskable interrupt or exception

```
        ...
        ...
• Saves EIPC to memory or register
• Saves EIPSW to memory or register
• EI instruction (enables interrupt acknowledge-
  ment)
        ...
• TRAP instruction                              ← Acknowledges exception such as TRAP instruction.
• Illegal op code                               ← Acknowledges exception such as illegal op code.
        ...
• Restores saved value to EIPSW
• Restores saved value to EIPC
• RETI instruction
```

Priorities 0 to 7 (0 is the highest) can be programmed for each maskable interrupt request for multiple interrupt processing control. To set a priority level, write values to the xxnPR0 to xxnPR2 bits of the interrupt request control register (xxnIC) corresponding to each maskable interrupt request. At reset, the interrupt request is masked by the xxnMK bit, and the priority level is set to 7 by the xxnPR0 to xxnPR2 bits.

Priorities of maskable interrupts

(High) Level 0 > Level 1 > Level 2 > Level 3 > Level 4 > Level 5 > Level 6 > Level 7 (Low)

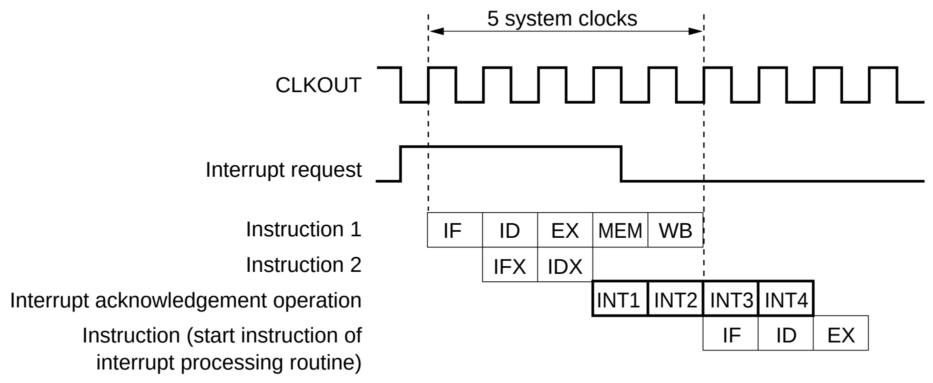
Interrupt servicing that has been suspended as a result of multiple interrupt servicing is resumed after the interrupt servicing of the higher priority has been completed and the RETI instruction has been executed.
A pending interrupt request is acknowledged after the current interrupt servicing has been completed and the RETI instruction has been executed.

Caution: In the non-maskable interrupt servicing routine (time until the RETI instruction is executed), maskable interrupts are not acknowledged but are suspended.

5.7 Interrupt Latency Time

The following table describes the interrupt latency time (from interrupt request generation to start of interrupt servicing).

Figure 5-28: Pipeline Operation at Interrupt Request Acknowledgement



Remark:		
	INT1 to INT4:	interrupt acknowledge processing
	IFX:	invalid instruction fetch
	IDX:	invalid instruction decode

Interrupt Latency Time (system clock)			Condition
	Internal interrupt	External interrupt	
Minimum	5	5 + analog delay time	Time to eliminate noise (analog delay) is also necessary for external interrupts, except when: • In IDLE/software STOP mode • External bit access • Two or more interrupt request non-sample instructions are executed in succession • Access to interrupt control register
Maximum	11	11 + analog delay time	

5.8 Periods Where Interrupt Is Not Acknowledged

An interrupt is acknowledged while an instruction is being executed. However, no interrupt will be acknowledged between interrupt non-sample instruction and next instruction.

Interrupt request non-sample instruction

- EI instruction
- DI instruction
- LDSR reg1, 0x5 instruction (Load PSW with reg1 value)
- LD.B reg2, PHCMD (write access to register PHCMD)
- LD.B reg3, PRCMD (write access to register PRCMD)

[MEMO]

Chapter 6 Clock Generator

6.1 Features

The clock generator is the circuit that generates the clock pulses supplied to the CPU and the peripheral hardware. It consists mainly of an oscillator, a PLL synthesizer, dividers and a controller.

The oscillator is able to generate a clock from an external ceramic or crystal resonator, in a range from 6 to 8 MHz. When the oscillator is stopped, as in STOP mode, and restarted, a stabilization time is insured by an on-chip counter.

The PLL synthesizer is designed to multiply by 8 the external frequency input at the X1 pin, and can supply the chip with an internal clock of up to 32 MHz.
When the system is reset, the PLL multiplier is disabled, and should be started by software to allow high-speed operation.

The controller allows the selection of the various prescalers/dividers and manages the switching between operating and standby modes.

The features of the clock generator are shown below:

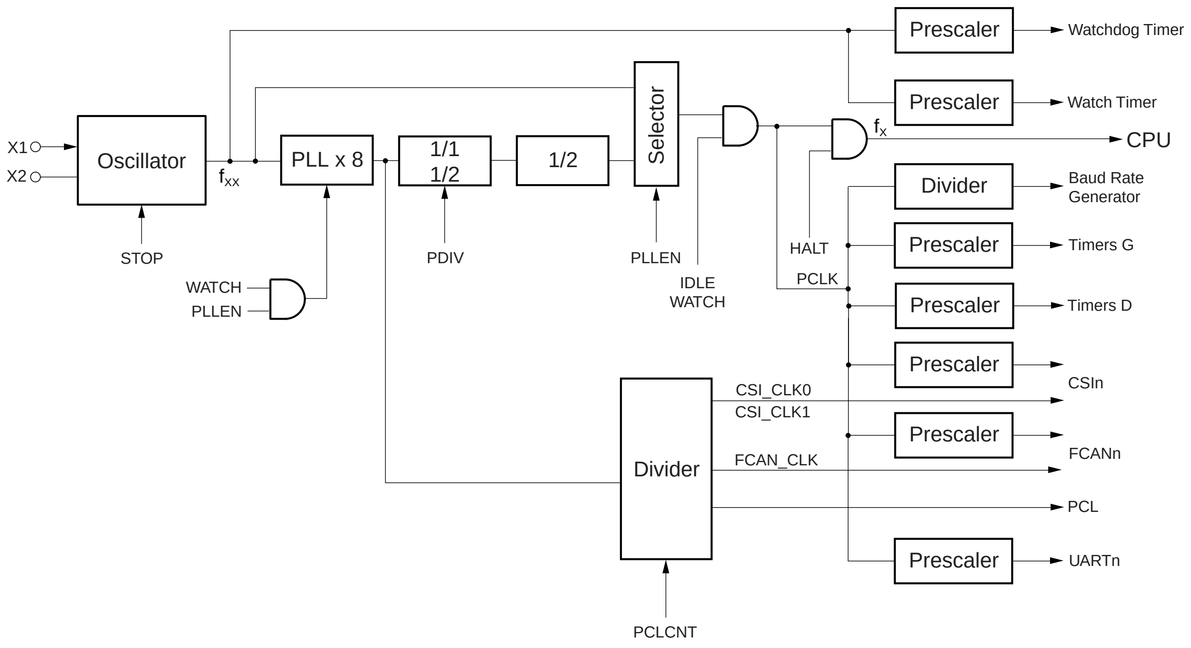
- Oscillator for external resonator
- Multiplier using PLL synthesizer
- Oscillator stabilization time counter
- Dividers for on-chip peripherals
- Power saving control
  - HALT mode
  - IDLE Mode
  - WATCH mode
  - STOP mode

These operating and standby modes can be combined and switched to suit the target application, and enable effective implementation of low-power systems.

6.2 Configuration

The following figure shows a functional description of the clock generator.

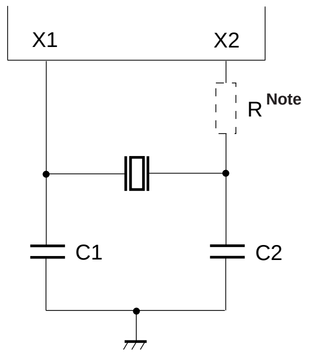
Figure 6-1: Clock Generator Configuration



6.3 Clock Oscillator

The system clock oscillator oscillates with a crystal or ceramic resonator connected to the X1 and X2 pins. A maximum 32 MHz system clock can be generated by connecting a 8 MHz crystal resonator or ceramic resonator to the X1 and X2 pins.

Figure 6-2: Main System Clock Oscillator



Note: Resistor R is optional and depends on resonator supplier.

6.4 Control Registers

The clock generator has the following control registers:

- CKC (Clock Control Register)
- PDIV (PLL Divider Factor Register)
- PCLCNT (PCL and peripheral clock Control Register)

and the following status register:

- PSTAT (PLL Status Register)

In addition the standby modes are controlled by the following control registers:

- PSM (Power Save Mode Register)
- PSC (Power Save Control Register)

CKC and PSC are specific registers that can only be written using a specific sequence of instructions. Please, refer to Chapter 3.2.3 ”Special registers” on page 62 for a description of these specific registers access method.

6.4.1 Clock control register CKC

This is an 8-bit register that controls the internal system clock frequency in enabling the PLL.
As the several stages of the clock generation need stabilization time and synchronisation, HELIOS is equipped with one timer, the Time Base Counter, which “freezes” the clock delivery to the chip while the clock is not ready. The choice between 2 time values for this Time Base Counter is possible.
CKC is a specific register, and can be written only after a write access to the PHCMD register (see section 3.2.3 ”Special registers” on page 62). If an illegal store operation to this specific register takes place, it can be checked by the PERR flag of the status register (PHS).

This register can be read/written in 8-bit or 1-bit units.

Figure 6-3: Clock Control Register CKC

Symbol	7	6	5	4	3	2	1	0	Address	At Reset
CKC	PLLEN	0	TBCS	0	0	0	0	0	FFFF F822H	00H
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

PLLEN	PLL enable
0	PLL disable mode (direct mode): system clock is f_{XX}
1	PLL enable mode: after the PLL lock-up time has passed, the system clock is automatically switched to PLL output (4 × f_{XX} or 2 × f_{XX}, depending on PDIV0)

TBCS	Time base count (oscillation stabilization time) select					
	WATCH mode release			STOP mode release		
	Number of f_{XX} periods	Count Time		Number of f_{XX} periods	Count Time	
		f_{XX}=6.00MHz	f_{XX}=8.00MHz		f_{XX}=6.00MHz	f_{XX}=8.00MHz
0	5300	0.88 ms	0.66 ms	50300	8.4 ms	6.3 ms
1	10600	1.77 ms	1.33 ms	100600	16.8 ms	12.6 ms

Caution: When releasing the WATCH mode, TBCS must be 1, so set TBCS = 1 before entering the WATCH mode.

6.4.2 PLL divider factor register PDIV

This register enables the internal PLL divider by 2, and can be used to scale the system frequency. PDIV must be set before PLLEN.
This register can be read/written with 1-bit or 8-bit memory access.
Bit 1 is read as 0, and should be written with 0.

Figure 6-4: PLL Divider Factor Register PDIV

Symbol	7	6	5	4	3	2	1	0	Address	At Reset
PDIV	0	0	0	0	0	0	0	PDIV0	FFFFF842H	00H
R/W	R	R	R	R	R	R	R/W	R/W		

PDIV0	PLL divider factor
0	If PLLEN = 1, system clock is PLL output ($f_{XX} \times 4$)
1	If PLLEN = 1, system clock is PLL output divided by 2 ($f_{XX} \times 2$)

6.4.3 PCL and peripheral clock control register PCLCNT

It is possible to output the CPU Clock on pin P02/INTP02/PCL. The processor clock output frequency is equal to the selected CPU frequency f_{CPU}. Corresponding ports registers have to be set accordingly: PMC02 bit of PMC0 must be set to 1.
(Please refer to Chapter 13 ”Port Feature” on page 437.)

When setting HALT mode, PCL still operates. In WATCH, IDLE and STOP mode, PCL is disabled.

PCLCNT controls the PLL divider, and can be used to set the PCL clock output frequency, and to generate some clock signals dispatched to FCAN, CSI0, CSI1 and CSI10 clock controllers.
This register can be read/written with 8 bits memory access.

Figure 6-5: PCL and Peripheral Clock Control Register PCLCNT (1/2)

	7	6	5	4	3	2	1	0	Address	At Reset
PCLCNT	CDEN	0	CSIPCL SEL2	CSIPCL SEL1	CSIPCL SEL0	PCL SEL	FCAN SEL1	FCAN SEL0	FFFF F840H	00H
	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W		

CDEN	Clock Divider Enable
0	Clock divider for FCAN, CSI0, CSI1, CSI10 and PCL is disabled
1	Clock divider for FCAN, CSI0, CSI1, CSI10 and PCL is enabled

Note: It is better to start at first the PLL, before enable the clock divider via the CDEN bit.

CSICPL SEL2	CSICPL SEL1	CSICPL SEL0	CSI_CLK0 (MHz)		CSI_CLK1 (MHz)		PCL SEL	PCL (MHz)	
			f_{XX}=8 MHz	f_{XX}=6 MHz	f_{XX}=8 MHz	f_{XX}=6 MHz		f_{XX}=8 MHz	f_{XX}=6 MHz
0	X	0	-	8	-	-	0	10.67	8
							1	21.33	16
0	X	1	8	6	-	-	0	8	6
							1	16	12
1	0	X	5.33	4	-	8	0	5.33	4
							1	10.67	8
1	1	X	4	3	8	6	0	4	3
							1	8	6

Remark: X: don’t care

Figure 6-5: PCL and Peripheral Clock Control Register PCLCNT (2/2)

FCANSEL1	FCANSEL0	FCAN_CLK (MHz)	
		f_{XX}=8 MHz	f_{XX}=6 MHz
0	0	PCLK	PCLK
0	1	-	16
1	0	PCLK	PCLK
1	1	16	-

Empty cells represent a combination of settings that does not insure correct operation.

Note: PCLK is the peripheral clock and is internally connected to the CPU clock

Caution: the input clock of the FCAN interface should not be greater than 16 MHz. Connecting the FCAN clock to the CPU clock (alias the Peripheral Clock) is provided for use when the PLL is disabled: in this case PCLK is the external oscillator clock. When the PLL is enabled, set (FCANSEL1, FCANSEL0) = (1, 1) when running @ 8 MHz, and to (0, 1) when running @ 6 MHz.

6.4.4 PLL status register PSTAT

This register holds the status of the PLL.
Following a power-on reset or when exiting the STOP or WATCH mode, an amount of time will be required for the PLL to stabilize before using any of the HELIOS hardware functions which rely on clock. This required time is called PLL lock-up time: the status in which the frequency is not stable is called unlock status and the status in which it has been stabilized is called lock status.
There is a VBSTAT flag in the PSTAT register which reflects the PLL's frequency stabilization state. Static processing such as setting of the on-chip hardware units and initialization of the register data and memory data, however, can be executed before the VBSTAT flag is set.

This register can be read only in 8-bit units.

Figure 6-6: PLL Status Register PSTAT

Symbol	7	6	5	4	3	2	1	0	Address	At Reset
PSTAT	VBSTAT	0	0	0	0	0	0	0	FFFF F824H	00H
R	R	R	R	R	R	R	R	R		

VBSTAT	PLL status
0	PLL unlocked: system clock is f_{XX}
1	PLL locked: system clock is PLL output ($f_{XX} \times 4$ or $f_{XX} \times 2$) if PLLEN = 1

6.4.5 Power save mode register PSM

This is a 8-bit register to specify power save mode. Read/Write is possible by 8-bit or 1-bit memory operation instructions. All contents are set to “00H” by $\overline{\text{RESET}}$ input and bit 2, 3 and 7 are fixed to 0 by hardware.

Figure 6-7: Power Save Mode Register Format

Symbol	7	6	5	4	3	2	1	0	Address	At Reset
PSM	0	0Note	0Note	0Note	0	0	PSM1	PSM0	FFFF F820H	00H
R/W	R	R/W	R/W	R/W	R	R	R/W	R/W		

Note: Bits 6, 5, 4 should be always overwritten by 0. If this value is set to “1”, then proper operation cannot be guaranteed.

PSM1	PSM0	Selection of Standby Mode This selection is valid when STB bit (PSC.1) of PSC register has been set to "1".
0	0	IDLE Mode
0	1	STOP mode
1	0	WATCH Mode
1	1	Setting prohibited

6.4.6 PSC - power save control register

This register controls entering and leaving power save mode. This register is a specific register: any write access can only be done just after a dummy write to PRCMD register (see Chapter 3.2.3 ”Special registers” on page 62).
Read access to PSC is possible without any write access to PRCMD. PSC register must be written with store instruction execution by CPU only.
Depending of the setting of the PSM1 and PSM0 bits of PSM register, either WATCH, STOP or IDLE mode will be set. Standby mode can be released by either unmasked INT, NMI or $\overline{\text{RESET}}$ signal.
This register can be read in 8-bit and 1-bit units.

Figure 6-8: Power Save Control Register Format

	7	6	5	4	3	2	1	0	Address	After Reset
PSC	NMIM2	0	NMIM0	INTM	0	0	STB	0	FFFF F1FEH	00H
	R/W	R/W	R/W	R/W	R	R	R/W	R		

STB	Standby mode enable
0	Normal mode
1	Set standby mode WATCH, STOP or IDLE, according to PSM register

Bit Name	Wake-up Sources Mask
NMIM2	Wake-up from non-maskable interrupt request from Watchdog timer 0: enabled 1: disabled
NMIM0	Wake-up from non-maskable interrupt request from NMI pin 0: enabled 1: disabled
INTM	Wake-up from maskable interrupt request (INTn-INT0) from INTPn-INTP0 pin 0: enabled 1: disabled

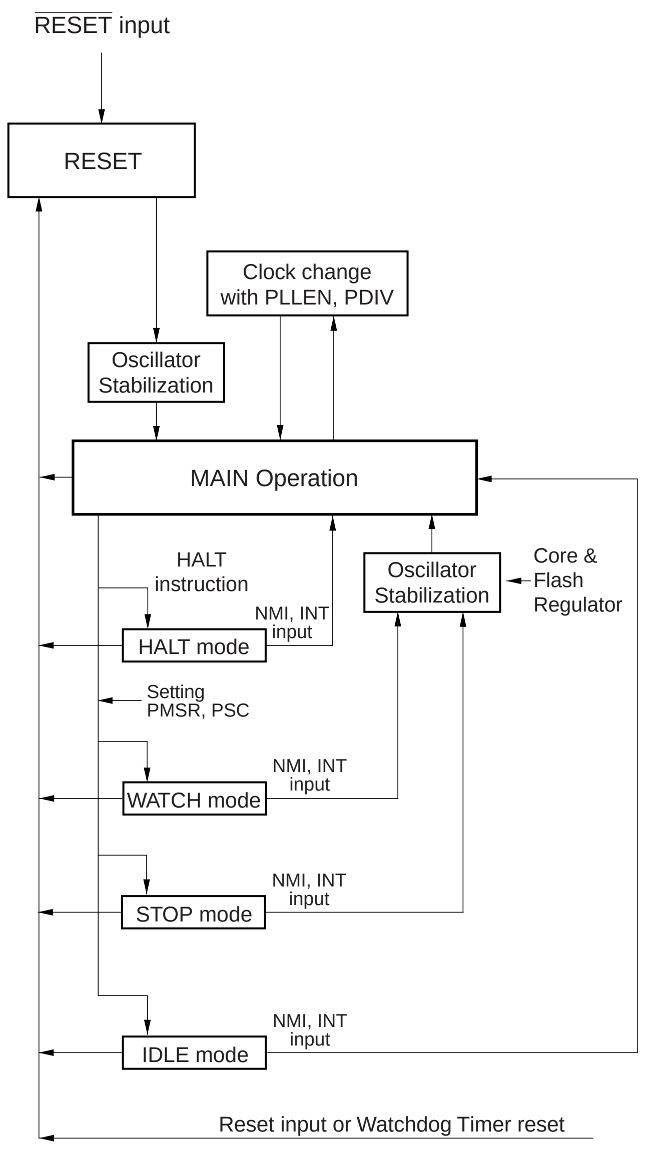
Remark: If INTM, NMIM2, NMIM0 and STB bit in the PSC register are set simultaneously, settings for INTM, NMIM2 and NMIM0 are ignored. If INTM, NMI2 or NMIM0 bit have to be modified to enter power save mode, set them before setting STB bit.

6.5 System Operating Modes

HELIOS can operate in the following modes:

- Main clock operation
- HALT mode
- IDLE mode
- WATCH mode
- STOP mode

Figure 6-9: System Operating and Standby Modes Block Diagram



Note: It is necessary to switch off the PLL before you enter Halt and Idle mode.

6.5.1 Main clock operation mode

In this mode, the clock oscillator and the PLL output can supply the CPU. CPU frequency is selected through PDIV0 bit of PDIV register. Available clock frequencies are f_{XX}, $f_{XX} \times 2$, $f_{XX} \times 4$.
Corresponding settings are described in section 3.2.3 "Special registers" on page 62.

After reset signal is released, device always starts operating with f_{XX} system clock after oscillation stabilization time elapsed, and with the PLL disabled.

6.5.2 HALT mode

In this mode, the supply of the operating clock to the CPU is stopped, and on-chip peripherals continue to operate. Combining this mode with the normal operating mode to provide intermittent operations enables the overall system power consumption to be reduced.

In the HALT mode, program execution is stopped but the contents of all registers and internal RAM 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 6-1: Operating Status in HALT Mode

Item		Status
Main oscillator		Operates
PLL controller		Operates
CPU		Stopped
Serial Communications	UART6n	Operates
	CSIn	Operates
	FCANn	Operates
Timers	Watchdog Timer	Operates
	Watch Timer	Operates
	Timer Gn	Operates
	Timer Dn	Operates
A/D converter		Operates
Ports		Unchanged
External interrupts		Operate
PCL		Operates
Internal data		Retains all internal data before entering HALT mode, such as CPU registers, status, data, and on-chip RAM.

(1) Entering HALT mode

This mode is entered by executing the dedicated instruction (HALT) and can be released by any interrupt request.

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. This is the reason why the implementation of 5 NOP instructions following the HALT instruction is recommended.

(2) HALT mode release

The HALT mode is released by an unmasked maskable interrupt request, a NMI request, $\overline{\text{RESET}}$ signal input or Watchdog timer reset.

(a) Release by interrupt or NMI request

The HALT mode is released unconditionally by an

- unmasked maskable interrupt request, regardless of its priority level, and regardless of the INTM bit in the PSC register.
- an unmaskable interrupt request, regardless NMIM0 or NMIM2 bits in the PSC register.

However, depending on the interrupt enable status and on the current interrupt level, the operation differs on interrupt priority levels as follows:

- If the priority level of incoming interrupt is lower than the priority level of the current interrupt: the incoming interrupt is not acknowledged: the HALT mode is released, the incoming interrupt request is kept pending, and the program continues in sequence, after the HALT instruction.
- If the priority level of incoming interrupt is higher than the priority level of the current interrupt (NMI has the highest priority):
  the incoming interrupt request is acknowledged: the HALT mode is released, then
  - if PSW.ID = 0 (the interrupt state is: enable)
    the program branches to the vector address of the incoming interrupt, and the RETI of the incoming interrupt handler returns to the instruction following the HALT instruction.
  - if PSW.ID = 1 (the interrupt state is: disable)
    the program branches to the instruction following the HALT instruction.

Table 6-2: Operation after Releasing HALT Mode by Interrupt Request

Release Source	Interrupt Enabled (EI) Status	Interrupt Disabled (DI) Status
Non-maskable interrupt request	Execution branches to the NMI handler address	
Maskable interrupt request (higher priority)	Execution branches to the interrupt handler address	The next instruction is executed
Maskable interrupt request (lower or same priority)	The next instruction is executed	The next instruction is executed

(b) Release by Watchdog timer reset

The device operation is the same as normal reset operation.

(c) Release by $\overline{\text{RESET}}$ pin

The device operation is the same as normal reset operation.

6.5.3 IDLE mode

In this mode, the supply of the operating clock to the CPU is stopped, and some on-chip peripherals continue to operate. Combining this mode with the normal operating mode to provide intermittent operations enables the overall system power consumption to be reduced.

In the IDLE mode, program execution is stopped but the contents of all registers and internal RAM 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 IDLE mode is tabulated below.

Table 6-3: Operating Status in IDLE Mode

Item		Status
Main oscillator		Operates
PLL controller		Operates
CPU		Stopped
Serial Communications	UART6n	Stopped
	CSIn	Operates in slave mode Operates with PLL DIVIDER clock
	FCANn	Operates if PLL DIVIDER clock (PCLCNT register) has been selected otherwise it is stopped
Timers	Watchdog Timer	Operates
	Watch Timer	Operates
	Timer Gn	Operates in counter mode
	Timer Dn	Stopped
A/D converter		Stopped
Ports		Unchanged
External interrupts		Operate
PCL		Operates if PLL DIVIDER clock (PCLCNT register) has been selected otherwise it is stopped (Low level)
Internal data		Retains all internal data before entering IDLE mode, such as CPU registers, status, data, and on-chip RAM.

(1) Entering IDLE mode

The IDLE mode is entered by writing to the PSC register and can be released by any interrupt request.

To enter this mode, the bit 0 and 1 of the PSM register must be set to "00". Then PSC register should be set with store (ST/SST) instruction or bit manipulation instructions just after a write access to the PRCMD register.
Refer to Chapter 3.2.3 ”Special registers” on page 62.

(2) IDLE mode release

The IDLE mode is released by an unmasked maskable interrupt request, a NMI request, $\overline{\text{RESET}}$ signal input or Watchdog timer reset.
After IDLE mode release, 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 and the first instruction is executed.

(a) Release by interrupt or NMI request

The IDLE mode is released by

- the detection of a valid edge on NMI pin (unmaskable interrupt request), if the NMIM0 bit in the PSC register is 0.
- an unmaskable interrupt request from the watchdog timer, if the NMIM2 bit in the PSC register is 0.
- unmasked maskable interrupt request, regardless of its priority level, if the INTM bit in the PSC register is 0.

However, depending on the interrupt enable status and on the current interrupt level, the operation differs on interrupt priority levels as follows:

- If the priority level of incoming interrupt is lower than the priority level of the current interrupt: the incoming interrupt is not acknowledged: the IDLE mode is released, the incoming interrupt request is kept pending, and the program continues in sequence, after the instruction which caused entering the standby mode (writing 1 to the STB bit of the PSC register).
- If the priority level of incoming interrupt is higher than the priority level of the current interrupt (NMI has the highest priority):
  the incoming interrupt request is acknowledged: the IDLE mode is released, then

  - if PSW.ID = 0 (the interrupt state is: enable)
    the program branches to the vector address of the incoming interrupt, and the RETI of the incoming interrupt handler returns to the instruction following the instruction which caused entering the standby mode (writing 1 to the STB bit of the PSC register).

  - if PSW.ID = 1 (the interrupt state is: disable)
    the program branches to the instruction following the instruction which caused entering the standby mode (writing 1 to the STB bit of the PSC register).

Table 6-4: Operation after Releasing IDLE Mode by Interrupt Request

Release Source	Interrupt Enabled (EI) Status	Interrupt Disabled (DI) Status
Non-maskable interrupt request	Execution branches to the NMI handler address	
Maskable interrupt request (higher priority)	Execution branches to the interrupt handler address	The next instruction is executed
Maskable interrupt request (lower or same priority)	The next instruction is executed	The next instruction is executed

(b) Release by Watchdog timer reset

The device operation is the same as normal reset operation.

(c) Release by $\overline{\text{RESET}}$ pin

The device operation is the same as normal reset operation.

6.5.4 WATCH mode

In this mode, the clock oscillator continues to operate.
The clock is no more supplied to the CPU and to the peripherals, but the watchdog timer and the watch timer continue to operate.

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 peripheral hardware units in WATCH mode is tabulated below.

Table 6-5: Operating Status in WATCH Mode

Item		Status
Main oscillator		Operates
PLL controller		Stopped
CPU		Stopped
Serial Communications	UART6n	Stopped
	CSIn	Operates in slave mode
	FCANn	Stopped
Timers	Watchdog Timer	Operates
	Watch Timer	Operates
	Timer Gn	Operates in counter mode
	Timer Dn	Stopped
A/D converter		Stopped
Ports		Unchanged
External interrupts		Operate
PCL		Stopped
Internal data		Retains all internal data before entering WATCH mode, such as CPU registers, status, data, and on-chip RAM.

(1) Entering WATCH mode

The WATCH mode is entered by writing 1 to STB bit of the PSC register while PSM1 = 1 and PSM0 = 0, and can be released by any interrupt request.
To enter this mode, the bit 0 of PSMR, PSM register must be set to “0”. Then PSC register should be set with store (ST/SST) instruction or bit manipulation instructions just after the write access to the PRCMD register. Refer to Chapter 3.2.3 ”Special registers” on page 62.

(2) WATCH mode release

The WATCH mode can be released by a non-maskable interrupts request, an unmasked maskable interrupt request, Watchdog timer reset or $\overline{\text{RESET}}$ signal input.
After WATCH mode release, the first instruction is executed after oscillation stabilization time set in the TBCS bit of the register CKC elapsed.

Caution: When releasing the WATCH mode, TBCS must be 1, so set TBCS = 1 before entering the WATCH mode.

(a) Release by interrupt or NMI

The WATCH mode is released by

- the detection of a valid edge on NMI pin (unmaskable interrupt request), if the NMIM0 bit in the PSC register is 0.
- an unmaskable interrupt request from the watchdog timer, if the NMIM2 bit in the PSC register is 0.
- unmasked maskable interrupt request, regardless of its priority level, if the INTM bit in the PSC register is 0.

If WATCH mode is released by interrupt or NMI, the CPU starts operation after the stabilization time set in the TBCS bit of the register CKC elapsed.

However, depending on the interrupt enable status and on the current interrupt level, the operation differs on interrupt priority levels as follows:

- If the priority level of incoming interrupt is lower than the priority level of the current interrupt: the incoming interrupt is not acknowledged: the WATCH mode is released, the incoming interrupt request is kept pending, and the program continues in sequence, after the instruction which caused entering WATCH mode (setting STP in PSC to 1)

- If the priority level of incoming interrupt is higher than the priority level of the current interrupt (NMI has the highest priority):
  the incoming interrupt request is acknowledged: the WATCH mode is released, then

  - if PSW.ID = 0 (the interrupt state is: enable)

    the program branches to the vector address of the incoming interrupt, and the RETI of the incoming interrupt handler returns to the instruction following the instruction which caused entering WATCH mode (setting STP in PSC to 1)

  - if PSW.ID = 1 (the interrupt state is: disable)

    the program branches to the instruction following the instruction which caused entering WATCH mode (setting STP in PSC to 1).

Table 6-6: Operation after Releasing Watch Mode by Interrupt Request

Release Source	Interrupt Enabled (EI) Status	Interrupt Disabled (DI) Status
Non-maskable interrupt request	Execution branches to the NMI handler address	
Maskable interrupt request (higher priority)	Execution branches to the interrupt handler address	The next instruction is executed
Maskable interrupt request (lower or same priority)	The next instruction is executed	The next instruction is executed

(b) Release by Watchdog timer reset

The device operation is the same as normal reset operation.

(c) Release by $\overline{\text{RESET}}$ pin

The device operation is the same as normal reset operation.

Figure 6-10: Watch Mode Release by NMI or INT

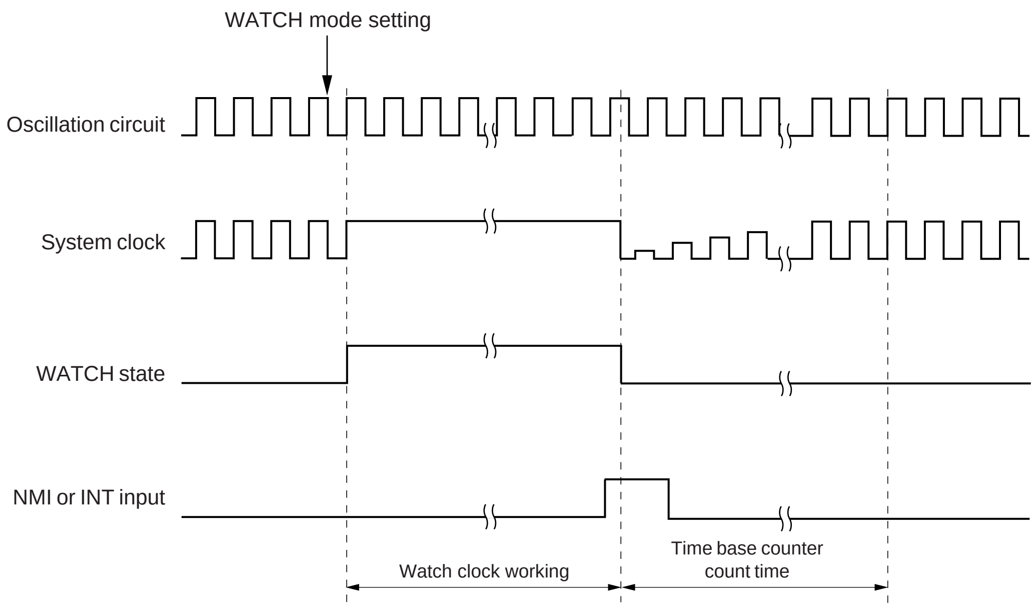
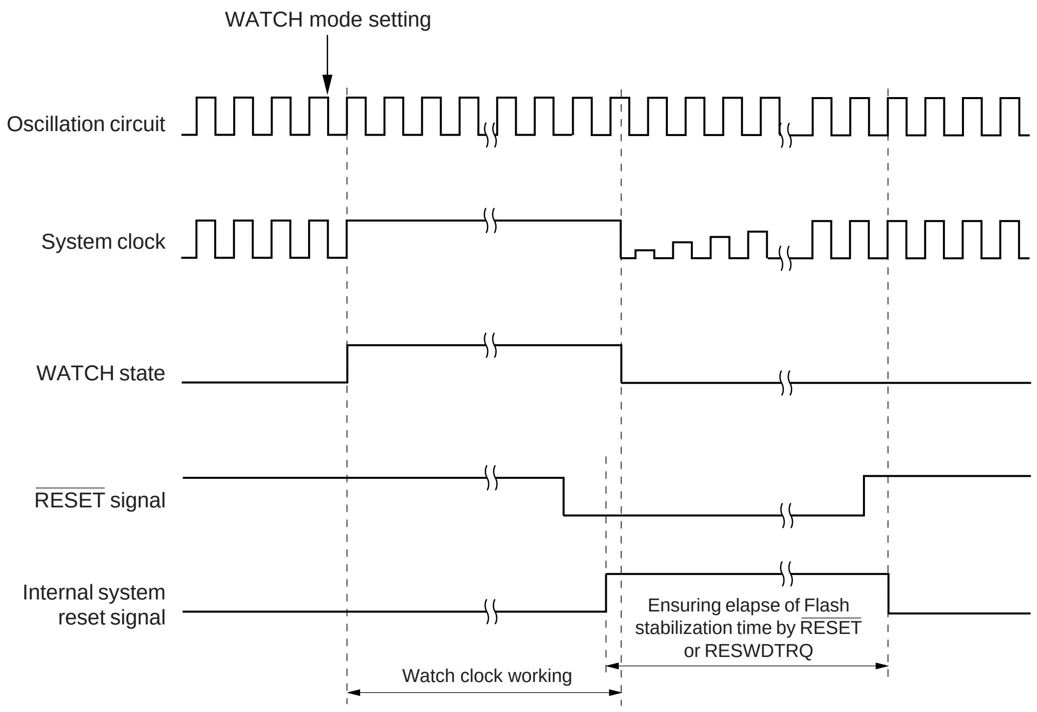


Figure 6-11: Watch Mode Release by Reset (External or Watchdog)


WATCH mode setting
Oscillation circuit
System clock
WATCH state
$\overline{\text{RESET}}$ signal
Internal system reset signal
Ensuring elapse of Flash stabilization time by $\overline{\text{RESET}}$ or RESWDTRQ
Watch clock working

6.5.5 STOP mode

In this mode, the CPU clock is stopped including the clock generators (oscillator and PLL synthesizer), resulting in stop of the entire system for ultra-low power consumption.

In this mode, the program execution stops, but the contents of all registers and internal RAM prior to entering this mode are retained. HELIOS peripherals operations are also stopped.

The state of the peripheral hardware units in STOP mode is tabulated below.

Table 6-7: Operating Status in STOP Mode

Item		Status
Main oscillator		Stopped
PLL controller		Stopped
CPU		Stopped
Serial Communications	UART6n	Stopped
	CSIn	Operates in slave mode
	FCANn	Stopped
Timers	Watchdog Timer	Stopped
	Watch Timer	Stopped
	Timer Gn	Operates in counter mode
	Timer Dn	Stopped
A/D converter		Stopped
Ports		Unchanged
External interrupts		Operate
PCL		Stopped (Low level)
Internal data		Retains all internal data before entering STOP mode, such as CPU registers, status, data, and on-chip RAM.

Caution: Please refer to the HELIOS Electrical specification (EASE-ES-2000-0.1) for minimum wake up time after STOP mode release.

(1) Entering STOP mode

To enter this mode, the bit PSM1 and PSM0 (bit 0 and bit 1 of the PSM register) must be set to "01". Then STP bit of PSC register should be set with store (ST/SST) instruction or bit manipulation instructions just after the write access to the PRCMD register.
Refer to Chapter 3.2.3 ”Special registers” on page 62.

(2) STOP mode release

The STOP mode can be released by a non-maskable interrupt request, an unmasked maskable interrupt request, Watchdog timer reset or $\overline{\text{RESET}}$ signal input.

After STOP mode release, The time base counter (TBC) is used to make sure that the time during which the oscillation of the oscillator circuit stabilizes, and flash stabilization after the software STOP mode has been released.
Select the count clock of the TBC by using the TBCS bit of the PSC register.

(a) Release by interrupt or NMI

The STOP mode is released by

- the detection of a valid edge on NMI pin (unmaskable interrupt request), if the NMIM0 bit in the PSC register is 0.
- an unmaskable interrupt request from the watchdog timer, if the NMIM2 bit in the PSC register is 0.
- unmasked maskable interrupt request, regardless of its priority level, if the INTM bit in the PSC register is 0.

If STOP mode is released by interrupt or NMI, the CPU starts operation after the stabilization time set in the TBCS bit of the register CKC elapsed.

When selecting TBC time, customer has to refer to the Electrical target specification (EASE-ES-2000-0.1) document to check what are the minimum wake up time from STOP mode release.

However, depending on the interrupt enable status and on the current interrupt level, the operation differs on interrupt priority levels as follows:

- If the priority level of incoming interrupt is lower than the priority level of the current interrupt:
  the incoming interrupt is not acknowledged: the STOP mode is released, the incoming interrupt request is kept pending, and the program continues in sequence, after the instruction which caused entering STOP mode (setting STP in PSC to 1)

- If the priority level of incoming interrupt is higher than the priority level of the current interrupt (NMI has the highest priority):
  the incoming interrupt request is acknowledged: the STOP mode is released, then

  - if PSW.ID = 0 (the interrupt state is: enable)

    the program branches to the vector address of the incoming interrupt, and the RETI of the incoming interrupt handler returns to the instruction following the instruction which caused entering STOP mode (setting STP in PSC to 1)

  - if PSW.ID = 1 (the interrupt state is: disable)

    the program branches to the instruction following the instruction which caused entering STOP mode (setting STP in PSC to 1)

Table 6-8: Operation after Releasing Stop Mode by Interrupt Request

Release Source	Interrupt Enabled (EI) Status	Interrupt Disabled (DI) Status
Non-maskable interrupt request	Execution branches to the NMI handler address	
Maskable interrupt request (higher priority)	Execution branches to the interrupt handler address	The next instruction is executed
Maskable interrupt request (lower or same priority)	The next instruction is executed	The next instruction is executed

(b) Release by Watchdog timer reset

The device operation is the same as normal reset operation.

(c) Release by $\overline{\text{RESET}}$ pin

The device operation is the same as normal reset operation.

Figure 6-12: Stop Mode Release by NMI or INT

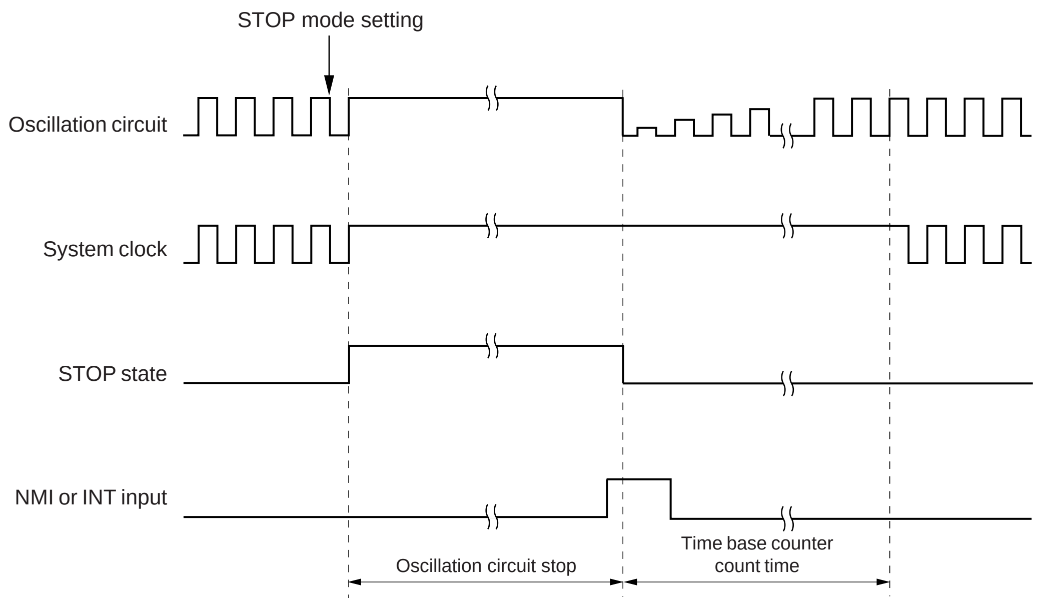
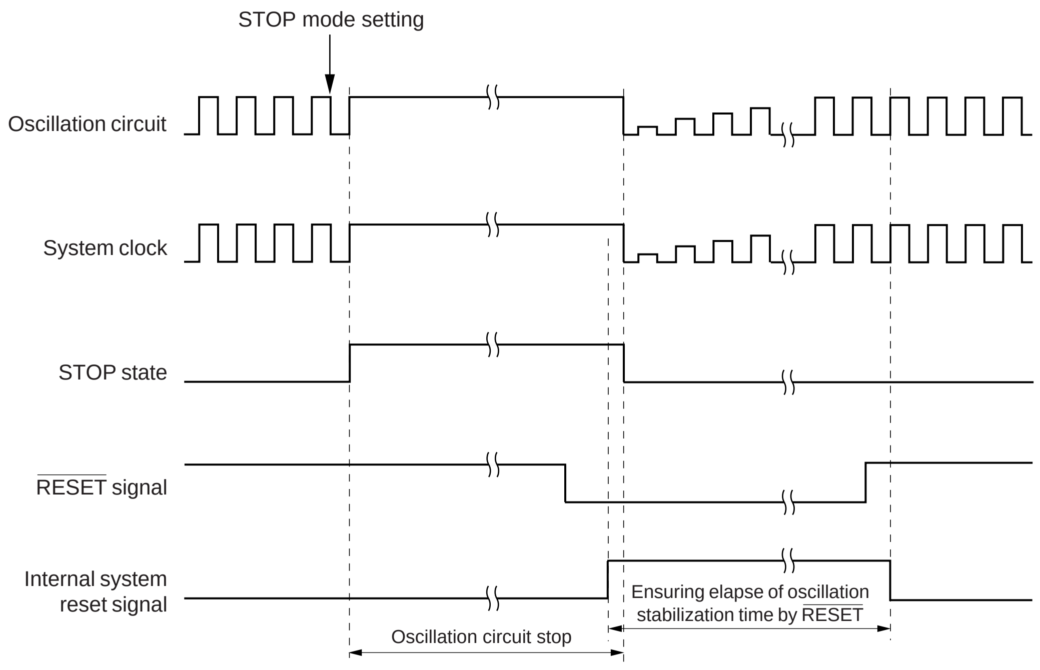


Figure 6-13: Stop Mode Release by Reset (External or Watchdog)


STOP mode setting
Oscillation circuit
System clock
STOP state
$\overline{\text{RESET}}$ signal
Internal system reset signal
Ensuring elapse of oscillation stabilization time by $\overline{\text{RESET}}$
Oscillation circuit stop

6.5.6 Securing Oscillation Stabilization Time

- When the HALT mode is released, no stabilization times are needed before execution of the first instruction, as the oscillator and the PLL were not stopped.

- When the IDLE mode is released, no stabilization times are needed before execution of the first instruction, as the oscillator and the PLL were not stopped.

- When the WATCH mode is released, no oscillation stabilization time is needed before execution of the first instruction, as the oscillator was not stopped. On the other hand a Flash power stabilization time is needed.
  - If the WATCH mode is released by the reset input, the Flash power stabilization time has to be insured by the user hardware, in setting the $\overline{\text{RESET}}$ pin low for at least 1 ms.
  - If the WATCH mode is released by an interrupt (NMI or maskable), the Flash power stabilization time is secured by the internal counter TBC.

- When the STOP mode is released, an oscillation stabilization time is needed before execution of the first instruction, as the oscillator was stopped.
  - If the STOP mode is released by the reset input, this oscillation stabilization time has to be insured by the user hardware, in setting the $\overline{\text{RESET}}$ pin low for at least 10 ms. The Flash power stabilization time is overlapped with the oscillation stabilization time.
  - If the STOP mode is released by an interrupt (NMI or maskable), this oscillation stabilization time is secured by the internal counter TBC.

After a power-up reset, as at STOP mode release by reset, the oscillation stabilization time has to be insured by the $\overline{\text{RESET}}$ pin, which should be held low for at least 10 ms after V_{DD} presence. The Flash power stabilization time is overlapped with the oscillation stabilization time.

[MEMO]

Chapter 7 Timer

7.1 Timer D

2 × 16-bit interval timer of Timer D are implemented:

- Timer D0
- Timer D1

7.1.1 Features Timer D

Timer Dn (TMD) functions as a 16-bit interval timer.

7.1.2 Function overview Timer Dn

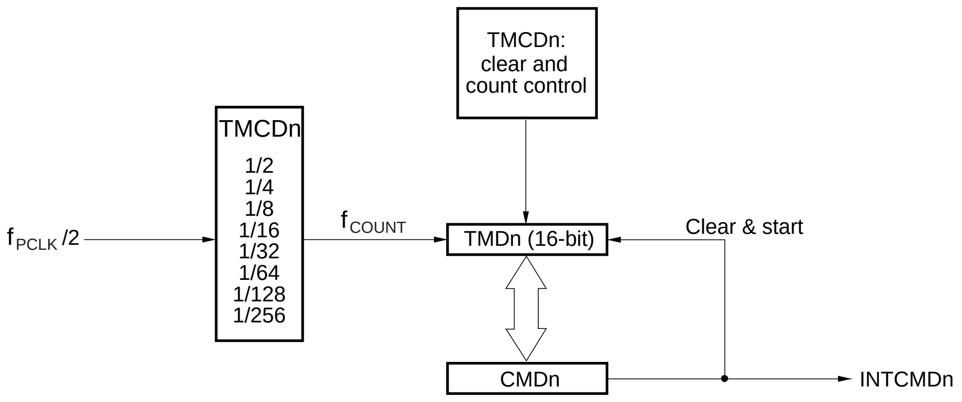
- Compare register: 1
- Count clock selected from divisions of internal peripheral clock
  (maximum frequency of count clock: $f_{PCLK}/4$ (8 MHz @ f_{XX} = 8 MHz))
- Prescaler division ratio
  8 division ratios can be selected related to the internal peripheral clock (f_{PCLK}). The range is from $f_{PCLK}/4$ to $f_{PCLK}/512$.
- Interrupt request sources: 1
  - Compare match interrupt
  INTCMDn 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. In HELIOS f_{PCLK} is the same frequency as f_{CPU}:
  external quartz frequency f_{XX} if PLLEN = 0,
  $f_{XX} \times 4$ if PLLEN = 1 and PDIV = 0,
  $f_{XX} \times 2$ if PLLEN = 1 and PDIV = 1. (see Chapter 6 ”Clock Generator” on page 145).

Figure 7-1 shows the block diagram of the channel of Timer Dn.

Figure 7-1: Block Diagram of Timer Dn (n = 0, 1)



Remark: n = 0, 1

7.1.3 Basic configuration

Table 7-1: 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}/4, f_{PCLK}/8, f_{PCLK}/16, f_{PCLK}/32, f_{PCLK}/64, f_{PCLK}/128, f_{PCLK}/256, f_{PCLK}/512	TMDn	R	–	–	–	–
		CMDn	R/W	INTCMDn	–	–	–
		TMCDn	R/W	–	–	–	–

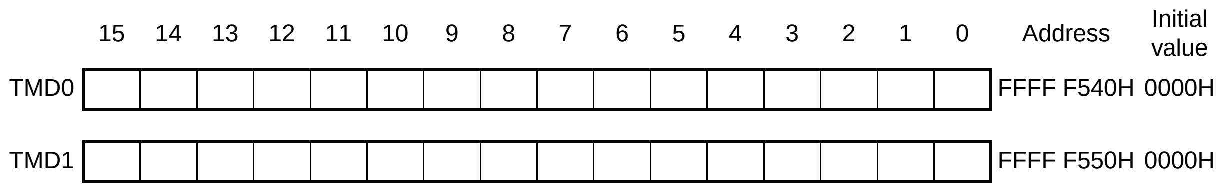
Remark: f_{PCLK}: internal peripheral clock

(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}/4 and f_{PCLK}/512 in 8 steps by the CSn2 to CSn0 bits of the TMCDn register.

TMDn is read-only in 16-bit units.

Figure 7-2: Timer Dn Counter Register (TMDn) (n = 0, 1)



The conditions for which the TMDn register becomes 0000H are shown below:

- Reset input
- CAEn bit = 0
- CEn bit = 0
- Match of TMDn register and CMDn register
- Overflow

(2) Timer Dn compare register (CMDn) (n = 0, 1)

CMDn and the TMDn registers' count value are compared, and an interrupt request signal (INTCMDn) is generated when a match occurs. Synchronized with this match, TMDn is cleared.
If the CAEn 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 synchronisation with the count clock. In the 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 7-3: 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

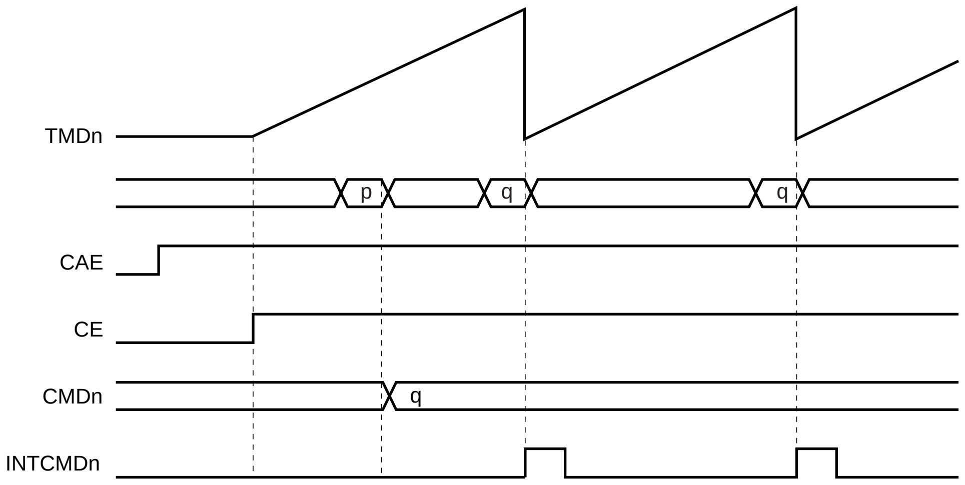
Cautions: 1. A write operation to the a CMDn register requires 2 f_{PCLK}/2 clocks until the value that was set in the CMDn register is transferred to internal units. When writing continuously to the CMDn register, be sure to reserve a time interval of at least 2 f_{PCLK}/2 clocks.

2. Writing to the CMDn register is allowed only once during a timer cycle: only the first value written to the CMDn register during a timer cycle (from 0000H until an INTCMDn interrupt is generated due to a match of the TMDn register and CMDn register) is effective.

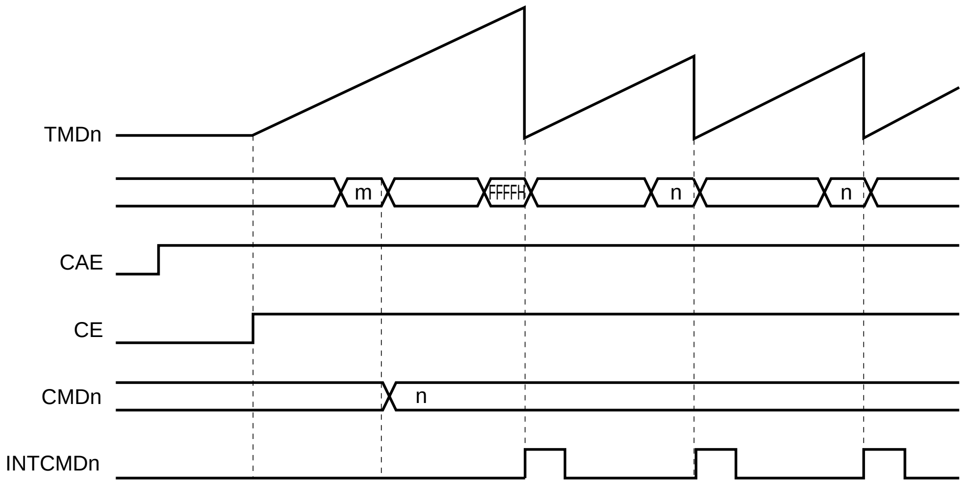
3. Note that an INTCMDn interrupt will be generated after an overflow if a value less than the counter value is written in the CMDn register during TMDn register operation
(Figure 7-4, "Timing of Timer Dn Operation," on page 175).

Figure 7-4: 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

7.1.4 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 7-5: Timer Dn Control Register (TMCDn) (n = 0, 1)

	7	6	5	4	3	2	1	0	Address	Initial value
TMCD0	0	CS02	CS01	CS00	0	0	CE0	CAE0	FFFF F544H	00H
TMCD1	0	CS12	CS11	CS10	0	0	CE1	CAE1	FFFF F554H	00H

Bit Position	Bit Name	Function
6 to 4	CSn2 to CSn0	Selects the TMDn count clock (n = 0, 1)

CSn2	CSn1	CSn0	Count Clock
0	0	0	f_{PCLK} /4
0	0	1	f_{PCLK} /8
0	1	0	f_{PCLK} /16
0	1	1	f_{PCLK} /32
1	0	0	f_{PCLK} /64
1	0	1	f_{PCLK} /128
1	1	0	f_{PCLK} /256
1	1	1	f_{PCLK} /512

Caution: Do not change the CSn2 to CSn0 bits during timer operation. If they are to be changed, they must be changed after setting the CEn bit to "0". If the CSn2 to CSn0 bits are overwritten during timer operation, the operation is not guaranteed.

Bit Position	Bit Name	Function
1	CEn	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: CEn bit is not cleared even if a match is detected by the compare operation. To stop the count operation, clear the CEn bit.**
0	CAEn	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 CAEn = 0 is set, the TMDn unit is reset asynchronously.** **2. When CAEn = 0, the TMDn unit is in a reset state. To operate TMDn, first set CAE = 1.** **3. When the CAEn bit is changed from 1 to 0, all the registers of the TMDn unit are initialized. When again setting CAEn = 1, be sure to then again set all the registers of the TMDn unit.**

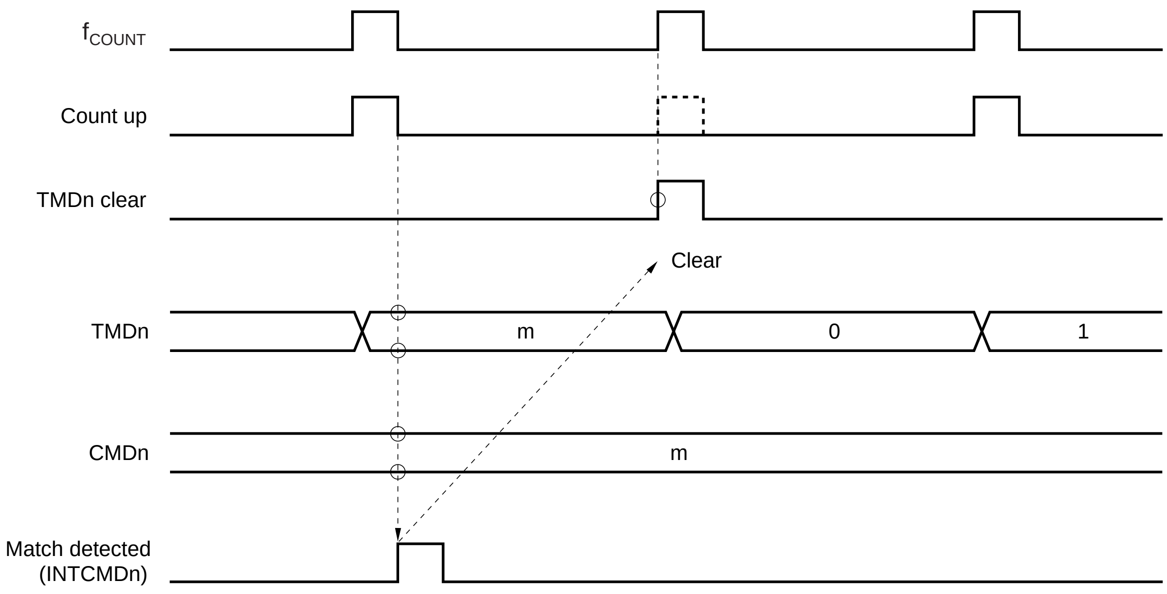
Caution: The CAEn bit and CEn bit cannot be set at the same time. Be sure to set the CAEn bit prior to setting the CEn bit.

7.1.5 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 (INTCMDn) 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 INTCMDn is generated. Although the TMDn value is cleared to "0" at the next count timing, INTCMDn is not generated according to this match.

Figure 7-6: Timing of Compare Operation (1/2)

(a) When CMDn is set to m (non-zero)



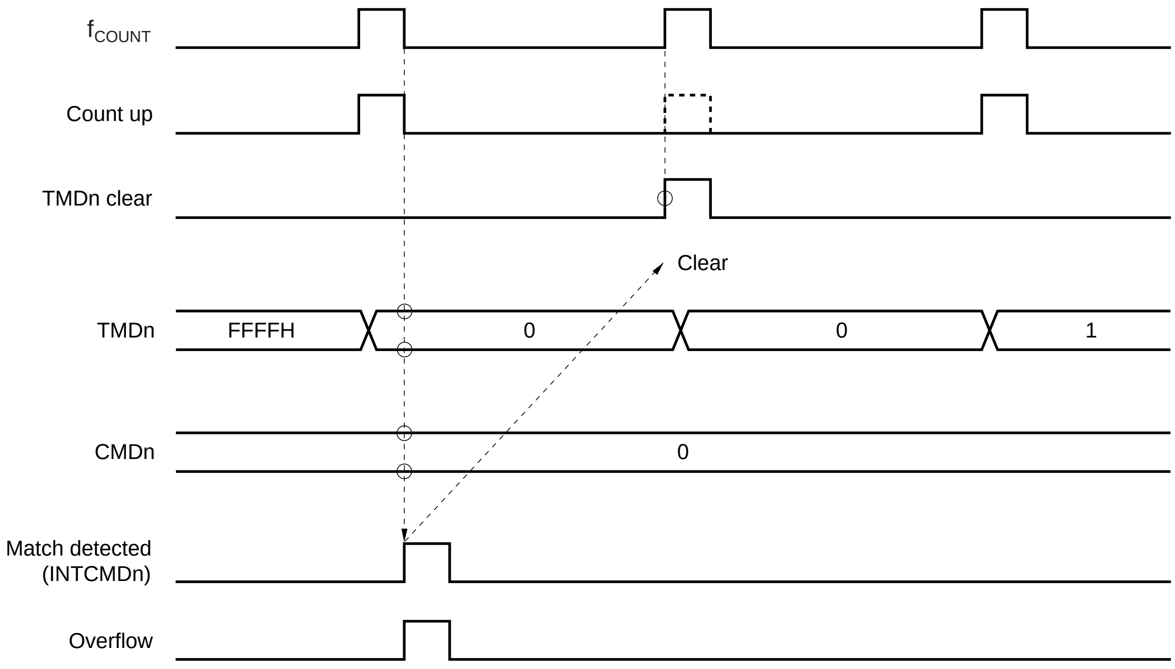
Remarks: **1.** Interval time = (m + 1) × Count clock cycle

2. m = 1 to 65535 (FFFFH)

3. n = 0, 1

Figure 7-6: Timing of Compare Operation (2/2)

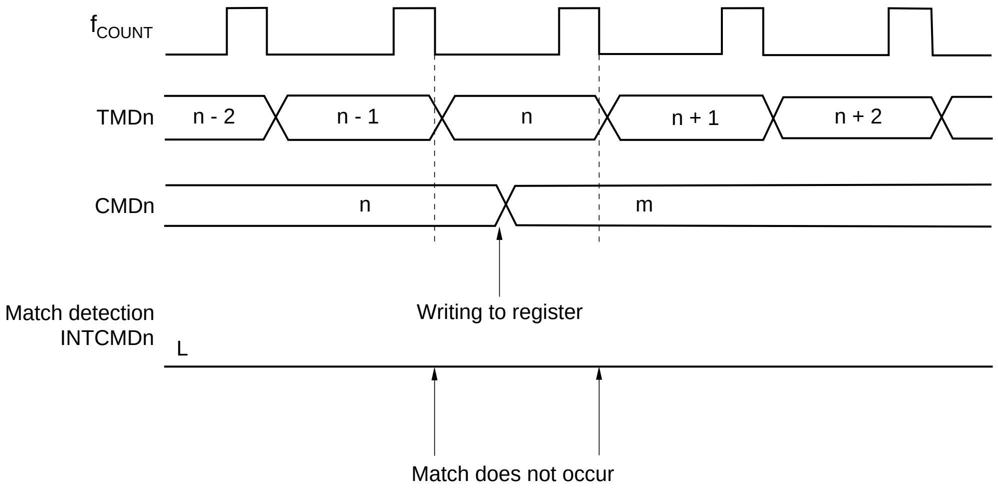
(b) When CMDn is set to 0



Remark: Interval time = (FFFFH + 2) × Count clock cycle

The value of the compare register CMDn is compared with the value of the TMDn register and a match between the two is detected immediately before the timer counts up. So in the following case, a match does not occur, and no interrupt is generated:

Figure 7-7: Writing to CMDn Register



7.1.6 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 (INTCMDn) are output at equal intervals (refer to Figure 7-6, “Timing of Compare Operation (1/2),” on page 177). The setup procedure is shown below (n = 0, 1).

<1> Set the CAEn bit to "1".
<2> Set each register:
Select the count clock using the CSn2 to CSn0 bits of the TMCDn register.
Set the compare value in the CMDn register.
<3> Start counting by setting the CEn bit to "1".
<4> If the TMDn register and CMDn register’s values match, an INTCMDn interrupt is generated.
<5> INTCMDn interrupts are generated thereafter at equal intervals.

7.1.7 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) 2 f_{PCLK}/2 clocks are required after a value is set in the CEn 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 CEn bit to “0” and then set the CEn bit to “1” after an interval of 2 f_{PCLK}/2 clocks has elapsed.

(4) Up to 2 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 2 f_{PCLK}/2 clocks.

(5) Writing to the CMDn register is allowed only once during a timer cycle: only the first value written to the CMDn register during a timer cycle (from 0000H until an INTCMDn interrupt is generated due to a match of the TMDn register and CMDn register) is effective.

(6) The count clock must not be changed during a timer operation. If the clock selection by CSn2 to CSn0 bits is going to be changed, it should be overwritten after the CEn bit is cleared to "0". If the count clock is changed during a timer operation, operation cannot be guaranteed.

(7) An INTCMDn 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

7.2 Timer G

2 × 16-bit multi purpose timer of Timer G are implemented:

- Timer G0
- Timer G1

7.2.1 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 HELIOS)
- 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)

7.2.2 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 (GCCn0, GCCn5) are assigned to dedicated counters (TMGn0, TMGn1)
  - 4 registers are freely assignable to one of the 2 counters

- Count clock division is selectable by a prescaler (frequency of peripheral clock: f_{PCLK} = 32 MHz Max)
  - In 8 steps from f_{PCLK} to f_{PCLK}/128
  - count clock may be different for each channel counter (TMGn0, TMGn1)

- Interrupt request sources
  - Edge detection circuit with noise elimination.
  - Compare-match interrupt requests: 6 types
    Perform comparison of the 6 capture/compare registers with one of the 2 counters (TMGn0, TMGn1) and generate the INTCCGny (y = 0 to 5) interrupt request upon compare match.
  - Timer counter overflow interrupt requests: 2 types
    In free running mode the INTTMGn0 (INTTMGn1) interrupt is generated when the count value of TMGn0 (TMGn1) goes from FFFFH to 0000H.

- PWM output function: 4 channels for each timer TMGn
  - Output pins TOGn1- through TOGn4 can be inverted at each occurrence of:
    - overflow of the corresponding timebase (TMGn0, TMGn1)
    - compare match between GCCn1 to GCCn4 register and the corresponding timebase (TMGn0, TMGn1).

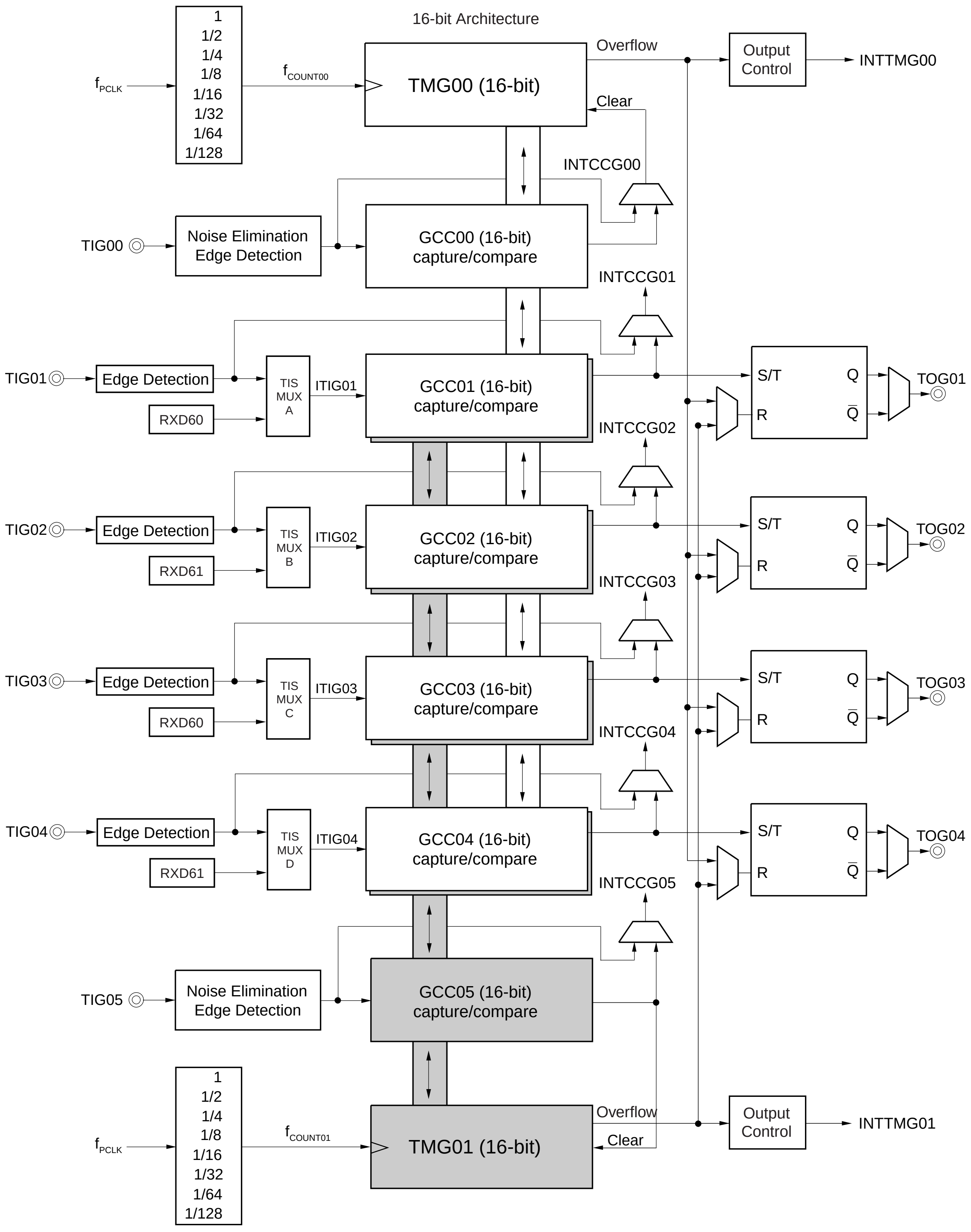
- Output delay operation
    - A clock-synchronised output delay can be added to the output signal of pins TOGn1 through TOGn4.

- Edge detection and noise elimination filter
  - TIG00, TIG05, TIG10 and TIG15 are provided with an analog noise eliminator.

Note: The TIGn1 to TIGn4 and TOGn1 to TOGn4 are each alternate function pins.

Figure 7-8, “Block Diagram of Timer G0,” on page 182 shows the block diagram of Timer Gn.

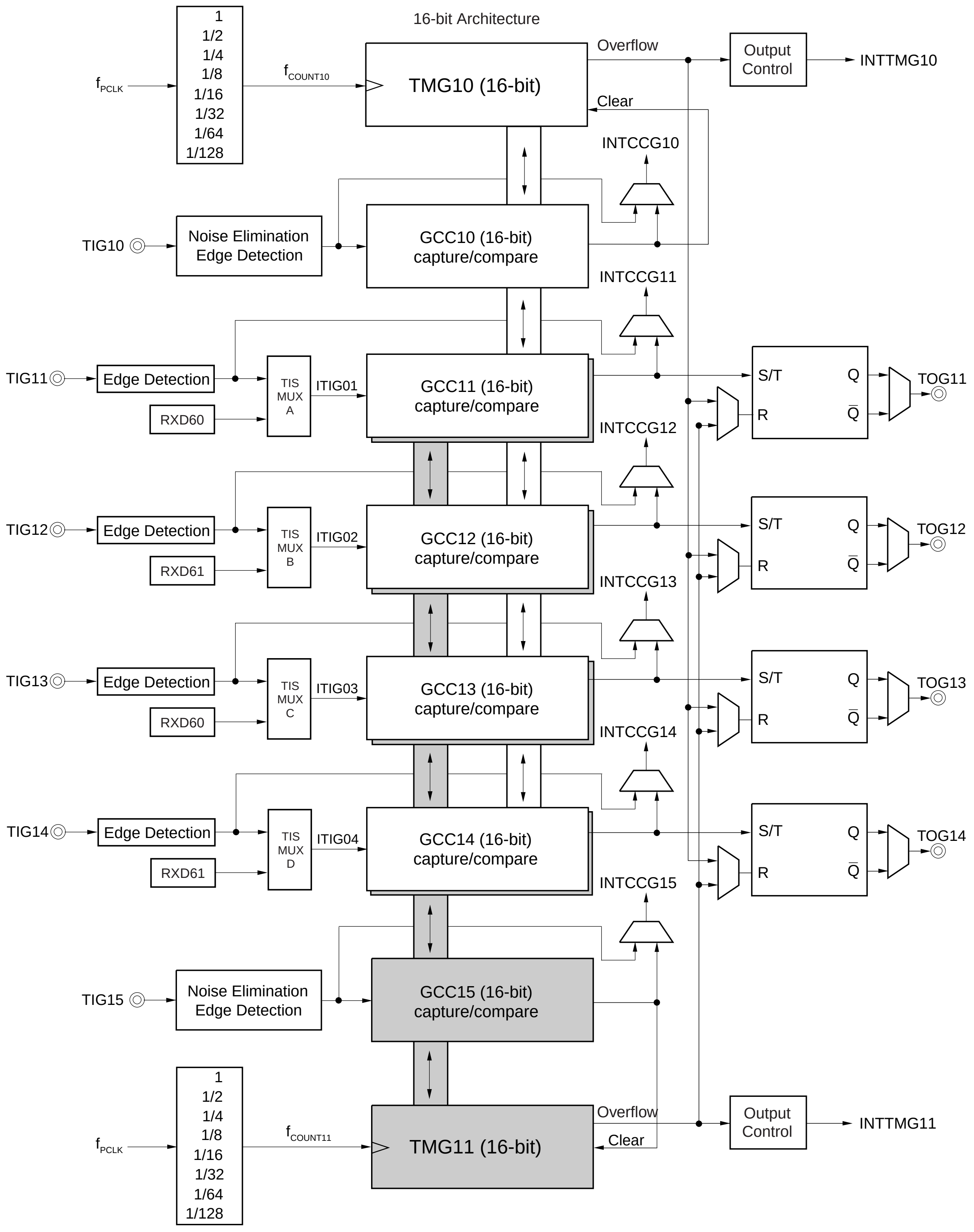
Figure 7-8: Block Diagram of Timer G0



Remark: f_{PCLK}: Internal peripheral clock (32 MHz max)

Note: TMG00/TMG01 are cleared by GCC00/GCC05 register compare match.

Figure 7-9: Block Diagram of Timer G1



Remark: f_{PCLK}: Internal peripheral clock (32 MHz Max.)

Note: TMG10/TMG11 are cleared by GCC10/GCC15 register compare match.

7.2.3 Basic configuration

The basic configuration is shown below.

Table 7-2: 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 7-10: 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 F588H	0000H
TMG10																	FFFF F608H	0000H

Figure 7-11: 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 F58AH	0000H
TMG11																	FFFF F60AH	0000H

(2) Timer Gn capture/compare registers of the 2 counters (GCCn0, GCCn5)

The GCCn0 and GCCn5 registers are 16-bit capture/compare registers of Timer Gn. These registers are respectively 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 TMGn0 (TMGn1). So this “match and clear mode” is used to reduce the full scale of the counter TMGn0 (TMGn1).

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

Caution: In compare mode, writing to these registers while the timer operates is prohibited. Write to this registers <u>before</u> POWERG and ENFGx bit (x = 0, 1) are set to “1” at the same time or <u>before</u> TMGnxE bit are set to “1”.

Figure 7-12: 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 F58CH	0000H
GCC10																	FFFF F60CH	0000H

Remark: This register is assigned fix to timebase TMGn0.

Figure 7-13: 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 F596H	0000H
GCC15																	FFFF F616H	0000H

Remark: This register is assigned fix to timebase TMGn1.

(3) Timer G capture/compare registers with external PWW-output function (GCCn1 to GCCn4)

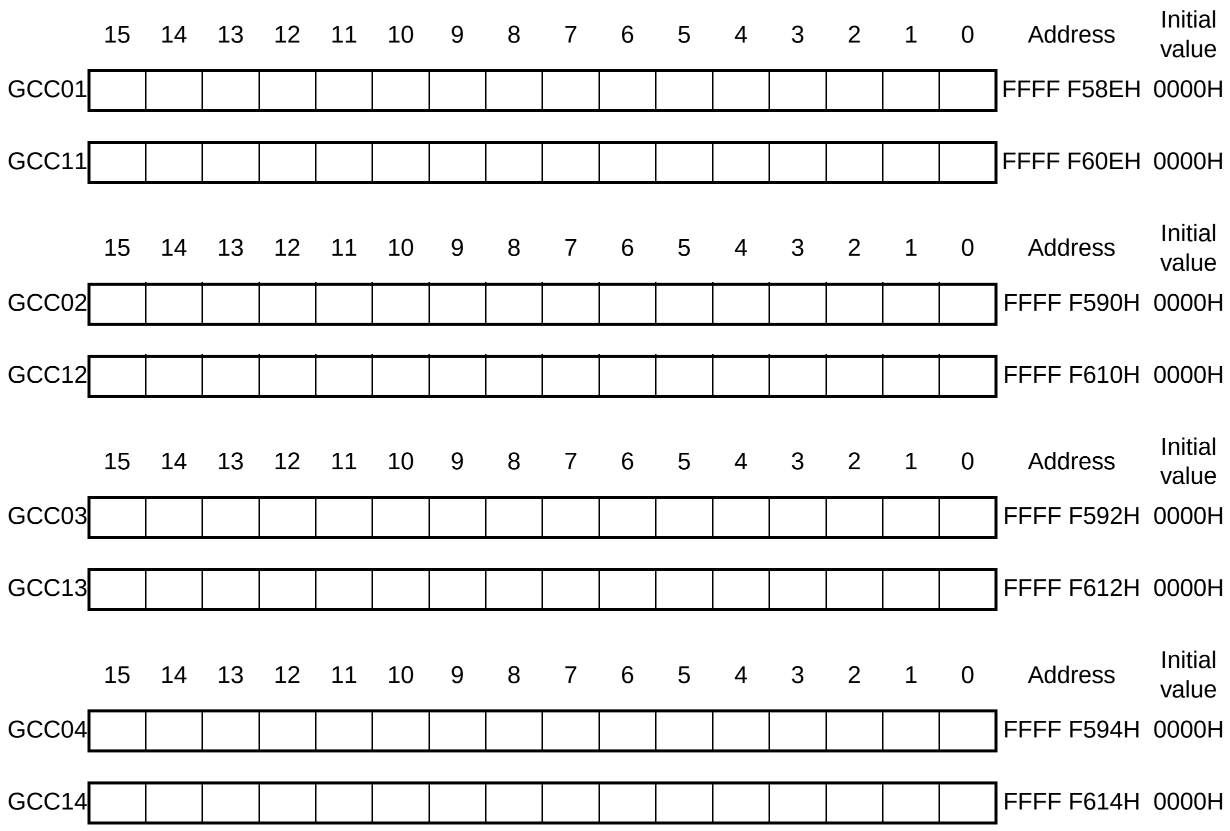
The GCCn1 to GCCn4 registers are 16-bit capture/compare registers of Timer Gn. They can be assigned to one of the 2 counters either TMGn0 or TMGn1.

In the **capture register mode**, these registers capture the value of TMGn0 when the TBGnm bit (m = 1 to 4, n = 0 to 1) of the TMGCMnH register = 0. When the TBGnm bit = 1, these registers capture the value of TMGn1.

In **compare mode**, these registers represent the actual compare value and the TOGnm-Output (m = 1 to 4, n = 0 to 1) can generate a PWW if they are activated.

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

Figure 7-14: Timer Gn Free Assignable Capture/Compare Registers (GCCnm) (m = 1 to 4)



Remarks: 1. In capture mode only reading is possible

2. In compare mode read/write is possible

7.2.4 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.
When using 16-bit access, the register name is TMGM0 or TMGM1.
When using 8-bit access, the register name is TMGMnL for the low-order 8 bits, and TMGMnH for the high-order 8 bits.
When using 1-bit access, use the bit name.

Figure 7-15: Timer Gn Mode Register (TMGMn) (1/2)

(a) Timer Gn Mode Register High (TMGMnH)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGM0H	POWERG0	OLDE0	CSE012	CSE011	CSE010	CSE002	CSE001	CSE000	FFFFF581H	00H
TMGM1H	POWERG1	OLDE1	CSE112	CSE111	CSE110	CSE102	CSE101	CSE100	FFFFF601H	00H

Bit Position		Bit Name	Function
16-bit TMGMn	8-bit TMGMnH		
15	7	POW-ERGn	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 14 peripheral-clocks (f_{PCLK}) are need to start the timer function
14	6	OLDEn	Set Output Delay Operation. 0: Don't perform output delay operation 1: Set output delay to n count-clocks
13 to 8	5 to 0	CSEnx2, CSEnx1, CSEnx0	Selects internal count clock of TMGn (see table below) **Caution: When the POWERG-bit is set, the rewriting of these bits is prohibited!** **Remarks: 1.** x = 0, 1 **2.** f_{PCLK}: peripheral clock

CSEnx2	CSEnx1	CSEnx0	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 7-15: Timer Gn Mode Register (TMGMn) (2/2)

(b) Timer Gn Mode Register Low (TMGMnL)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGM0L	CCSG05	CCSG00	0	0	CLRG01	TMG01E	CLRG00	TMG00E	FFFF F580H	00H
TMGM1L	CCSG15	CCSG10	0	0	CLRG11	TMG11E	CLRG00	TMG10E	FFFF F600H	00H

Bit Position	Bit Name	Function
7, 6	CCSGn5, CCSGn0	CCSGn0: specifies the mode for TMG00 and GCC00 CCSGn5: specifies the mode for TMG01and GCC05 0: Free running mode for TMGn0 (TMGn1), and capture mode for GCCn0 (GCCn5): A detected edge at pin TIGn0 (TIGn5) stores the value of TMGn0 (TMGn1) in GCCn0 (GCCn5) and the interrupt INTCCGn0 (INTCCGn5) is generated 1: Match and Clear mode for TMGn0 (TMGn1), and compare mode for GCCn0 (GCCn5): When the value of GCCn0 (GCCn5) matches the value of TMGn0 (TMGn1), the counter TMGn0 (TMGn1) is cleared and the interrupt INTCCGn0 (INTCCGn5) is generated **Caution: When the POWERG bit is set, rewriting of these bits is prohibited!**
3, 1	CLRGnx	Specifies software clear for TMGnx (x = 0, 1): 0: Continue TMGnx operation 1: Clears (0) the count value of TMGnx, and sets the output TOGnm of each GCCnm register assigned to TMGnx at its inactive level. **Remark:** TMGnx starts 2 system-clocks after this bit is set. This bit is a trigger, it is not stored and always read as 0.
2, 0	TMGnxE	Specifies TMGnx (x = 0, 1) count operation enable/disable 0: Stop count operation the counter holds the current value the corresponding TOGnx is deactivated 1: Enable count operation **Remarks: 1.** the counter needs at least 2count clocks (f_{PCLK}) to stop **2.** the counter needs at least 8 count clocks (f_{PCLK}) to start

(2) 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 TIGny-input-pins (n = 0 to 1, y = 0 to 5).

This register can be read/written in 16-bit, 8-bit or 1-bit units.
When using 16-bit access, the register name is TMGCM0 or TMGCM1.
When using 8-bit access, the register name is TMGCMnL for the low-order 8 bits, and TMGMCnH for the high-order 8 bits.
When using 1-bit access, use the bit name.

Figure 7-16: Timer Gn Channel Mode Register (TMGCMn)

(a) Timer Gn Channel Mode Register High (TMGCMnH)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGCM0H	TBG04	TBG03	TBG02	TBG01	IEG051	IEG050	IEG041	IEG040	FFFF F583H	00H
TMGCM1H	TBG14	TBG13	TBG12	TBG11	IEG151	IEG150	IEG141	IEG140	FFFF F603H	00H

(b) Timer Gn Channel Mode Register High (TMGCMnL)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGCM0L	IEG031	IEG030	IEG021	IEG020	IEG011	IEG010	IEG001	IEG000	FFFF F582H	00H
TMGCM1L	IEG131	IEG130	IEG121	IEG120	IEG111	IEG110	IEG101	IEG100	FFFF F602H	00H

Bit Position		Bit Name	Function
16-bit	8-bit		
TMGCMn 15 to 12	TMGCMnH 7 to 4	TBGnm	Assigns Capture/Compare-registers GCCn1 to GCCn4 and the corresponding input/output pins to one of the 2 counters TMGn0 or TMGn1: 0: assigns the TMGn0 counter to GCCny register and TIGny/TOGnm pin 1: assigns the TMGn1 counter to GCCny register and TIGny/TOGnm pin
11 to 8 and 3 to 0	TMGCMnH 3 to 0 and TMGCMnL 7 to 0	IEGny1, IEGny0	Specifies the valid edge of external capture signal input pin (TIGny) for the capture register performing capture-match with the assigned counter TMGn0 or TMGn1:

IEGny1	IEGny0	Valid Edge
0	0	Falling edge
0	1	Rising edge
1	0	No edge detection performed
1	1	Both rising and falling edges

Remarks: 1. y = 0 to 5

2. m = 1 to 4

3. n = 0 to 1

Caution: When the POWERGn bit is set, rewriting of TMGCMnH and TMGCMnL is prohibited.

(3) Timer Gn output control register (OCTLGn)

This register controls the timer output from the TOGnm pin (n = 0 to 1, m = 1 to 4) and the capture or compare modes for the GCCnm register.

This register can be read/written in 16-bit, 8-bit or 1-bit units.
When using 16-bit access, the register name is OCTLG0 or OCTLG01.
When using 8-bit access, the register name is OCTLGnL for the low-order 8 bits, and OCTLG0nH for the high-order 8 bits.
When using 1-bit access, use the bit name.

Cautions: 1. When the POWERGn bit is set, the rewriting of CCSGnm is prohibited

2. When the POWERGn bit and TMGnxE bit are set at the same time, the rewriting of the ALVGnm bits is prohibited.

Figure 7-17: Timer Gn Output Control Register OCTLGn (1/2)

(a) Timer Gn Output Control Register High OCTLGnH

	7	6	5	4	3	2	1	0	Address	Initial value
OCTLG0H	SWFG04	ALVG04	CCSG04	0	SWFG03	ALVG03	CCSG03	0	FFFF F585H	44H
OCTLG1H	SWFG14	ALVG14	CCSG14	0	SWFG13	ALVG13	CCSG13	0	FFFF F605H	44H

(b) Timer Gn Output Control Register Low OCTLGnL

	7	6	5	4	3	2	1	0	Address	Initial value
OCTLG0L	SWFG02	ALVG02	CCSG02	0	SWFG01	ALVG01	CCSG01	0	FFFF F584H	44H
OCTLG1L	SWFG12	ALVG12	CCSG12	0	SWFG11	ALVG11	CCSG11	0	FFFF F604H	44H

Figure 7-17: Timer Gn Output Control Register OCTLGn (2/2)

Bit Position		Bit Name	Function
16-bit	8-bit		
15, 11, 7, 3	7, 3	SWFGnm	Fixes the TOGnm pin output level according to the setting of ALVGnm bit. 0: disable TOGnm to inactive level 1: enable TOGnm **Caution: Don’t write this bit, before POWERGn bit of TMGMn is 0.**
14, 10, 6, 2	6, 2	ALVGnm	Specifies the active level of the TOGnm pin output. 0: Active level is 0 1: Active level is 1 **Caution: Don’t write this bit, before ENFGn0 or ENFGn1 of TMGSTn is 0, so first clear TMGn0E or TMGn1E bit of the TMGMn register and check ENFGn0 or ENFGn1 bit before writing.**
13, 9, 5, 1	5, 1	CCSGnm	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 assigned timebase the INTCCGnm interrupt occurs, if SWFGnm is set the PWM output mode is set

Remarks: 1. n = 0 to 1

2. m = 1 to 4

3. y = 0 to 5

(4) Time base status register (TMGSTn)

The TMGSTn register indicates the status of TMGn0 and TMGn1. For the CCFGny bit see section 7.2.7 ”Operation in Free-run mode” on page 197.

This register can be read in 8-bit or 1-bit units.

Figure 7-18: Timer Gn Status Register (TMGSTn)

	7	6	5	4	3	2	1	0	Address	Initial value
TMGST0	ENFG01	ENFG00	CCFG05	CCFG04	CCFG03	CCFG02	CCFG01	CCFG00	FFFF F586H	00H
TMGST1	ENFG11	ENFG10	CCFG15	CCFG14	CCFG13	CCFG12	CCFG11	CCFG10	FFFF F606H	00H

Bit Position	Bit Name	Function
5 to 0	CCFGny	Indicates TMGn0 or TMGn1 overflow status. 0: No overflow 1: Overflow **Caution: The CCFGny 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**
7 to 6	ENFGn1, ENFGn0	Indicates TMGn1 (TMGn0) operation. 0: indicates operation stopped 1: indicates operation

Remarks: 1. y = 0 to 5

2. n = 0, 1

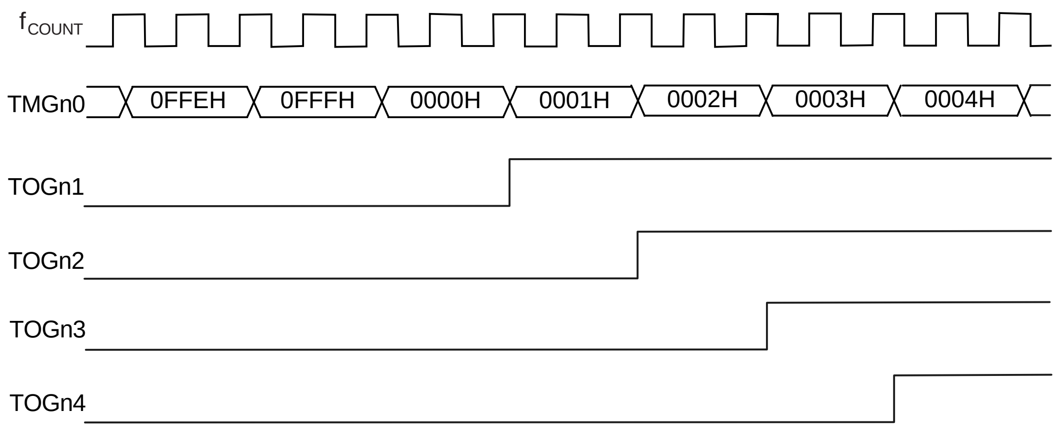
7.2.5 Output delay operation

When the OLDEn 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 $f_{PCLK}/2$ and 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 7-19: Timing of Output Delay Operation



In this case the count clock is set to $f_{PCLK}/2$.

7.2.6 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 timebase. TMGn0 (TMGn1) can be set by CCSGn0 bit (CCSGn5 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 TBGnm bit of the TMGCMnH register.
The tables below indicate the interrupt output and timer output states dependent on the register setting values.

Table 7-3: Interrupt Output and Timer Output States Dependent on the Register Setting Values

Register setting value				State of each output pin			
CCSGn0	TBGnm	SWFGnm	CCSGnm	INTTMGn0	INTCCGn0	INTCCGnm	TOGnm
0 Free-run mode	0	0	0	Overflow interrupt	TIn0 edge detection	TIGnm edge detection	Tied to inactive level
			1			CMPGnm match	
		1	0			TIGnm edge detection	
			1			CMPGnm match	PWM (free run)
1 Match and clear mode		0	0	Overflow interrupt[Note 1]	CMPGn0 match[Note 2]	TIGnm edge detection	Tied to inactive level
			1			CMPGnm match	
		1	0			TIGnm edge detection	
			1			CMPGnm match	PWM (match and clear)

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.

Remarks: 1. n = 0 to 1

2. m = 1 to 4

Table 7-4: Interrupt Output and Timer Output States Dependent on the Register Setting Values

Register setting value				State of each output pin			
CCSGn5	TBGnm	SWFGnm	CCSGnm	INTTMGn1	INTCCGn5	INTCCGnm	TOGnm
0 Free-run mode	1	0	0	Overflow interrupt	TIn5 edge detection	TIGnm edge detection	Tied to inactive level
			1			CMPGnm match	
		1	0			TIGnm edge detection	
			1			CMPGnm match	PWM (free run)
1 Match and clear mode		0	0	Overflow interrupt[Note 1]	CMPGn5 match[Note 2]	TIGnm edge detection	Tied to inactive level
			1			CMPGnm match	
		1	0			TIGnm edge detection	
			1			CMPGnm match	PWM (match and clear)

Notes: 1. An interrupt is generated only when the value of the GCCn5 register is FFFFH.

2. An interrupt is generated only when the value of the GCCn5 register is not FFFFH.

Remarks: 1. n = 0 to 1

2. m = 1 to 4

7.2.7 Operation in Free-run mode

This operation mode is the standard mode for Timer Gn operations. In this mode the 2 counters 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 7.2.8 on page 206 the fixed assigned register GCCn0 (GCCn5) is used to reduce the full-scale 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	
SWFGnm	0	disable TOGnm
TBGnm	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 TBGnm 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 CSEn12 to CSEn10 bits (TMGn1) or CSEn12 to CSEn02 bits (TMGn0).
(3) Select a valid TIGny edge with the IEGny1 and IEGny0 bits. A rising edge, falling edge, or both edges can be selected.
(4) Start timer operation by setting POWERGn bit and TMGn0E bit for TMGn0 or TMGn1E bit for TMGn1.

Capture Operation

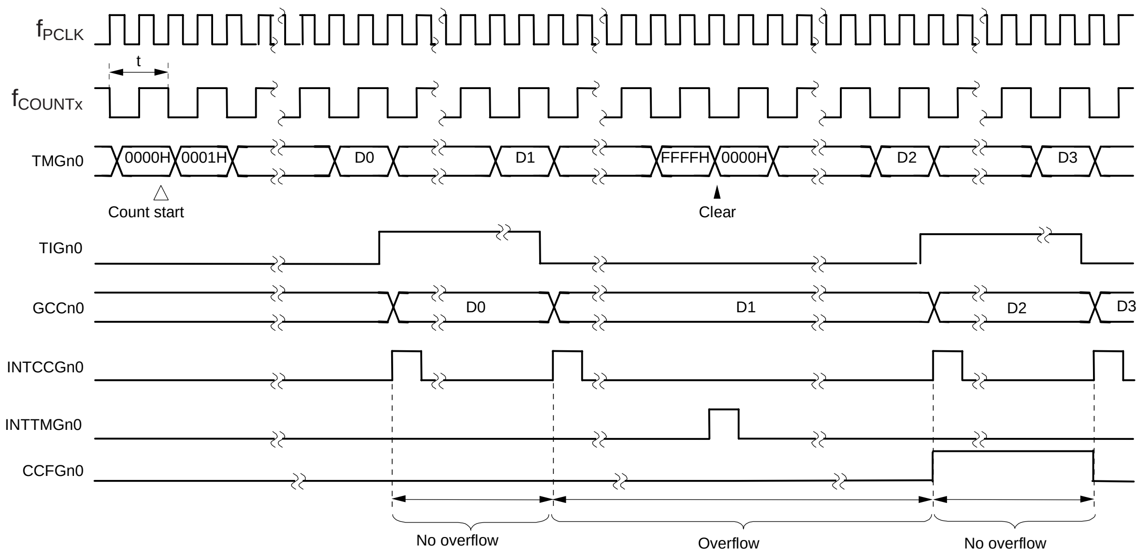
(1) When a specified edge is detected, the value of the counter is stored in GCCny and an edge detection interrupt (INTCCGny) is output.
(2) When the counter overflows, an overflow interrupt (INTTMGn0 or INTTMGn1) is generated.
(3) If an overflow has occurred between capture operations, the CCFGny flag is set when GCCny is read. Correct capture data by checking the value of CCFGny.

Using CCFGny:
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 CCFGny bit of the TMGSTn register is set.
<3> If the CCFGny bit is set, the counter was cleared from the previous captured value.
CCFGny is set when GCCny is read. So, after GCCny is read, the value of CCFGny should be read. Using the procedure above, the value of CCFGny corresponding to GCCny can be read normally.

Caution: If two or more overflows occur between captures, a software-based measure needs to be applied to count overflow interrupts (INTTMGn0, INTTMGn1).

Figure 7-20: 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.

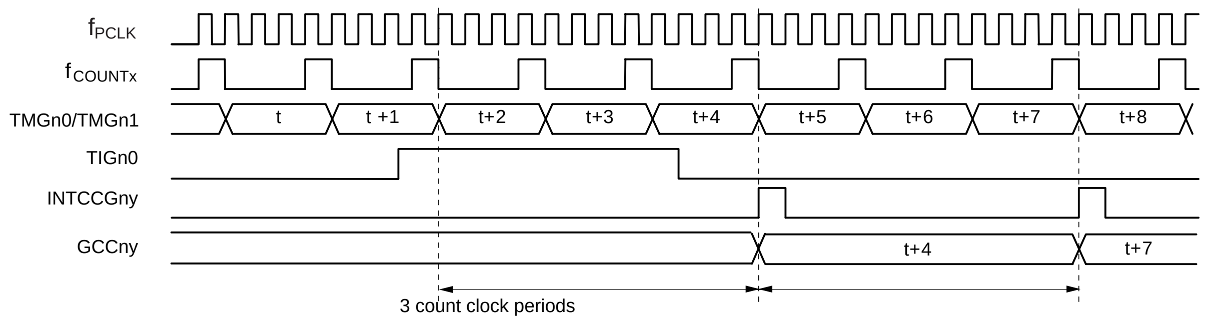
(b) Timing of capture trigger edge detection

The TIGn0 and TIGn5 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 (n= 0, 1; x= 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEnx2	0	Count clock = $f_{PCLK}/4$
CSEnx1	0	
CSEnx0	1	
IEGny1	1	detection of both edges
IEGny0	1	

Figure 7-21: Timing of Capture Trigger Edge Detection (free run)



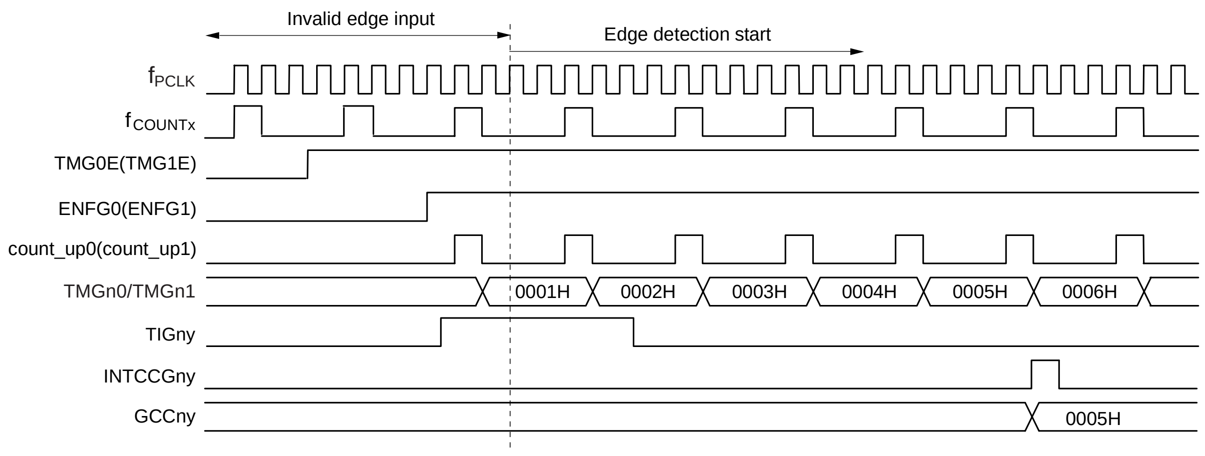
(c) Timing of starting capture trigger edge detection

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 (n =0, 1; x= 0, 1 and y = 0 to 5):

Bit	Value	Remark
CSEnx2	0	Count clock = $f_{PCLK}/4$
CSEnx1	0	
CSEnx0	1	
IEGny1	1	detection of both edges
IEGny0	1	

Figure 7-22: Timing of Starting Capture Trigger Edge Detection



(2) Compare operation (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	0	disable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Interval timer (free run)

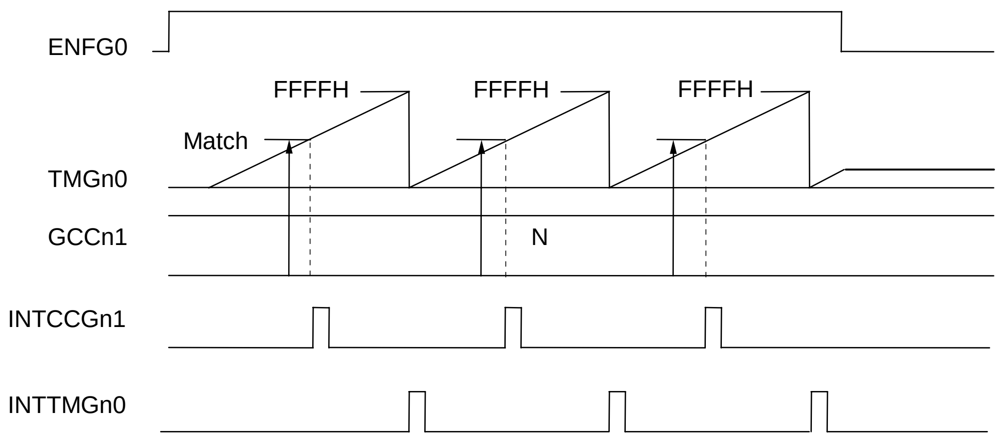
Setting method interval timer:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter (TMGn0 or TMGn1) must be selected with the TBGnm bit.
(2) Select a count clock cycle with the CSEn12 to CSEn10 bits (TMGn1 register) or CSEn02 to CSEn00 bits (TMGn0 register).
(3) Write data to GCCnm.
(4) Start timer operation by setting POWERGn and TMGn0E (or TMGn1E).

Compare Operation:

(1) When the value of the assigned counter matches the value of GCCnm, a match interrupt (INTC-CGnm) is output.
(2) When the counter overflows, an overflow interrupt (INTTMGn0/INTTMGn1) is generated.

Figure 7-23: 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 GCCny

INTCCGny 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 GCCny, INTCCGny is activated immediately after the counter starts.

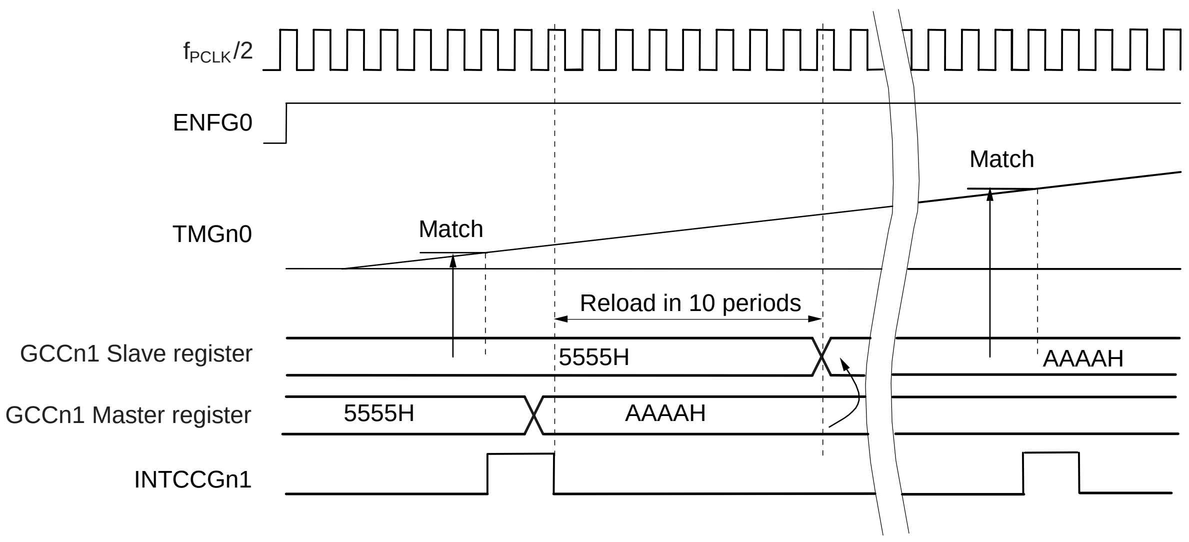
(c) When the value FFFFH is set in GCCnm

INTCCGny and INTTMGn0/INTTMGn1 are activated when the value of the counter changes from FFFFH to 0000H.

(d) When GCCny is rewritten during operation

When GCCn1 is rewritten from 5555H to AAAAH. TMGn0 is selected as the counter.
The following operation is performed:

Figure 7-24: 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 14 peripheral clocks periods (f_{PCLK}).

(3) PWM output (free run)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	0	free run mode
CCSGn5	0	
SWFGnm	1	enable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

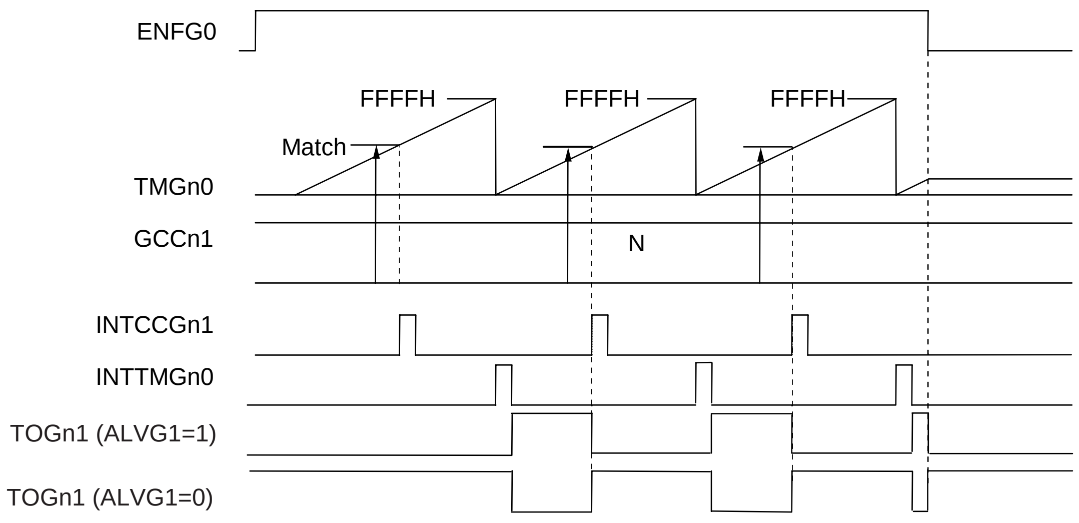
PWM setting method:

(1) An usable compare register is one of GCCn1 to GCCn4, and the corresponding counter must be assigned with the TBGm bit.
(2) Select a count clock cycle with the CSEn12 to CSEn10 bits (TMGn1 register) or CSEn02 to CSEn00 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 OLDEn bit of TMGMnH 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 POWERGn bit and TMGn0E bit (or TMGn1E 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 7-25: Timing of PWM Operation (free run)

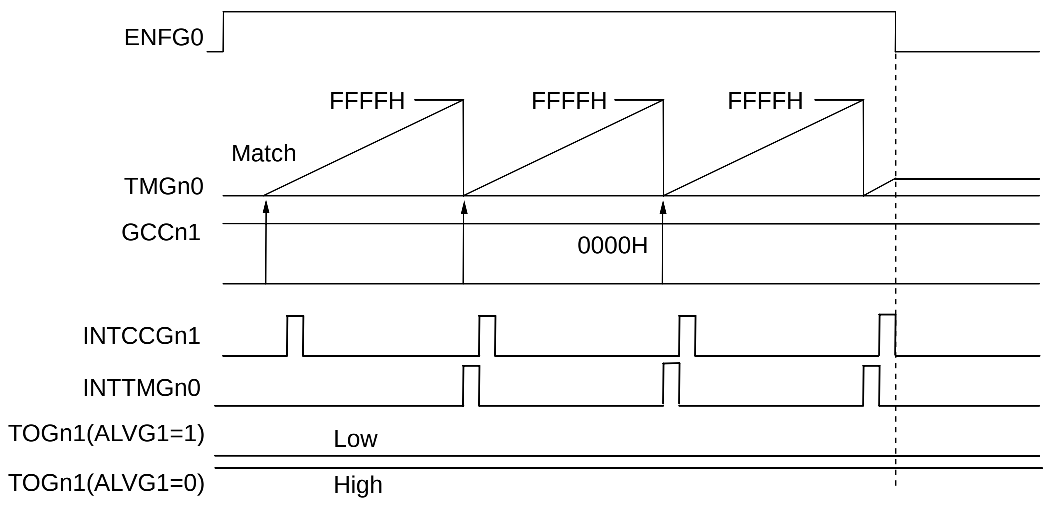


Data N is set in GCCn1, counter TMGn0 is selected.

(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 7-26: Timing when 0000H is set in GCCnm (free run)



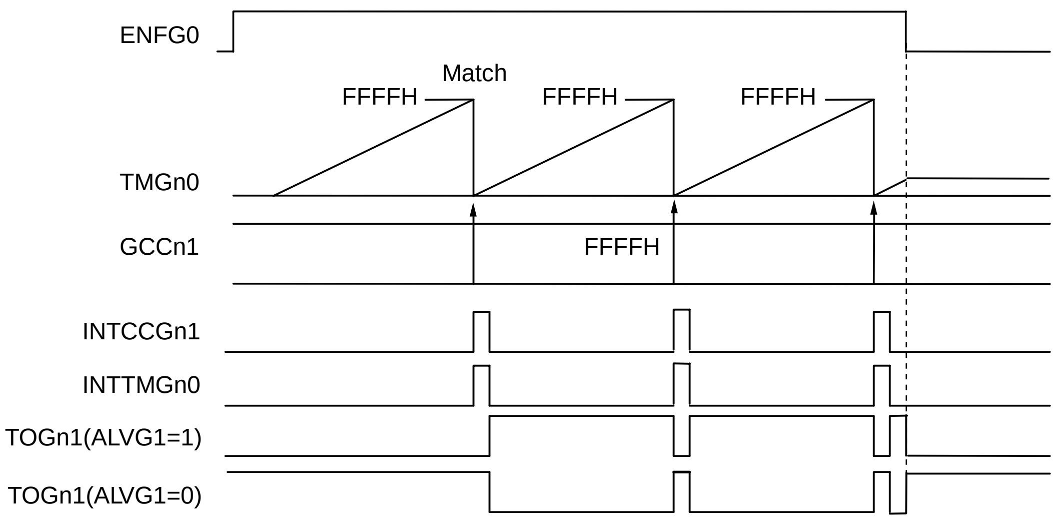
GCCn1 and TMGn0 are selected.

(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 7-27: Timing when FFFFH is set in GCCnm (free run)



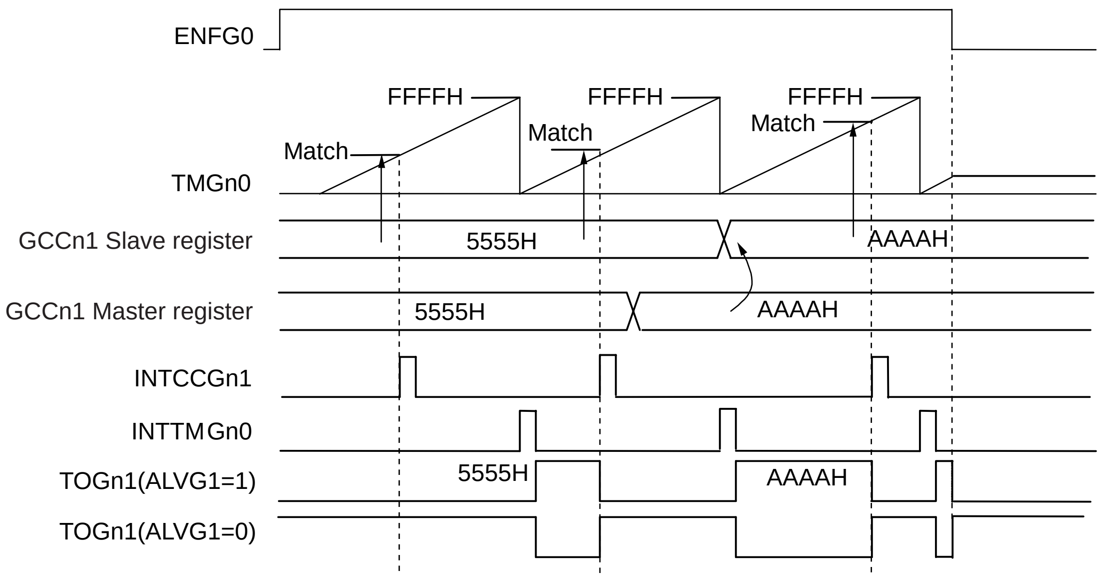
GCCn1 and TMGn0 are selected.

(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 7-28: 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.

7.2.8 Match and clear mode

The match and clear mode is mainly used to reduce the full-scale 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 and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	0	disable TOGnm
CCSGnm	0	Capture mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

(a) Example: Pulse width measurement or period measurement of the TIGnm input signal

Setting method:

(1) When using one of TOGn1 to TOGn4-pin, select the corresponding counter with the TBGnm bit. When CCSGn0 = 1, TIGn0 cannot be used. When CCSGn5 = 1, TIGn5 cannot be used.
(2) Select a count clock cycle with the CSEn12 to CSEn10 (TMGn1) bits or CSEn02 to CSEn00 (TMGn0) bits.
(3) Select a valid TIGnm edge with the IEGnm1 and IEGnm0 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 POWERGn bit and TMGn0E bit (or TMGn1E bit).

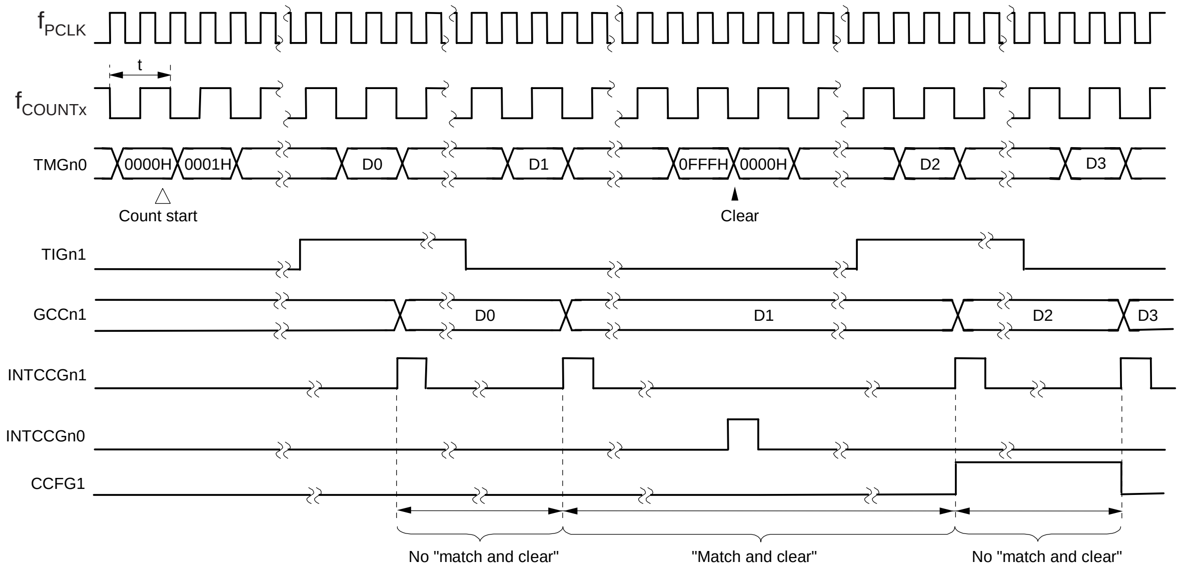
Operation:

(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 CCFGny flag is set when GCCny is read. Correct capture data by checking the value of CCFGny.

(b) Example: Capture where both edges of TIGnm 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 7-29: Timing when both Edges of TIGnm 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 7-21, “Timing of Capture Trigger Edge Detection (free run),” on page 200.)

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
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	0	disable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	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 TBGnm bit.

(2) Select a count clock cycle with the CSEn12 to CSEn10 bits (TMGn1) or CSEn02 to CSEn00 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 POWERGn bit and TMGnxE bit (x = 0, 1).

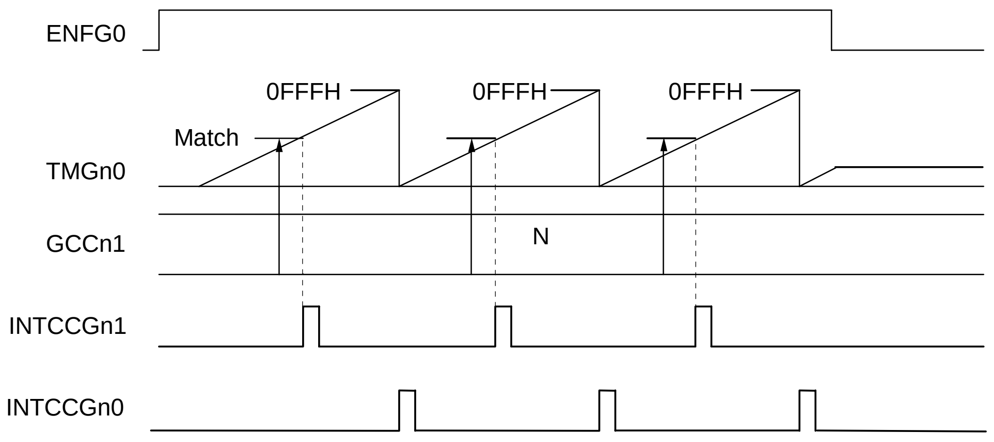
Operation:

(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 7-30: 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.

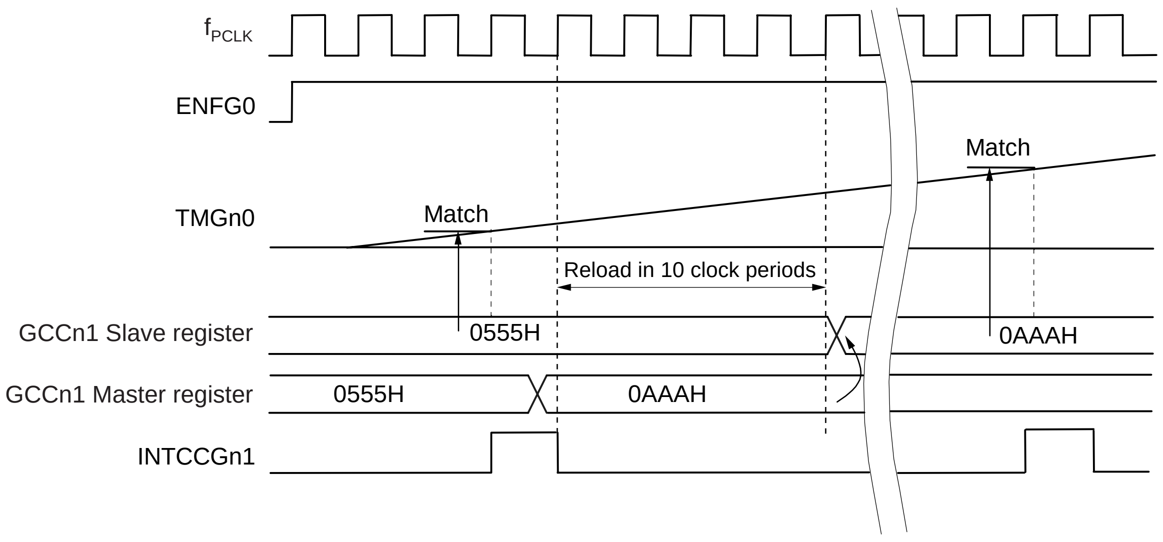
(e) When a value exceeding the value of GCCn0 or GCCn5 is set in GCCnm (m = 1 to 4) (match and clear)

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 7-31: Timing when GCCnm is Rewritten During Operation (Match and Clear)



Caution: To perform successive write access during operation, for rewriting the GCCnm register (m = 1 to 4), you have to wait for minimum 14 peripheral clocks periods (f_{PCLK}).

(3) PMW output (match and clear)

Basic settings (m = 1 to 4):

Bit	Value	Remark
CCSGn0	1	match and clear mode
CCSGn5	1	
SWFGnm	1	enable TOGnm
CCSGnm	1	Compare mode for GCCnm
TBGnm	X	assign counter for GCCnm 0: TMGn0 1: TMGn1

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 TBGnm bit (m = 1 to 4).
(2) Select a count clock cycle with the CSEn12 to CSEn10 (TMGn1) bits or CSEn02 to CSEn00 (TMGn0) bits.
(3) Specify the active level of a timer output (TOGnm) with the ALVGnm bit.
(4) When using multiple timer outputs, the user can prevent TOGnm from making transitions simultaneously by setting the OLDEn bit of TMGMnH 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. (0X0000H is forbidden)
(6) Write data to GCCnm.
(7) Start count operation by setting POWERGn bit and TMGn0E bit (or TMGn1E bit).

Operation of PWM (match and clear):

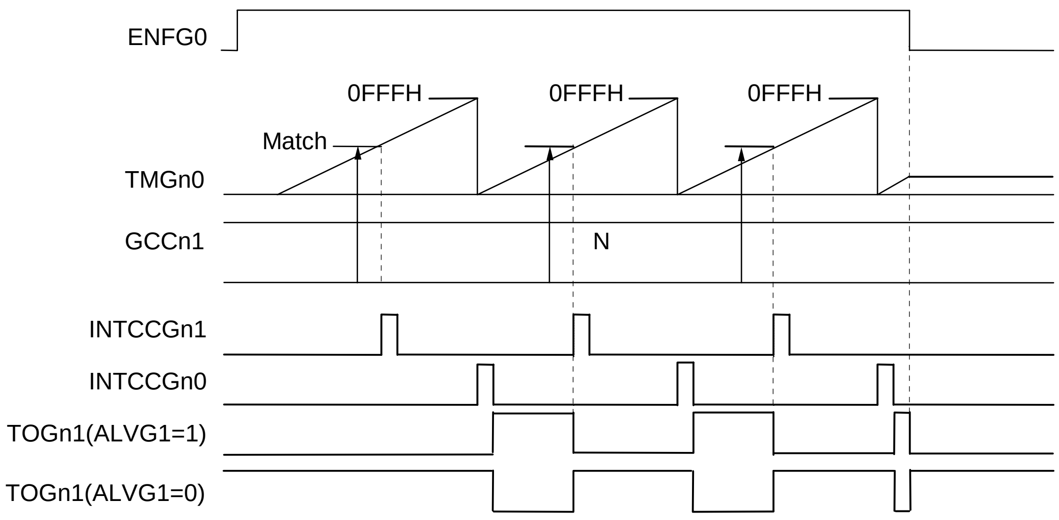
(1) When the value of the counter matches the value of GCCnm, a match interrupt (INTCCGnm) is output.

Caution: Do not set 0000H in GCCn0 or GCCn5 in match and clear mode.

(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 7-32: 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.

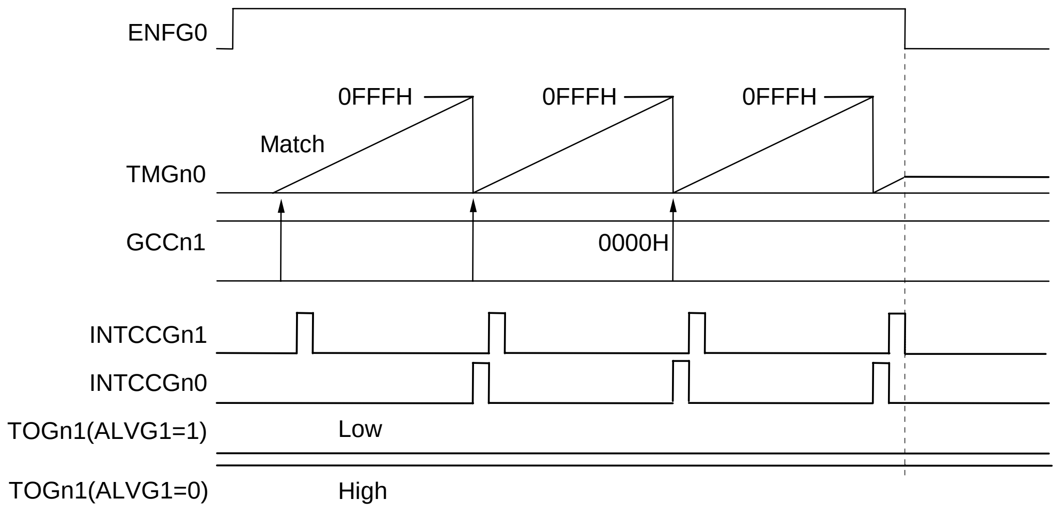
(a) When FFFFH is set in GCCn0 or GCCn5 (match and clear)

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 7-33: Timing when 0000H is Set in GCCnm (Match and Clear)

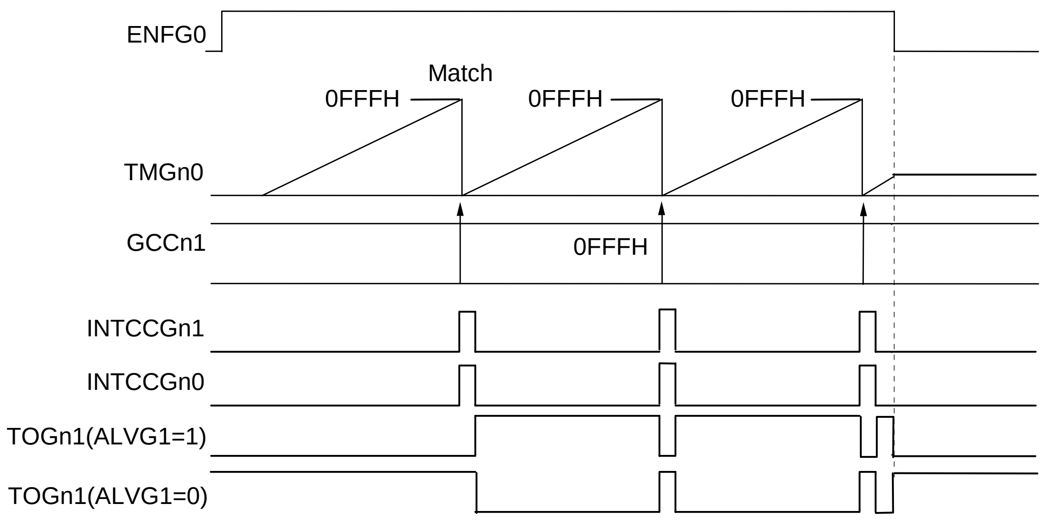


(c) When the same value as set in GCCn0 or GCCn5 is set in GCCnm (match and clear)

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.

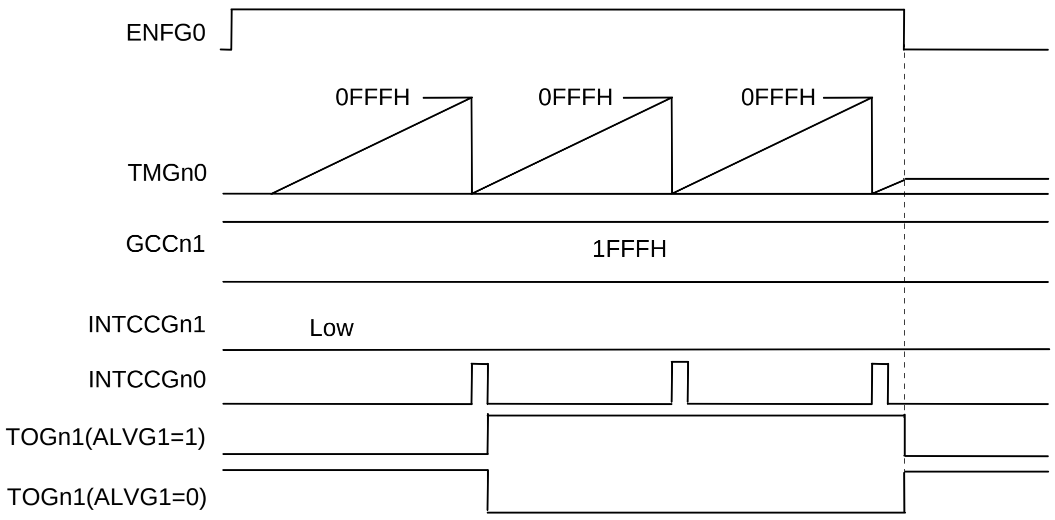
Figure 7-34: Timing when the same Value as set in GCCn0/GCCn5 is set in GCCnm (match and clear)



(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.

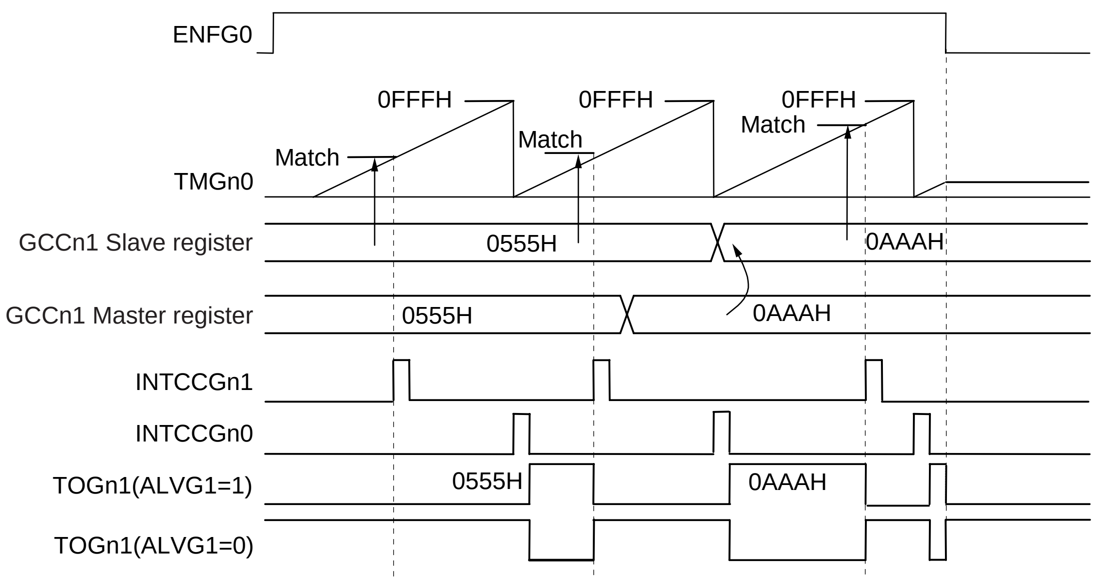
Figure 7-35: Timing when the Value of GCCnm Exceeding GCCn0 or GCCn5 (match and clear)



(e) When GCCnm is rewritten during operation (match and clear)

When GCCnm 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 7-36: 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.

7.2.9 Edge noise elimination

The edge detection circuit associated to the TIG00, TIG05, TIG10 and TIG15 has an analog noise elimination function. See Electrical Specifications for more details.

7.2.10 Precautions Timer Gn

(1) When POWERGn bit of TMGMnH register is set

The rewriting of the CSEn2 to CSEn0 bits (n = 0, 1) of TMGMnH register is prohibited.
These bits set the prescaler for the Timer Gn counter.

The rewriting of the CCSGny bits (y = 0 to 5) is prohibited.
This bits (OCTLGLn and OCTLGHn registers) set the capture mode or the compare mode to the GCCny register. For the GCCn0 register and the GCCn5 register these bits (TMGMnL register) set the “free run” or “match and clear” mode of the TMGn0 and TMGn1 counter.

The rewriting of the TMGCMLn 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 TIGnm input pins (falling, rising, both).

Even when POWERGn bit is set, TOGnm output can be switched by switching the ALVGm bit of OCTLGLn and OCTLGHn registers.
These bits configure the active level of the TOGnm-pins (m = 1 to 4).

(2) When POWERGn bit and TMGnxE bit are set (x = 0, 1)

The rewriting of ALVGnm 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 POWERGn bit is set to “0”, regardless of the SWFGnm bit (OCTLGLn and OCTLGHn registers), the TOGnm pins are tied to the inactive level.
The SWFGnm bit enables or disables the output of the TOGnm pins. This bit can be rewritten during timer operation.

The CLRGnx 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 CCFGny bits (y = 0 to 5) can be used for overflow detection.

Only the overflow of the TMGn0 or TMGn1 counter clears the CCFGny bit (TMGSTn register). The software-based clearing via CLRGn0 or CLRGn1 bit (TMGMnL register) doesn’t affect these bits.
The CCFGny 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 CSEnx2 to CSEnx0 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 TMGnxE (x = 0, 1) is set earlier or simultaneously with POWERGn bit, than the Timer Gn needs 14 peripheral clocks periods (f_{PCLK}) to start counting.
When TMGnxE (x = 0, 1) is set later than POWERGn bit, than the Timer Gn needs 8 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.

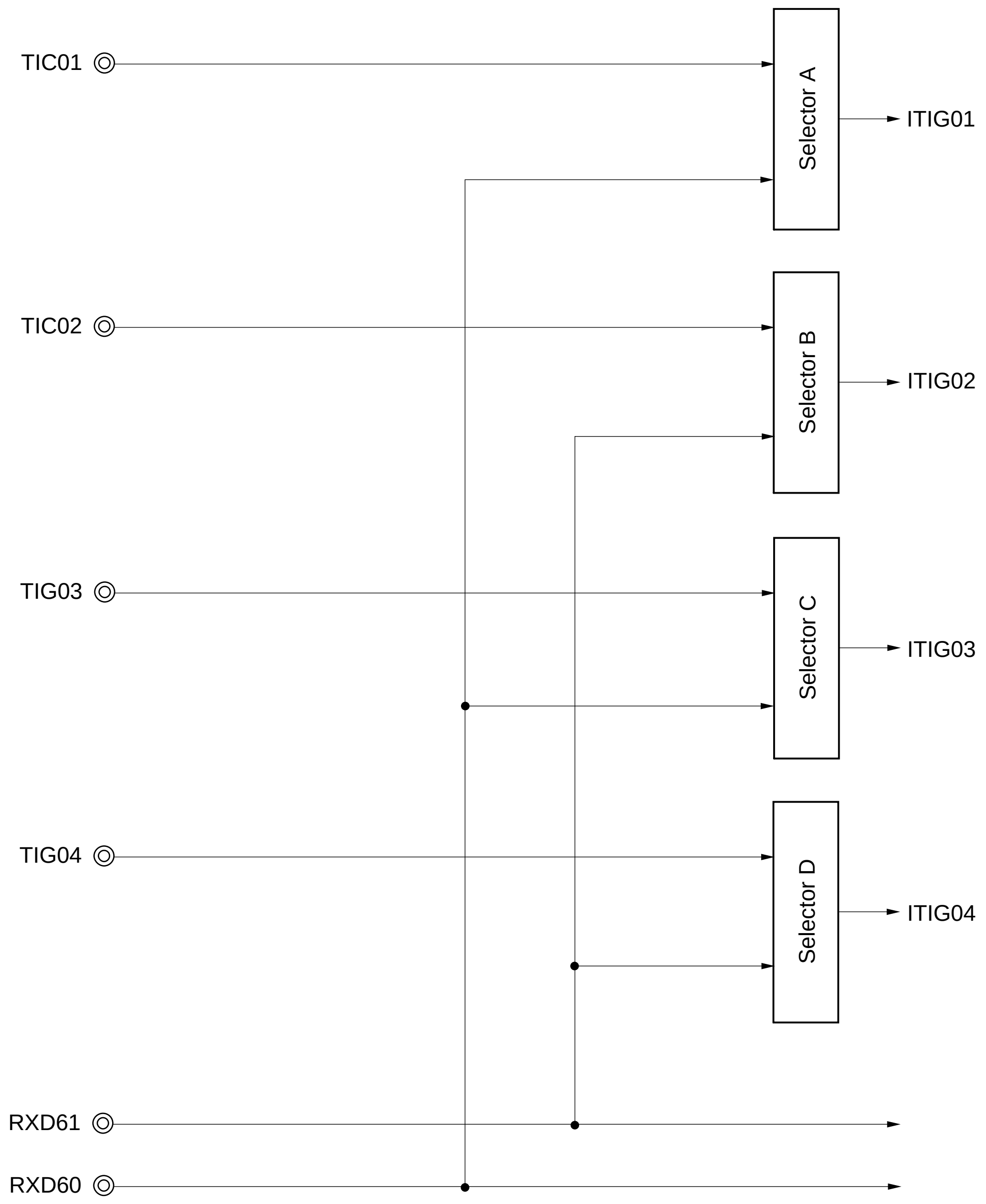
GCCnm register (m = 1 to 4) in Compare mode:
After setting the POWERGn bit you have to wait for 10 peripheral clocks periods (f_{PCLK}) to perform write access to the GCCnm register (m = 1 to 4).
To perform successive write access during operation, for rewriting the GCCnm register (n = 1 to 4), you have to wait for minimum 14 peripheral clocks periods (f_{PCLK}).

7.3 Timer Input Select

7.3.1 UART6n connection to Timer Inputs

Timer input select allow to connect RXD60 & RXD61 pins to Timer G0. Connection is controlled by the TIS register.

Figure 7-37: Timer Input Select Block Diagram



7.3.2 Timer input select control register

Figure 7-38: Timer Input Select Control Register Format

	7	6	5	4	3	2	1	0	Address	After reset
TIS	TISD1	TISD0	TISC1	TISC0	TISB1	TISB0	TISA1	TISA0	FFFF F6F0H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

TISD1	TISD0	Connection to ITIG04
0	0	Input from TIG04
0	1	Input from RXD61
1	0	Reserved.
1	1	Reserved.

TISC1	TISC0	Connection to ITIG03
0	0	Input from TIG03
0	1	Input from RXD60
1	0	Reserved.
1	1	Reserved.

TISB1	TISB0	Connection to ITIG02
0	0	Input from TIG02
0	1	Input from RXD61
1	0	Reserved.
1	1	Reserved.

TISA1	TISA0	Connection to ITIG00
0	0	Input from TIG00
0	1	Input from RXD60
1	0	Reserved.
1	1	Reserved.

[MEMO]

Chapter 8 Watch Timer

8.1 Function

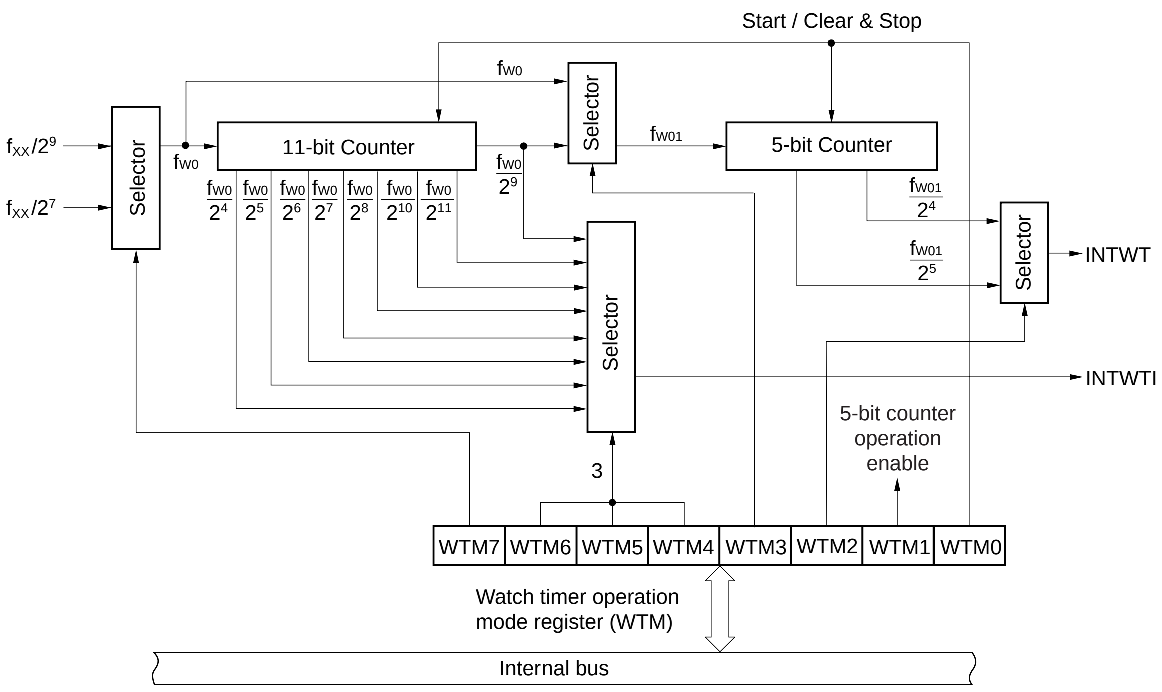
The watch timer has the following functions:

- Watch timer (interrupt intervals from 256 µs up to 1,048 s)
- Interval timer (interrupt intervals from 256 µs up to 131 ms)

The watch timer and interval timer functions can be used at the same time.

Figure 8-1 shows the block diagram of the watch timer.

Figure 8-1: Block Diagram of Watch Timer



Start / Clear & Stop
fW0
fXX/2^9
fXX/2^7
Selector
fW0
11-bit Counter
Selector
fW01
5-bit Counter
fW0/2^9
fW0/2^4 fW0/2^5 fW0/2^6 fW0/2^7 fW0/2^8 fW0/2^10 fW0/2^11
fW01/2^4
fW01/2^5
Selector
INTWT
Selector
INTWTI
5-bit counter operation enable
3
WTM7 WTM6 WTM5 WTM4 WTM3 WTM2 WTM1 WTM0
Watch timer operation mode register (WTM)
Internal bus

(1) Watch timer

The watch timer generates an interrupt request (INTWT) at time intervals of 256 µs to 1,048s by using the system clock prescaler $f_{XX}/2^9$ or $f_{XX}/2^7$.

(2) Interval timer

The interval timer generates an interrupt request (INTWTI) at time intervals of 256 µs to 131 ms.

8.2 Configuration

The watch timer consists of the following hardware:

Table 8-1: Configuration of Watch Timer

Item	Configuration
Counter	5 bits × 1
Prescaler	11 bits × 1
Control register	Watch timer mode control register (WTM)

8.3 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 can be set by an 1-bit or 8-bit memory manipulation instruction.

Caution: When the watch timer is used, the prescaler should not be cleared frequently. When rewriting WTM7 to WTM2 bit to other data, set WTM0 bit to “0” beforehand to stop the timer operation.

Figure 8-2: 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	FFFFF6E0H	R/W	00H

WTM7	Selects main input frequency from prescaler
1	$f_{XX} / 2^7$ (f_{W0} = 62.5 kHz @f_{XX} = 8 MHz)
0	$f_{XX} / 2^9$ (f_{W0} = 15.625 kHz @f_{XX} = 8 MHz)

Figure 8-2: Watch Timer Mode Control Register (WTM) (2/2)

5-bit Prescaler interval time selection for interval timer function						
WTM6	WTM5	WTM4	WTM7 = 1		WTM7 = 0	
			Interrupt frequency	Interval time	Interrupt frequency	Interval time
0	0	0	$f_{XX} / 2^{11}$ (3.906 kHz)	0.256 ms	$f_{XX} / 2^{13}$ (0.976 kHz)	1.024 ms
0	0	1	$f_{XX} / 2^{12}$ (1.953 kHz)	0.512 ms	$f_{XX} / 2^{14}$ (0.488 kHz)	2.048 ms
0	1	0	$f_{XX} / 2^{13}$ (0.976 kHz)	1.024 ms	$f_{XX} / 2^{15}$ (0.244 kHz)	4.096 ms
0	1	1	$f_{XX} / 2^{14}$ (0.488 kHz)	2.048 ms	$f_{XX} / 2^{16}$ (0.122 kHz)	8.192 ms
1	0	0	$f_{XX} / 2^{15}$ (0.244 kHz)	4.096 ms	$f_{XX} / 2^{17}$ (0.061 kHz)	16.384 ms
1	0	1	$f_{XX} / 2^{16}$ (0.122 kHz)	8.192 ms	$f_{XX} / 2^{18}$ (0.030 kHz)	32.768 ms
1	1	0	$f_{XX} / 2^{17}$ (0.061 kHz)	16.384 ms	$f_{XX} / 2^{19}$ (0.015 kHz)	65.536 ms
1	1	1	$f_{XX} / 2^{18}$ (0.030 kHz)	32.768 ms	$f_{XX} / 2^{20}$ (0.007 kHz)	131.072 ms

5-bit Prescaler interval time selection for watch timer function					
WTM3	WTM2	WTM7 = 1		WTM7 = 0	
		Interrupt frequency	Interval time	Interrupt frequency	Interval time
0	0	$f_{XX} / 2^{11}$ (3.906 kHz)	0.256 ms	$f_{XX} / 2^{13}$ (0.976 kHz)	1.024 ms
0	1	$f_{XX} / 2^{12}$ (1.953 kHz)	0.512 ms	$f_{XX} / 2^{14}$ (0.488 kHz)	2.048 ms
1	0	$f_{XX} / 2^{20}$ (7.628 Hz)	0.131 s	$f_{XX} / 2^{22}$ (1.907Hz)	0.524 s
1	1	$f_{XX} / 2^{21}$ (3,814 Hz)	0.262 s	$f_{XX} / 2^{23}$ (0,953Hz)(	1.048 s

WTM1	Controls Operation of Interval Timer
0	Stops operation (clears 5-bit Counter after stop)
1	Starts

WTM0	Enables Operation of the whole Watch Timer
0	Stops operation (clears both prescaler and timer)
1	Enables operation

Notes: 1. WTM7 to WTM2 cannot be changed while the watch timer is operating, so change WTM7 to WTM2 only when WTM0 is 0.

2. Calculated times for system clock frequency f_{XX} = 8.000 MHz

8.4 Operations

8.4.1 Operation as watch timer

The count operation of the first stage 11 bits counter is enabled by setting the WTM0 bit of the watch timer mode control register (WTM) to “1”. (When WTM0 is cleared to “0”, the 11-bit prescaler and 5-bit counter are cleared, and the watch timer stops the count operation.)

The watch timer is then monitored by the bit WTM1:

- when WTM1 is set to “1”, the 5 bits counter starts, and the INTWT interrupt request is generated at each timer overflow.
- when WTM1 is cleared to “0”, the 5 bits counter is cleared and stopped, and no more INTWT is generated.

Due to the cascade architecture of the 2 counters, setting WTM1 asynchronously of the 11 bits counter introduces an error on the first INTWT generation of up to $2^9/f_W$. To avoid this, first set WTM1 to 1 and then set WTM0 to 1.

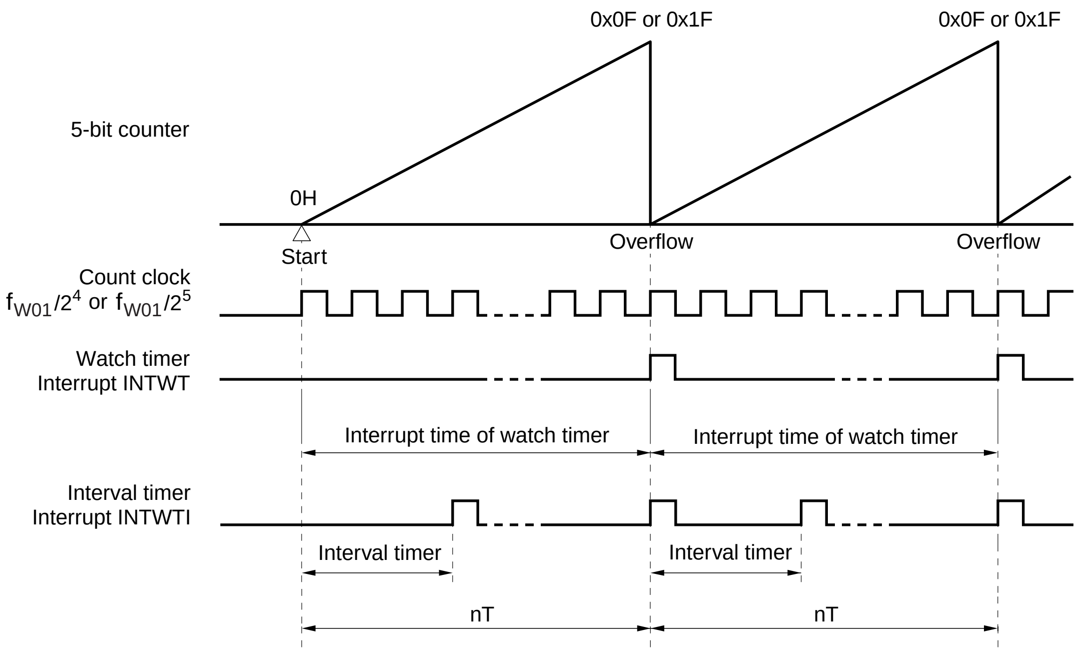
8.4.2 Operation as interval timer

As long as WTM0 is set to 1, the watch timer repeatedly generates the INTWTI interrupt at time intervals specified by the bits WTM6 to WTM4 of the watch timer mode control register WTM.

8.4.3 Watch timer and Interval timer simultaneously

Operation of Watch timer and Interval timer can be used simultaneously.

Figure 8-3: Example Watch Timer and Interval Timer Simultaneously



Remark: f_{WO1}: Watch timer clock frequency

Chapter 9 Watchdog Timer

9.1 Features

The watchdog timer counts an input clock signal derivated from the main oscillator.

Upon overflow, the watchdog timer can generate

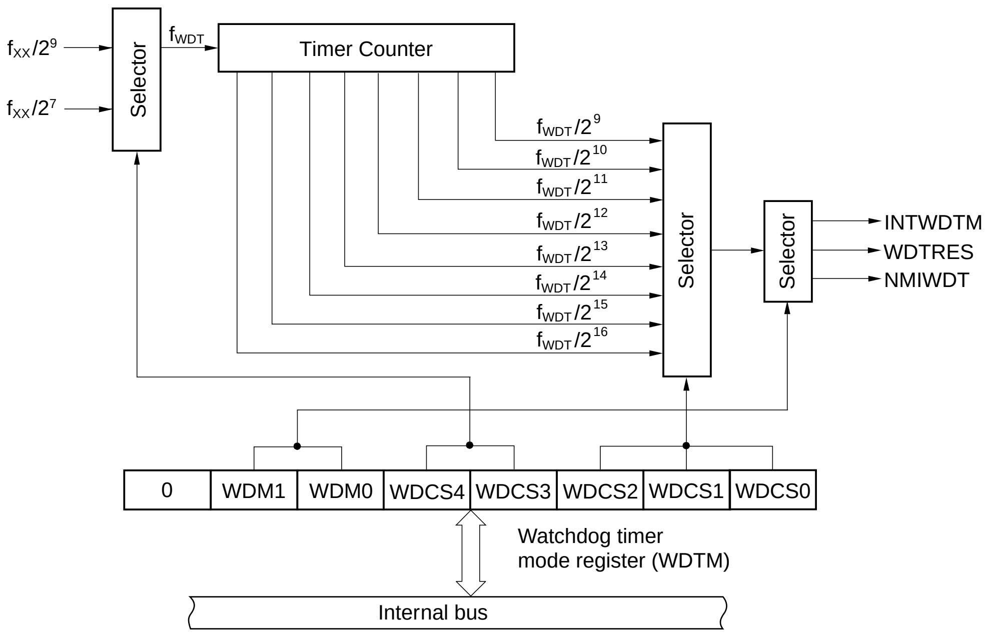
- a non-maskable interrupt request signal (NMIWDT)
- a system reset signal (WDTRES)
- a maskable interrupt request signal (INTWDTM)

The selection of the input clock signal and the type of interrupt request generated is done with the watchdog timer mode register (WDTM).

After a system reset, the watchdog timer is enabled, and starts in the reset mode, i.e. the mode where it generates the WDTRES signal on overflow.
The watchdog timer can be reconfigured only once after system reset.
After the watchdog timer has been reconfigured, any write access to the mode register WDTM will immediately lead to a watchdog timer overflow, and then the selected event (WDTRES, NMIWDT, INTWDTM) is generated.

A dedicated value has to be written to the WDTE register in order to clear and restart the watchdog timer. Any wrong data (different from the dedicated value) written to WDTE will lead to a watchdog timer overflow, and then the selected event (WDTRES, NMIWDT, INTWDTM) is generated.

Figure 9-1: Block Diagram of Watchdog Timer



Remark:	INTWDTM:	Request signal for maskable interrupt through watchdog timer overflow
	NMIWDT:	Request signal for non-maskable interrupt through watchdog timer overflow
	WDTRES:	Reset signal through watchdog timer overflow

9.2 Configuration

Watchdog timer consists of the following hardware.

Table 9-1: Configuration of Watchdog Timer

Item	Configuration
Control register	Watchdog timer mode register (WDTM) Watchdog timer enable register (WDTE)

9.2.1 Watchdog Timer control registers

(1) Watchdog timer mode register (WDTM)

This register sets the overflow time and operation clock of watchdog timer.
WDTM is set with an 8-bit memory manipulation instruction. This register can be read any number of times, but it can be written only once following system reset.
Writing to this register will automatically clear the watchdog timer counter to 0000H.

$\overline{\text{RESET}}$ input sets WDTM to 67H.

Figure 9-2: Watchdog Timer Mode Register (WDTM) Format

	7	6	5	4	3	2	1	0	Address	After reset
WDTM	0	WDM1	WDM0	WDCS4	WDCS3	WDCS2	WDCS1	WDCS0	FFFF F680H	67H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

WDM1	WDM0	Selection of operation mode of watchdog timer
0	0	Maskable interrupt request mode (generation of INTWDTM)
0	1	Non-maskable interrupt request mode (generation of NMIWDT)
1	-	Reset mode (generation of WDTRES)

Cautions: 1. To stop the operation of watchdog timer, write "1FH" to the WDTM register. (this can be done only once after reset)

2. For details about bits WDCS0 to WDCS4, refer to Table 9-2, “Watchdog Timer Clock Selection,” on page 228.

Table 9-2: Watchdog Timer Clock Selection

WDCS4	WDCS3	WDCS2	WDCS1	WDCS0	Count Frequency (Hz)	First Interrupt delay (ms)	Interrupt period (ms)
0	0	0	0	0	$f_{XX}/2^{16}$ (122.07)	8.192	10.240
0	0	0	0	1	$f_{XX}/2^{17}$ (61.035)	16.384	18.432
0	0	0	1	0	$f_{XX}/2^{18}$ (30.517)	32.768	34.816
0	0	0	1	1	$f_{XX}/2^{19}$ (15.258)	65.536	67.584
0	0	1	0	0	$f_{XX}/2^{20}$ (7.6294)	131.072	133.12
0	0	1	0	1	$f_{XX}/2^{21}$ (3.8147)	262.144	264.192
0	0	1	1	0	$f_{XX}/2^{22}$ (1.9073)	524.288	526.336
0	0	1	1	1	$f_{XX}/2^{23}$ (0.9537)	1048.57	1050.62
0	1	0	0	0	$f_{XX}/2^{18}$ (30.517)	32.768	40.960
0	1	0	0	1	$f_{XX}/2^{19}$ (15.258)	65.536	73.728
0	1	0	1	0	$f_{XX}/2^{20}$ (7.6294)	131.072	139.264
0	1	0	1	1	$f_{XX}/2^{21}$ (3.8147)	262.144	270.336
0	1	1	0	0	$f_{XX}/2^{22}$ (1.9073)	524.288	532.480
0	1	1	0	1	$f_{XX}/2^{23}$ (0.9537)	1048.57	1056.76
0	1	1	1	0	$f_{XX}/2^{24}$ (0.4768)	2097.15	2105.34
0	1	1	1	1	$f_{XX}/2^{25}$ (0.2384)	4194.30	4202.49
1	×	×	×	×	Operation stopped		

Remark: Values given for f_{XX} = 8.000 MHz

(2) Watchdog timer enable register (WDTE)

The counter of the watchdog timer is cleared and counting restarted by writing “ACH” to WDTE.
WDTE is set by an 8-bit memory manipulation instruction.
$\overline{\text{RESET}}$ input sets WDTE to 9AH.

Figure 9-3: Watchdog Timer Enable Register (WDTE) Format

	7	6	5	4	3	2	1	0	Address	After reset
WDTE	RUN	0	0	1	1	0	1	0	FFFF F681H	9AH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

Cautions: 1. When a value other than “ACH” is written to the WDTE register, an interrupt is generated, according to the WDM1, WDM0 bits when “RUN” bit was previously set to 1.

2. When a 1-bit memory manipulation instruction is executed for the WDTE register, an interrupt is generated, according to the WDM1, WDM0 bits.

3. The read value of the WDTE register is “9AH” (value that differs from written value “ACH”).

9.3 Operation

Watchdog timer automatically starts in the reset mode following reset release.
The WDTM register can be written only once after system reset. To configure the watchdog timer, write the operation mode and the interval time to the WDTM register using 8-bit memory manipulation instructions. After this is done, the operation of the watchdog timer cannot be stopped.
To configure the watchdog timer as non-operating, write 1FH to the WDTM register.

Watchdog timer can be cleared and restarted by writing “ACH” in WDTE register. Writing any other value in WDTE register will lead to immediate timer overflow and selected interrupt.

9.3.1 Watchdog timer generating a reset

This is the default operating mode of the watchdog timer after hardware reset.
The watchdog timer generates a reset request, if WDTM has been set with WDM1 = 1,

- at overflow of the timer
- when a value different from “ACH” is written to WDTE
- at the second write operation to WDTM
- when bit access is attempted to WDTE

After a hardware reset is input, the watchdog timer starts its operation with the following settings:

- Clock selected as count clock is $f_{XX}/2^7$
- WDTRES selected as output signal
- Maximum overflow period selected (WDCS2 to WDCS0 = 111, period = 1 sec)

9.3.2 Watchdog timer as interval timer

This mode can be selected by writing the corresponding settings in WDTM register.
An interrupt, rather than a reset request, will be generated in the following cases:

- at overflow of the timer
- when a value different from “ACH” is written to WDTE
- at the second (and following) write operation to WDTM
- when bit access is attempted to WDTE

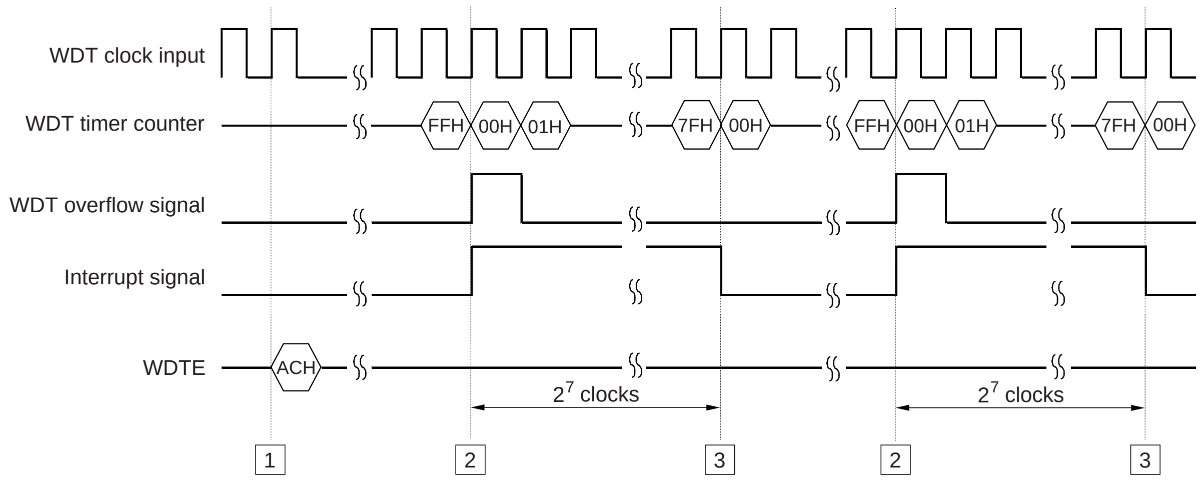
Changing WDTM value can only be done once. Any further write access to WDTM will lead to the selected interrupt generation:

- INTWDTM: maskable interrupt
- NMIWDT: non maskable interrupt

Watchdog timer can be cleared by writing “ACH” in the WDTE register. Writing any other value in WDTE register will lead to immediate timer overflow and selected interrupt generation.

At timer overflow, the watchdog interrupt signal is generated and will set the corresponding flag of the corresponding interrupt register.
This signal will be held for 2^7 watchdog input clock f_{WDT} cycles (8192 µs @ f_{XX}= 8 MHz and f_{WDT}= $f_{XX}/2^9$). Then the watchdog interrupt signal is released, timer is cleared and restarts counting (see figure below).

Figure 9-4: Watchdog Operation at Timer Overflow



1 Timer counter refreshed by writing "ACH" in WDTE

2 Timer overflow - Interrupt signal - timer restart

3 Timer is cleared and restart after 2^7 clocks

Chapter 10 Serial Interface Function

10.1 Features

The serial interface function provides three types of serial interfaces combining a total of 7 transmit/receive channels. All seven channels can be used simultaneously.
The three interface formats are as follows.

- Asynchronous serial interfaces (UART60, UART61): 2 channels
- Clocked serial interfaces (CSI00, CSI01, CSI10): 3 channels
- FCAN controller: 2 channels

Remark: For details about the FCAN controller, refer to Chapter 11 ”FCAN Interface Function” on page 319.

UART60 and UART61, transmit/receive 1-byte serial data following a start bit and support full-duplex communication.
CSI00, CSI01 and CSI10 perform data transfer according to three signals, namely serial clocks ($\overline{\text{SCK00}}$, $\overline{\text{SCK01}}$ and $\overline{\text{SCK10}}$), serial inputs (SI00, SI01 and SI10), and serial outputs (SO00, SO01 and SO10) (3-wire serial I/O).
FCAN conforms to CAN specification version 2.0 Part B, and provides 32-message buffers.

10.2 Serial Interface UART6n

10.2.1 Features

- Transfer rate: 300bps up to 312,5Kbps
- 2 channels (UART60 and UART61)
- Full duplex communications
  - On-chip reception buffer register (RXBn)
  - On-chip transmission buffer register (TXBn)
- Two-pin configuration
  - TXD6n: transmitted data output pin
  - RXD6n: received data input pin
- Reception error detection functions
  - Parity error
  - Framing error
  - Overrun error
- Interrupt sources: 3 types
  - Reception error interrupt (INTSRE6n)
  - Reception completion interrupt (INTSR6n)
  - Transmission completion interrupt (INTST6n)
- Data length of transfer data can be selected from 7 or 8 bits.
- Dedicated internal 8-bit baud rate generator allowing any baud rate to be set
- Twelve operating clock inputs selectable
- MSB- or LSB-first transfer selectable
- Inverted transmission operation selectable
- Tuning break field transmission from 13 to 20 bits
- More than 11 bits can be identified for tuning break field reception (SBF reception flag provided).

Cautions:

1. **The default value of the TXD6n pins is the high level. Take care when using the TXD6n pins as port pins.**
2. **The TXD6n output inversion function inverts only the transmission side and not the reception side. To use this function, the reception side must be ready for reception of inverted data (it must be able to recognize a low-level start bit).**
3. **If clock supply to serial interface UART6n is not stopped (e.g., in HALT mode), normal operation continues. If clock supply to serial interface UART6n is stopped (e.g., in STOP mode), each register stops operating, and holds the value immediately before clock supply was stopped. The TXD6n pins also hold the value immediately before clock supply was stopped and output it. However, the operation is not guaranteed after clock supply is resumed. Therefore, reset the circuit so that POWERn = 0, RXEn = 0, and TXEn = 0.**
4. **If data is continuously transmitted, the transfer rate from the stop bit to the next start bit is delayed two clocks. However, this does not affect the result of transfer because the reception side initializes the timing when it has detected a start bit. Do not use the continuous transmission function when using the LIN protocol.**

Remark: n = 0,1

10.2.2 Functions of serial interfaces UART6n (UART60, UART61)

Serial interfaces UART6n have the following two modes.

(1) Operation STOP mode

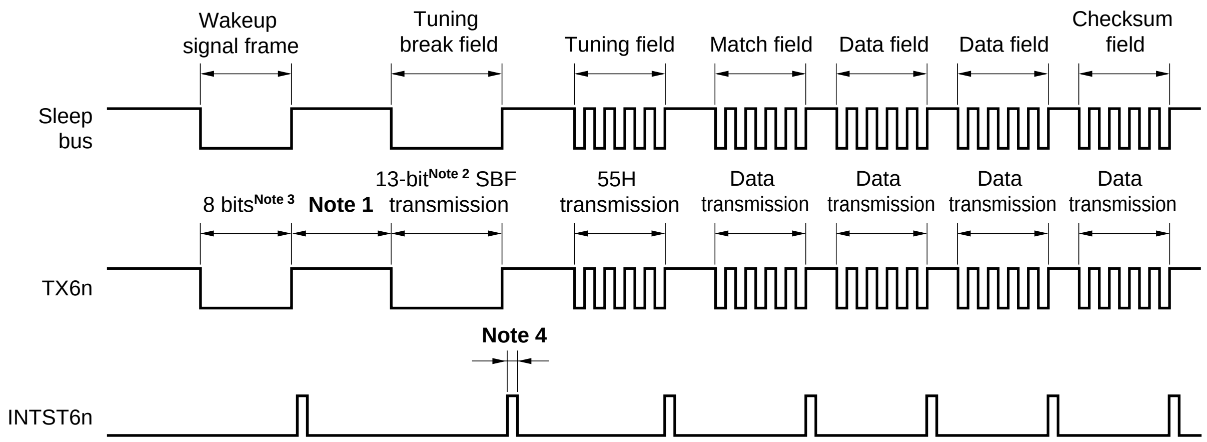
This mode is used when serial transfer is not executed; it can reduce the power consumption. For details, refer to section 10.2.5 (1)"Operation STOP mode" on page 247.

(2) Asynchronous serial interface (UART) mode

This mode supports the LIN (Local Interconnect Network) bus. The functions of this mode are outlined below.

Figures 10-1 and 10-2 outline the transmission and reception operations of LIN.

Figure 10-1: LIN Transmission Operation



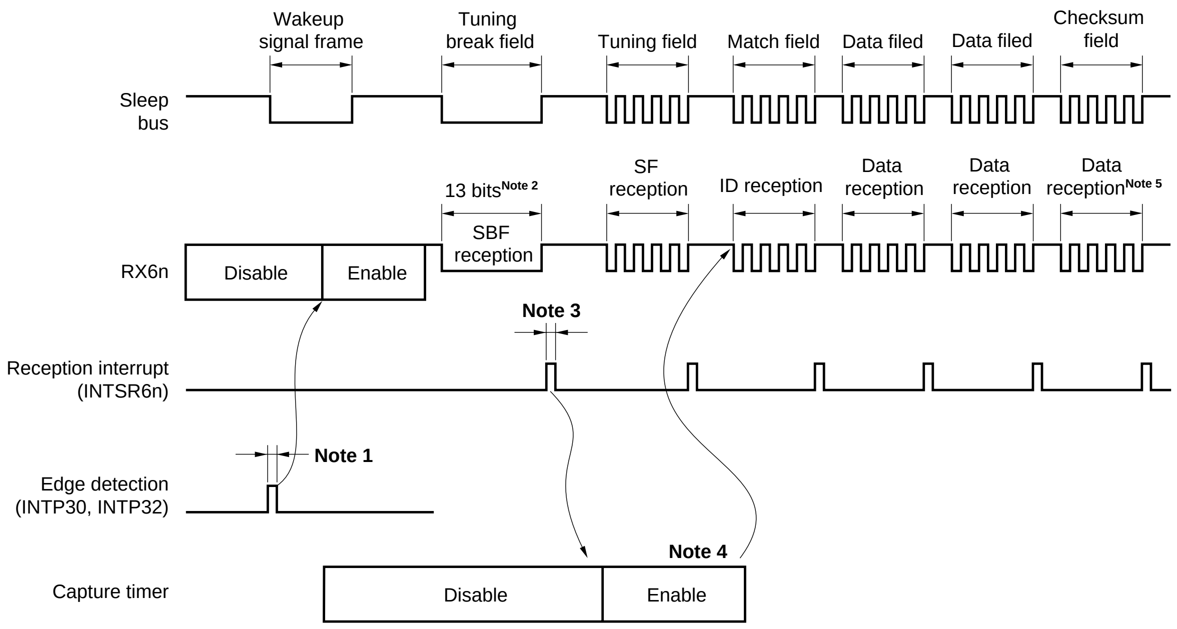
Notes: 1. The interval between each field should be controlled by software.

2. The tuning break field is output by hardware. The output width is equal to the bit length set by bits 4 to 2 (SBL2n to SBL0n) of the asynchronous serial interface control register (ASICLn). If the output width needs to be adjusted more accurately, use dedicated baud rate generator control register (BRGCn).

3. The wake-up signal frame should be generated by software, with a transfer of the value 80H in the 8-bit mode.

4. INTST6n is output on completion of each transmission. It is also output when SBF is transmitted.

Figure 10-2: LIN Reception Operation



Notes: 1. The wake-up signal is detected at the edge of the pin, enables UART6n and sets the SBF reception mode.

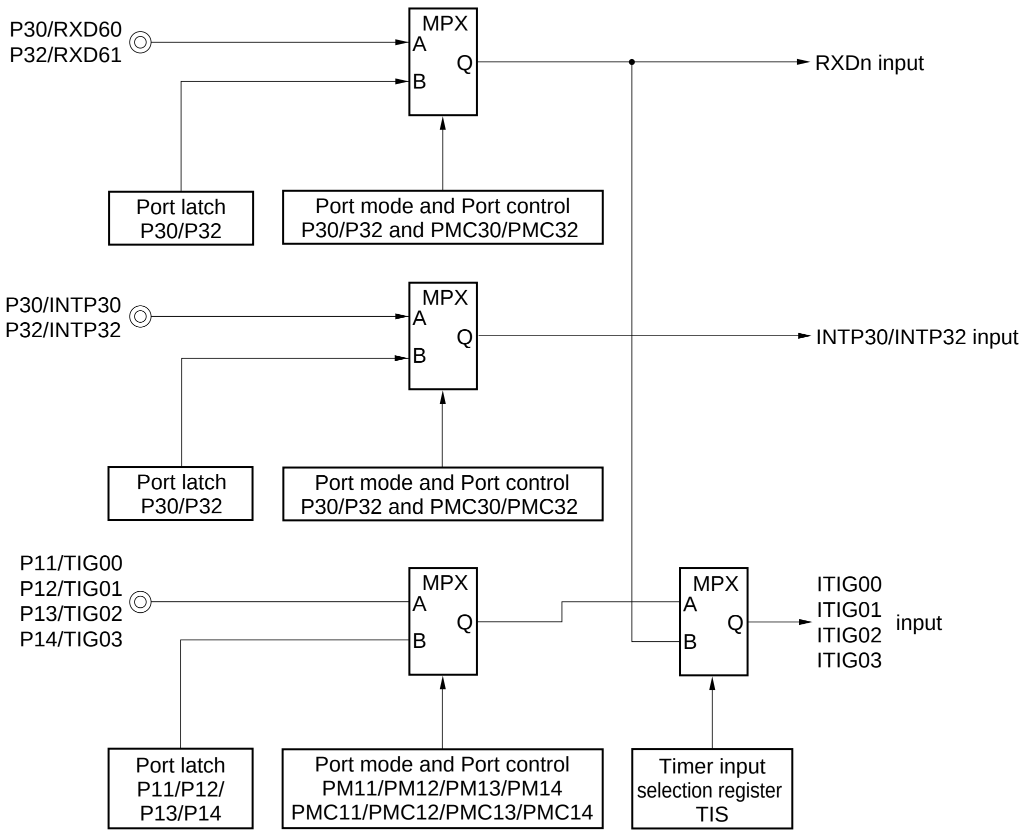
2. Reception continues until the STOP bit is detected. When 11 bits or more of SBF have been detected, it is assumed that SBF reception has been completed correctly, and an interrupt signal is output. If less than 11 bits of SBF have been detected, it is assumed that an SBF reception error has occurred. The interrupt signal is not output and the SBF reception mode is restored.

3. If SBF reception has been completed correctly, an interrupt signal is output. This SBF reception completion interrupt enables the capture timer. Detection of errors OVEn, PEn, and FEn is suppressed, and error detection processing of UART communication and data transfer of the shift register and RXBn is not performed. The shift register holds the reset value FFH.

4. Calculate the baud rate error from the value obtained from the capture timer, disable UART6n after SF reception, and then re-set dedicated baud rate generator control register (BRGCn).

5. Distinguish the checksum field by software. Also perform processing by software to initialize UART6n after reception of the checksum field and to set the SBF reception mode again.

Figure 10-3: Port Configuration for LIN Reception Operation


P30/RXD60
P32/RXD61
MPX
A
B
Q
RXDn input
Port latch P30/P32
Port mode and Port control P30/P32 and PMC30/PMC32
P30/INTP30
P32/INTP32
INTP30/INTP32 input
P11/TIG00
P12/TIG01
P13/TIG02
P14/TIG03
ITIG00
ITIG01
ITIG02
ITIG03
input
Port latch P11/P12/ P13/P14
Port mode and Port control PM11/PM12/PM13/PM14 PMC11/PMC12/PMC13/PMC14
Timer input selection register TIS

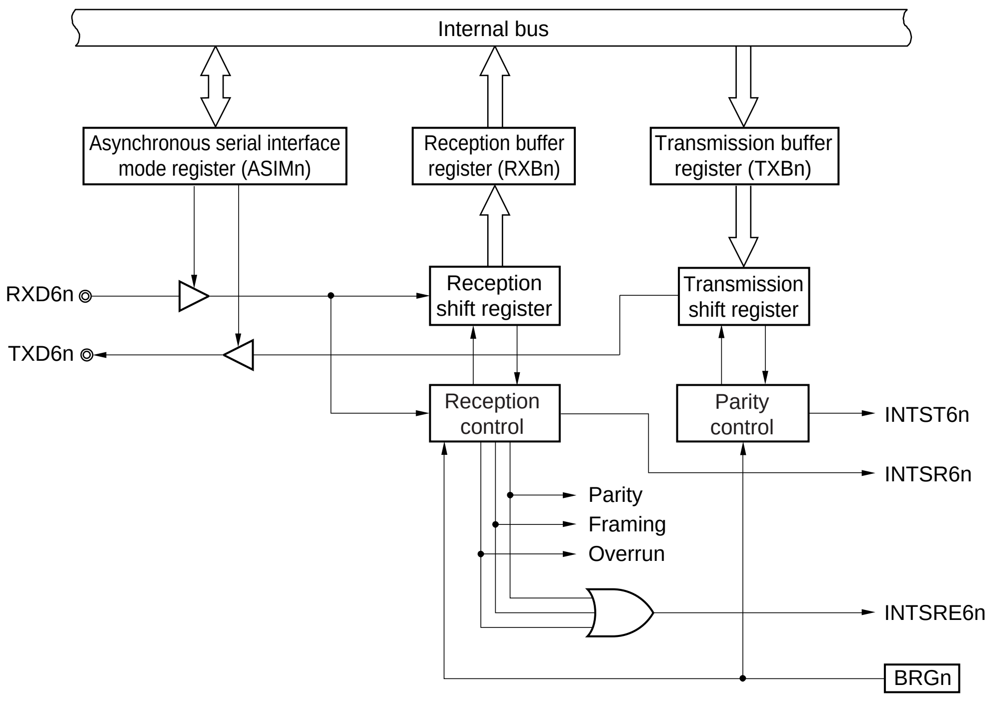
10.2.3 Configuration of Serial Interface UART6n

Serial interface UART6n consists of the following hardware.

Table 10-1: Configuration of Serial Interface UART6n

Item	Configuration
Registers	Receive buffer register (RXBn) Receive shift register (RXSn) Transmit buffer register (TXBn) Transmit shift register (TXSn)
Control registers	Asynchronous serial interface operation mode register (ASIMn) Asynchronous serial interface reception error status register (ASISn) Asynchronous serial interface transmission status register (ASIFn) Clock selection register (CKSRn) Dedicated Baud rate generator control register (BRGCn) Asynchronous serial interface control register (ASICLn)

Figure 10-4: UART60, UART61 Block Diagram


Internal bus
Asynchronous serial interface mode register (ASIMn)
Reception buffer register (RXBn)
Transmission buffer register (TXBn)
RXD6n
TXD6n
Reception shift register
Transmission shift register
Reception control
Parity control
INTST6n
INTSR6n
Parity
Framing
Overrun
INTSRE6n
BRGn

Remark: n = 0, 1

(1) Receive buffer register (RXBn)

This 8-bit register stores parallel data converted by the receive shift register.
Each time 1 byte of data has been received, new received data is transferred to this register from receive shift register (RXSn). If the data length is set to 7 bits, data is transferred as follows.

- In LSB-first reception, the received data is transferred to bits 0 to 6 of RXBn and the MSB of RXBn is always 0.
- In MSB-first reception, the received data is transferred to bits 1 to 7 of RXBn and the LSB of RXBn is always 0.

If an overrun error (OVEn) occurs, the received data is not transferred to RXBn.
RXBn can be read by an 8-bit memory manipulation instruction. No data can be written to this register.

$\overline{\text{RESET}}$ input sets this register to FFH.

Figure 10-5: Format of Receive Buffer Register (RXBn)

	7	6	5	4	3	2	1	0	Address	After reset
RXB0									FFFF FA02H	FFH
	R	R	R	R	R	R	R	R		

	7	6	5	4	3	2	1	0	Address	After reset
RXB1									FFFF FA12H	FFH
	R	R	R	R	R	R	R	R		

(2) Receive shift register (RXSn)

This register converts the serial data input to the RXD6n pin into parallel data.
RXSn cannot be directly manipulated by a program.

(3) Transmit buffer register (TXBn)

This buffer register is used to set transmitted data. Transmission is started when data is written to TXBn. This register can be read or written by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to FFH.

Figure 10-6: Format of Transmit Buffer Register (TXBn)

	7	6	5	4	3	2	1	0	Address	After reset
TXB0									FFFF FA04H	FFH

	7	6	5	4	3	2	1	0	Address	After reset
TXB1									FFFF FA14H	FFH

Cautions: 1. Do not write data to TXBn when bit 1 (TXBFn) of asynchronous serial interface transmission status register (ASIFn) is 1.

2. Do not refresh (write the same value to) TXBn by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of the asynchronous serial interface operation mode register (ASIMn) are 1 or when bit 7 (POWERn) and bit 5 (RXEn) of ASIMn are 1). However, if the same value is continuously transmitted in the transmission mode (POWERn = 1 and TXEn = 1), the same value can be written.

(4) Transmit shift register (TXSn)

This register transmits the data transferred from TXBn from the TXD6n pin as serial data. Data is transferred from TXBn immediately after TXBn is written for the first transmission, or immediately before INTST6n occurs after one frame was transmitted for continuous transmission. Data is transferred from TXBn and transmitted from the TXD6n pin at the falling edge of the internal clock. TXSn cannot be directly manipulated by a program.

10.2.4 Registers Controlling Serial Interface UART6n

Serial interface UART6n is controlled by the following six registers.

- Asynchronous serial interface operation mode register (ASIMn)
- Asynchronous serial interface reception error status register (ASISn)
- Asynchronous serial interface transmission status register (ASIFn)
- Clock selection register (CKSRn)
- Dedicated Baud rate generator control register (BRGCn)
- Asynchronous serial interface control register (ASICLn)

(1) Asynchronous serial interface operation mode register (ASIMn)

This 8-bit register controls the serial transfer operations of serial interface UART6n.
This register can be set by an 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to 01H.

Remark: ASIMn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1).

Figure 10-7: Format of Asynchronous Serial Interface Operation Mode Register (ASIMn) (1/2)

	7	6	5	4	3	2	1	0	Address	After reset
ASIM0	POWER0	TXE0	RXE0	PS10	PS00	CL0	SL0	ISRM0	FFFF FA00H	01H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
ASIM1	POWER1	TXE1	RXE1	PS11	PS01	CL1	SL1	ISRM1	FFFF FA10H	01H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

POWERn	Enables/disables operation of internal operation clock
0[Note 1]	Disables operation of the internal operation clock (fixes the clock to low level) and asynchronously resets the internal circuit.
1[Note 2]	Enables operation of the internal operation clock

TXEn	Enables/disables transmission
0	Disables transmission (synchronously resets the transmission circuit).
1	Enables transmission

Notes: 1. The output of the TXD6n pin goes high and the input from the RXD6n pin is fixed to the high level when POWERn = 0.

2. Operation of the internal operation clock is enabled at the second input clock after 1 is written to the POWERn bit.

Caution: At startup, set POWERn to 1 and then set TXEn to 1. Clear TXEn to 0 first, and then clear POWERn to 0.

Figure 10-7: Format of Asynchronous Serial Interface Operation Mode Register (ASIMn) (2/2)

RXEn	Enables/disables reception
0	Disables reception (synchronously resets the reception circuit).
1	Enables reception

PS1n	PS0n	Transmission operation	Reception operation
0	0	Does not output parity bit.	Reception without parity
0	1	Outputs 0 parity.	Reception as 0 parity[Note]
1	0	Outputs odd parity.	Judges as odd parity.
1	1	Outputs even parity.	Judges as even parity.

CLn	Specifies character length of transmitted/received data
0	Character length of data = 7 bits
1	Character length of data = 8 bits

SLn	Specifies number of stop bits of transmitted data
0	Number of stop bits = 1
1	Number of stop bits = 2

ISRMn	Enables/disables occurrence of reception completion interrupt in case of error
0	“INTSRE6n” occurs in case of error (at this time, INTSR6n does not occur).
1	“INTSR6n” occurs in case of error (at this time, INTSRE6n does not occur).

Note: If “reception as 0 parity” is selected, the parity is not judged. Therefore, bit 2 (PEn) of asynchronous serial interface status register (ASISn) is not set and the error interrupt does not occur.

Cautions:

1. **At startup, set POWERn to 1 and then set RXEn to 1. Clear RXEn to 0 first, and then clear POWERn to 0.**
2. **Clear the TXEn and RXEn bits to 0 before rewriting the PS1n, PS0n, and CLn bits.**
3. **Fix the PS1n and PS0n bits to 0 when using the LIN protocol.**
4. **Make sure that TXEn = 0 when rewriting the SLn bit. Reception is always performed with “the number of stop bits = 1”, and therefore, is not affected by the set value of the SLn bit.**
5. **Make sure that RXEn = 0 when rewriting the ISRMn bit.**

(2) Asynchronous serial interface reception error status register (ASISn)

This register indicates an error status on completion of reception by serial interface UART6n. It includes three error flag bits (PEn, FEn, OVEn).
This read-only register can be read only with 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input clears this register to 00H if bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 0.
00H is read when this register is read.

Figure 10-8: Format of Asynchronous Serial Interface Reception Error Status Register (ASISn)

	7	6	5	4	3	2	1	0	Address	After reset
ASIS0	0	0	0	0	0	PE0	FE0	OVE0	FFFF FA03H	00H
	R	R	R	R	R	R	R	R		

	7	6	5	4	3	2	1	0	Address	After reset
ASIS1	0	0	0	0	0	PE1	FE1	OVE1	FFFF FA13H	00H
	R	R	R	R	R	R	R	R		

PEn	Status flag indicating parity error
0	If POWERn = 0 or RXEn = 0, or after ASISn register is read
1	If the parity of transmitted data does not match the parity bit on completion of reception

FEn	Status flag indicating framing error
0	If POWERn = 0 or RXEn = 0, or after ASISn register is read
1	If the stop bit is not detected on completion of reception

OVEn	Status flag indicating overrun error
0	If POWERn = 0 or RXEn = 0, or after ASISn register is read
1	If received data is set to the RXBn register and the next reception operation is completed before the data is read.

Cautions: 1. The operation of the PEn bit differs depending on the values set in the PS1n and PS0n bits of asynchronous serial interface mode register (ASIMn).

2. The first bit of the received data is checked as the stop bit, regardless of the number of stop bits.

3. If an overrun error occurs, the next received data is not written to receive buffer register (RXBn) but discarded.

4. Because the PEn, FEn, and OVEn bits are cleared when ASISn is read, they may conflict with the setting of the error flags. To avoid the conflict, retry processing is only performed once at most when ASISn is read.

(3) Asynchronous serial interface transmission status register (ASIFn)

This register indicates the status of transmission by serial interface UART6n. It includes two status flag bits (TXBFn and TXSFn).
Transmission can be continued without disruption even during an interrupt period, by writing the next data to the TXBn register after data has been transferred from the TXBn register to the TXSn register.
This read-only register can be read only with 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input clears this register to 00H if bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 0.

Figure 10-9: Format of Asynchronous Serial Interface Transmission Status Register (ASIFn)

	7	6	5	4	3	2	1	0	Address	After reset
ASIF0	0	0	0	0	0	0	TXBF0	TXSF0	FFFF FA05H	00H
	R	R	R	R	R	R	R	R		

	7	6	5	4	3	2	1	0	Address	After reset
ASIF1	0	0	0	0	0	0	TXBF1	TXSF1	FFFF FA15H	00H
	R	R	R	R	R	R	R	R		

TXBFn	Transmit buffer data flag
0	If POWERn = 0 or TXEn = 0, or if no data is present in TxBn register
1	If a data is present in TxBn (has not been transferred to TxSn)

TXSFn	Transmit shift register data flag
0	If POWERn = 0 or TXEn = 0, or if TxSn is empty
1	If data transmission is in progress

Cautions: 1. To continuously transmit data, write the data of the first byte to TXBn, check that the value of the TXBFn flag is 0, and then write the data of the second byte to TXBn. The operation is not guaranteed if data is written to TXBn while the TXBFn flag is 1.

2. While continuous transmission is being executed, check the value of the TXSFn flag after the transmission completion interrupt to determine the subsequent write processing to TXBn.

3. If TXSFn is 1:Continuous transmission is in progress. Data of 1 byte can be written.

4. If TXSFn is 0:Continuous transmission is complete. Data of 2 bytes can be written. When doing so, observe Caution 1 above.

5. While continuous transmission is in progress, check that TXSFn is 0 after the transmission completion interrupt, and then execute clearing (POWERn = 0 or TXEn = 0). If clearing is executed while the TXSFn flag is 1, the transmitted data cannot be guaranteed.

(4) Clock selection register (CKSRn)

This register selects the base clock of serial interface UART6n. CKSRn can be set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input clears this register to 00H.

Remark: CKSRn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1).

Figure 10-10: Format of Clock Selection Register (CKSRn)

	7	6	5	4	3	2	1	0	Address	After reset
CKSR0	0	0	0	0	TPS30	TPS20	TPS10	TPS00	FFFF FA06H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
CKSR1	0	0	0	0	TPS31	TPS21	TPS11	TPS01	FFFF FA16H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

TPS3n	TPS2n	TPS1n	TPS0n	Base clock (f_{XCLK})
0	0	0	0	PCLK (32 MHz)
0	0	0	1	PCLK/2 (16 MHz)
0	0	1	0	PCLK/2^2 (8 MHz)
0	0	1	1	PCLK/2^3 (4 MHz)
0	1	0	0	PCLK/2^4 (2 MHz)
0	1	0	1	PCLK/2^5 (1 MHz)
0	1	1	0	PCLK/2^6 (500 kHz)
0	1	1	1	PCLK/2^7 (250 kHz)
1	0	0	0	PCLK/2^8 (125 kHz)
1	0	0	1	PCLK/2^9 (62.5 kHz)
1	0	1	0	PCLK/2^{10} (31.25 kHz)
1	0	1	1	PCLK/2^{11} (15.625 kHz)
Other				Setting prohibited

Cautions: 1. Make sure POWERn = 0 when rewriting TPS3n to TPS0n.

2. The maximum base clock frequency (f_{XCLK}) to be selected with the Clock selector (TPSn bit) should not exceed 25Mhz. At 32Mhz operating frequency, first configuration of the clock selector is not allowed (CKSRn = 00H).

Remark: Values in parentheses are for operation with external clock f_{XX} = 8 MHz

(5) Dedicated Baud rate generator control register (BRGCn)

This register selects the base clock of serial interface UART6n.
BRGCn can be set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to FFH.

Remark: BRGCn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1).

Figure 10-11: Format of Dedicated Baud Rate Generator Control Register (BRGCn)

	7	6	5	4	3	2	1	0	Address	After reset
BRGC0	MDL70	MDL60	MDL50	MDL40	MDL30	MDL20	MDL10	MDL00	FFFF FA07H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
BRGC1	MDL71	MDL61	MDL51	MDL41	MDL31	MDL21	MDL11	MDL01	FFFF FA17H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

MDL7n	MDL6n	MDL5n	MDL4n	MDL3n	MDL2n	MDL1n	MDL0n	k	Output clock selection of 8-bit counter
Any value less than 04									Setting prohibited
0	0	0	0	0	1	0	0	4	$f_{XCLK}/4$
0	0	0	0	0	1	0	1	5	$f_{XCLK}/5$
0	0	0	0	0	1	1	0	6	$f_{XCLK}/6$
• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •
1	1	1	1	1	1	0	0	252	$f_{XCLK}/252$
1	1	1	1	1	1	0	1	253	$f_{XCLK}/253$
1	1	1	1	1	1	1	0	254	$f_{XCLK}/254$
1	1	1	1	1	1	1	1	255	$f_{XCLK}/255$

Cautions: 1. Make sure that bit 6 (TXEn) and bit 5 (RXEn) of the ASIMn register = 0 when rewriting the MDL7n to MDL0n bits.

2. The baud rate is the output clock of the 8-bit counter divided by 2.

Remarks: 1. f_{XCLK}: Frequency of base clock (Clock) selected by the TPS3n to TPS0n bits of CKSRn register

2. k: Value set by MDL7n to MDL0n bits (k = 4, 5,..., 255)

3. ×: Don't care

(6) Asynchronous serial interface control register (ASICLn)

This register controls the serial transfer operations of serial interface UART6n.
ASICLn can be set by a 1-bit transfer instruction or an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to16H.

Remark: ASICLn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1). However, transfer is started by refresh because bit 6 (SBRTn) and bit 5 (SBTTn) of ASICLn are cleared to 0 when communication is complete (when an interrupt signal is generated).

Figure 10-12: Format of Asynchronous Serial Interface Control Register (ASICLn) (1/2)

	7	6	5	4	3	2	1	0	Address	After reset
ASICL0	SBRF0	SBRT0	SBTT0	SBL20	SBL10	SBL00	DIR0	TXDLV0	FFFF FA08H	16H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
ASICL1	SBRF1	SBRT1	SBTT1	SBL21	SBL11	SBL01	DIR1	TXDLV1	FFFF FA18H	16H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

SBRFn	Status flag
0	If POWERn = 0 or RXEn = 0 or if SBF reception has been completed correctly
1	SBF reception in progress

SBRTn	SBF reception trigger
0	-
1	SBF reception trigger: enables SBF reception detection

Cautions:
1. **In the case of an SBF reception error, return the mode to the SBF reception mode and hold the status of the SBRFn flag.**
2. **Before setting the SBRTn bit, make sure that bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1.**
3. **The read value of the SBRTn bit is always 0. SBRTn is automatically cleared to 0 after SBF reception has been correctly completed.**
4. **Before setting the SBTTn bit to 1, make sure that bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1.**
5. **The read value of the SBTTn bit is always 0. SBTTn is automatically cleared to 0 at the end of SBF transmission.**

Figure 10-12: Format of Asynchronous Serial Interface Control Register (ASICLn) (2/2)

SBTTn	SBF transmission trigger
0	-
1	SBF transmission trigger

SBL2n	SBL1n	SBL0n	SBF transmission output width control
1	0	1	SBF is output with 13-bits length.
1	1	0	SBF is output with 14-bits length.
1	1	1	SBF is output with 15-bits length.
0	0	0	SBF is output with 16-bits length.
0	0	1	SBF is output with 17-bits length.
0	1	0	SBF is output with 18-bits length.
0	1	1	SBF is output with 19-bits length.
1	0	0	SBF is output with 20-bits length.

DIRn	MSB/LSB-first transfer
0	MSB-first transfer
1	LSB-first transfer

TXDLVn	Enables/disables inverting TXD6n output
0	Normal output of TXD6n
1	Inverted output of TXD6n

Caution: Before rewriting the DIRn and TXDLVn bits, clear the TXEn and RXEn bits to 0.

10.2.5 Operation of serial interface UART6n

This section explains the two modes of serial interface UART6n.

(1) Operation STOP mode

In this mode, serial transfer cannot be executed; therefore, the power consumption can be reduced. In addition, the pins can be used as ordinary port pins in this mode.

(a) Register setting

The operation stop mode is set by asynchronous serial interface operation mode register (ASIMn). ASIMn can be set by a 1-bit or 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to 01H.

Remark: ASIMn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1).

Figure 10-13: Register ASIMn in Operation STOP Mode (1/2)

	7	6	5	4	3	2	1	0	Address	After reset
ASIM0	POWER0	TXE0	RXE0	PS10	PS00	CL0	SL0	ISRM0	FFFF FA00H	01H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
ASIM1	POWER1	TXE1	RXE1	PS11	PS01	CL1	SL1	ISRM1	FFFF FA10H	01H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

POWERn	Enables/disables operation of internal operation clock
0 Note 1	Disables operation of the internal operation clock (fixes the clock to low level) and asynchronously resets the internal circuit.
1 Note 2	Enables operation of the internal operation clock

Notes: 1. The output of the TXD6n pin goes high and the input from the RXD6n pin is fixed to the high level when POWERn = 0.

2. Operation of the internal operation clock is enabled at the second input clock after 1 is written to the POWERn bit.

Cautions: 1. At startup, set POWERn to 1 and then set TXEn to 1. Clear TXEn to 0 first, and then clear POWERn to 0.

2. At startup, set POWERn to 1 and then set RXEn to 1. Clear RXEn to 0 first, and then clear POWERn to 0.

Figure 10-13: Register ASIMn in Operation STOP Mode (2/2)

TXEn	Enables/disables transmission
0	Disables transmission (synchronously resets the transmission circuit).
1	Enables transmission

RXEn	Enables/disables reception
0	Disables reception (synchronously resets the reception circuit).
1	Enables reception

Cautions: 1. At startup, set POWERn to 1 and then set TXEn to 1. Clear TXEn to 0 first, and then clear POWERn to 0.

2. At startup, set POWERn to 1 and then set RXEn to 1. Clear RXEn to 0 first, and then clear POWERn to 0.

(2) Asynchronous serial interface (UART) mode

In this mode, data of 1 byte is transmitted/received following a start bit, and a full-duplex operation can be performed.
A dedicated UART baud rate generator is incorporated, so that communication can be executed at a wide range of baud rates.

(a) Register setting

The UART mode is set by asynchronous serial interface operation mode register (ASIMn), asynchronous serial interface reception error status register (ASISn), asynchronous serial interface transmission status register (ASIFn), clock selection register (CKSRn), dedicated baud rate generator control register (BRGCn), and asynchronous serial interface control register (ASICLn).

- **Asynchronous serial interface operation mode register (ASIMn)**

This 8-bit register controls the serial transfer operations of serial interface UART6n.
ASIMn can be set by an 1-bit or an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to 01H.

Remark: ASIMn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1).

Figure 10-14: Register ASIMn in Asynchronous Serial Interface (UART) Mode (1/2)

	7	6	5	4	3	2	1	0	Address	After reset
ASIM0	POWER0	TXE0	RXE0	PS10	PS00	CL0	SL0	ISRM0	FFFF FA00H	01H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
ASIM1	POWER1	TXE1	RXE1	PS11	PS01	CL1	SL1	ISRM1	FFFF FA10H	01H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

POWERn	Enables/disables operation of internal operation clock
0[Note 1]	Disables operation of the internal operation clock (fixes the clock to low level) and asynchronously resets the internal circuit.
1[Note 2]	Enables operation of the internal operation clock

Notes: 1. The output of the TXD6n pin goes high and the input from the RXD6n pin is fixed to the high level when POWERn = 0.

2. Operation of the internal operation clock is enabled at the second input clock after 1 is written to the POWERn bit.

Caution: At startup, set POWERn to 1 and then set TXEn to 1. Clear TXEn to 0 first, and then clear POWERn to 0.

Figure 10-14: Register ASIMn in Asynchronous Serial Interface (UART) Mode (2/2)

TXEn	Enables/disables transmission
0	Disables transmission (synchronously resets the transmission circuit).
1	Enables transmission

RXEn	Enables/disables reception
0	Disables reception (synchronously resets the reception circuit).
1	Enables reception

PS1n	PS0n	Transmission operation	Reception operation
0	0	Does not output parity bit.	Reception without parity
0	1	Outputs 0 parity.	Reception as 0 parity**Note**
1	0	Outputs odd parity.	Judges as odd parity.
1	1	Outputs even parity.	Judges as even parity.

CLn	Specifies character length of transmitted/received data
0	Character length of data = 7 bits
1	Character length of data = 8 bits

SLn	Specifies number of stop bits of transmitted data
0	Number of stop bits = 1
1	Number of stop bits = 2

ISRMn	Enables/disables occurrence of reception completion interrupt in case of error
0	“INTSRE6n” occurs in case of error (at this time, INTSR6n does not occur).
1	“INTSR6n” occurs in case of error (at this time, INTSRE6n does not occur).

Note: If “reception as 0 parity” is selected, the parity is not judged. Therefore, bit 2 (PEn) of asynchronous serial interface status register (ASISn) is not set and the error interrupt does not occur.

Cautions:
1. **At startup, set POWERn to 1 and then set RXEn to 1. Clear RXEn to 0 first, and then clear POWERn to 0.**
2. **Clear the TXEn and RXEn bits to 0 before rewriting the PS1n, PS0n, and CLn bits.**
3. **Fix the PS1n and PS0n bits to 0 when using the LIN protocol.**
4. **Make sure that TXEn = 0 when rewriting the SLn bit. Reception is always performed with "the number of stop bits = 1", and therefore, is not affected by the set value of the SLn bit.**
5. **Make sure that RXEn = 0 when rewriting the ISRMn bit.**

- **Asynchronous serial interface reception error status register (ASISn)**

This register indicates an error status on completion of reception by serial interface UART6n. It includes three error flag bits (PEn, FEn, OVEn).
This read-only register can be set only by 8-bit memory manipulation instructions.

$\overline{\text{RESET}}$ input clears this register to 00H if bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 0.
00H is read when this register is read.

Figure 10-15: Register ASISn in Asynchronous Serial Interface (UART) Mode

	7	6	5	4	3	2	1	0	Address	After reset
ASIS0	0	0	0	0	0	PE0	FE0	OVE0	FFFF FA03H	00H
	R	R	R	R	R	R	R	R		

	7	6	5	4	3	2	1	0	Address	After reset
ASIS1	0	0	0	0	0	PE1	FE1	OVE1	FFFF FA13H	00H
	R	R	R	R	R	R	R	R		

PEn	Status flag indicating parity error
0	If POWERn = 0 or RXEn = 0, or after ASISn register is read
1	If the parity of transmitted data does not match the parity bit on completion of reception

FEn	Status flag indicating framing error
0	If POWERn = 0 or RXEn = 0, or after ASISn register is read
1	If the stop bit is not detected on completion of reception

OVEn	Status flag indicating overrun error
0	If POWERn = 0 or RXEn = 0, or after ASISn register is read
1	If received data is set to the RXBn register and the next reception operation is completed before the data is read.

Cautions: 1. The operation of the PEn bit differs depending on the set values of the PS1n and PS0n bits of asynchronous serial interface mode register (ASIMn).

2. The first bit of the received data is checked as the stop bit, regardless of the number of stop bits.

3. If an overrun error occurs, the next received data is not written to receive buffer register (RXBn) but discarded.

4. Because the PEn, FEn, and OVEn bits are cleared when ASISn is read, they may conflict with the setting of the error flags. To avoid the conflict, retry processing is only performed once at most when ASISn is read.

- **Asynchronous serial interface transmission status register (ASIFn)**

This register indicates the status of transmission by serial interface UART6n. It includes two status flag bits (TXBFn and TXSFn).
Transmission can be continued without disruption even during an interrupt period, by writing the next data to the TXBn register after data has been transferred from the TXBn register to the TXSn register.
This register can be set by an 8-bit memory manipulation instruction, and is read-only.

$\overline{\text{RESET}}$ input clears this register to 00H if bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 0.

Figure 10-16: Register ASIFn in Asynchronous Serial Interface (UART) Mode

	7	6	5	4	3	2	1	0	Address	After reset
ASIF0	0	0	0	0	0	0	TXBF0	TXSF0	FFFF FA05H	00H
	R	R	R	R	R	R	R	R		

	7	6	5	4	3	2	1	0	Address	After reset
ASIF1	0	0	0	0	0	0	TXBF1	TXSF1	FFFF FA15H	00H
	R	R	R	R	R	R	R	R		

TXBFn	Transmit buffer data flag
0	If POWERn = 0 or TXEn = 0, or if no data is present in TXBn
1	If data is written to transmit buffer register (TXBn) (if data exists in TXBn)

TXSFn	Transmit shift register data flag
0	If POWERn = 0 or TXEn = 0, or if TxSn is empty
1	If data transmission is in progress)

Cautions: 1. To continuously transmit data, write the data of the first byte to TXBn, check that the value of the TXBFn flag is 0, and then write the data of the second byte to TXBn. The operation is not guaranteed if data is written to TXBn while the TXBFn flag is 1.

2. While continuous transmission is being executed, check the value of the TXSFn flag after the transmission completion interrupt to determine the subsequent write processing to TXBn.

- **If TXSFn is 1:Continuous transmission is in progress. Data of 1 byte can be written.**
- **If TXSFn is 0:Continuous transmission is complete. Data of 2 bytes can be written. When doing so, observe Caution 1 above.**

3. While continuous transmission is in progress, check that TXSFn is 0 after the transmission completion interrupt, and then execute clearing (POWERn = 0 or TXEn = 0). If clearing is executed while the TXSFn flag is 1, the transmitted data cannot be guaranteed.

- **Asynchronous serial interface control register (ASICLn)**

This register controls the serial transfer operations of serial interface UART6n.
ASICLn can be set by a 1-bit transfer instruction or an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to16H.

Remark: ASICLn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1). However, transfer is started by refresh because bit 6 (SBRTn) and bit 5 (SBTTn) of ASICLn are cleared to 0 when communication is complete (when an interrupt signal is generated).

Figure 10-17: Register ASICLn in Asynchronous Serial Interface (UART) Mode (1/2)

	7	6	5	4	3	2	1	0	Address	After reset
ASICL0	SBRF0	SBRT0	SBTT0	SBL20	SBL10	SBL00	DIR0	TXDLV0	FFFF FA08H	16H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
ASICL1	SBRF1	SBRT1	SBTT1	SBL21	SBL11	SBL01	DIR1	TXDLV1	FFFF FA18H	16H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

SBRFn	SBF reception status flag
0	If POWERn = 0 or RXEn = 0 or if SBF reception has been completed correctly
1	SBF reception in progress

SBRTn	SBF reception trigger
0	-
1	SBF reception trigger: enables SBF reception detection

Cautions:

1. **In the case of an SBF reception error, return the mode to the SBF reception mode and hold the status of the SBRFn flag.**
2. **Before setting the SBRTn bit, make sure that bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1.**
3. **The read value of the SBRTn bit is always 0. SBRTn is automatically cleared to 0 after SBF reception has been correctly completed.**
4. **Before setting the SBTTn bit to 1, make sure that bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1.**
5. **The read value of the SBTTn bit is always 0. SBTTn is automatically cleared to 0 at the end of SBF transmission.**

Figure 10-17: Register ASICLn in Asynchronous Serial Interface (UART) Mode (2/2)

SBTTn	SBF transmission trigger
0	-
1	SBF transmission trigger

SBL2n	SBL1n	SBL0n	SBF transmission output width control
1	0	1	SBF is output with 13-bits length.
1	1	0	SBF is output with 14-bits length.
1	1	1	SBF is output with 15-bits length.
0	0	0	SBF is output with 16-bits length.
0	0	1	SBF is output with 17-bits length.
0	1	0	SBF is output with 18-bits length.
0	1	1	SBF is output with 19-bits length.
1	0	0	SBF is output with 20-bits length.

DIRn	MSB/LSB-first transfer
0	MSB-first transfer
1	LSB-first transfer

TXDLVn	Enables/disables inverting TXD6n output
0	Normal output of TXD6n
1	Inverted output of TXD6n

Caution: Before rewriting the DIRn and TXDLVn bits, clear the TXEn and RXEn bits to 0.

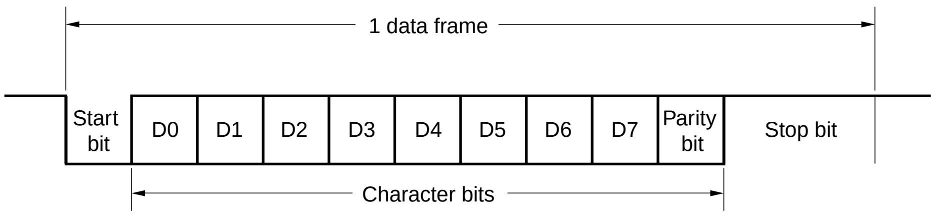
(3) Communication operation

(a) Normal transmitted/received data format

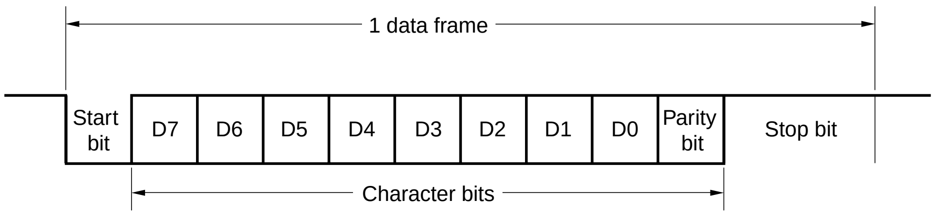
Figure 10-18 shows the format of the transmitted/received data.

Figure 10-18: Format of Normal UART Transmitted/Received Data

(a) LSB-first transmission/reception



(b) MSB-first transmission/reception



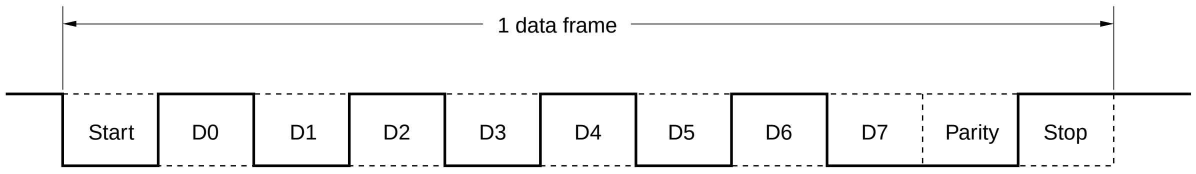
One data frame consists of the following bits.

- Start bit... 1 bit
- Character bits... 7 or 8 bits
- Parity bit... Even parity, odd parity, 0 parity, or no parity
- Stop bit... 1 or 2 bits

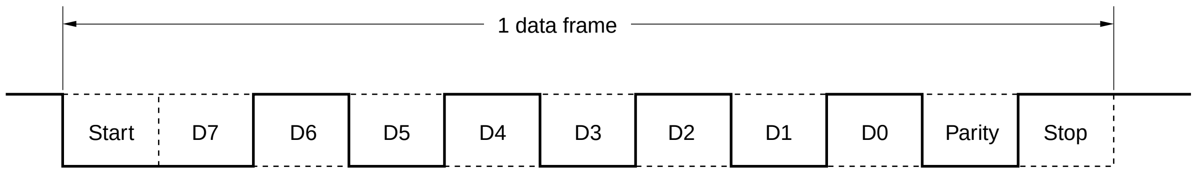
The character bit length, parity, and stop bit length in one data frame are specified by asynchronous serial interface mode register (ASIMn).
Whether data is transferred with the LSB or MSB first is specified by bit 1 (DIRn) of asynchronous serial interface control register (ASICLn).
Whether the TXD6n pin outputs normal or inverted data is specified by bit 0 (TXDLVn) of ASICLn.

Figure 10-19: Example of Normal UART Transmitted/Received Data Format

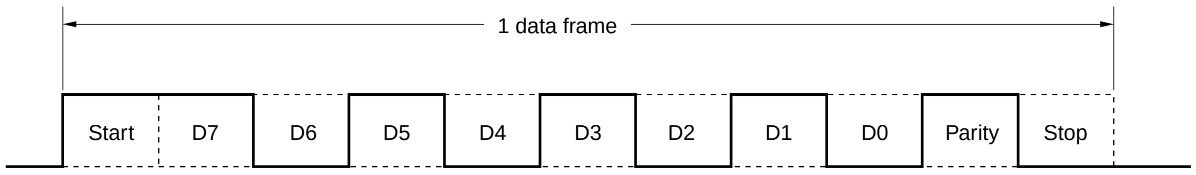
(a) Data length: 8 bits, LSB first, Parity: Even parity, Stop bit: 1 bit, Transfer data: 55H



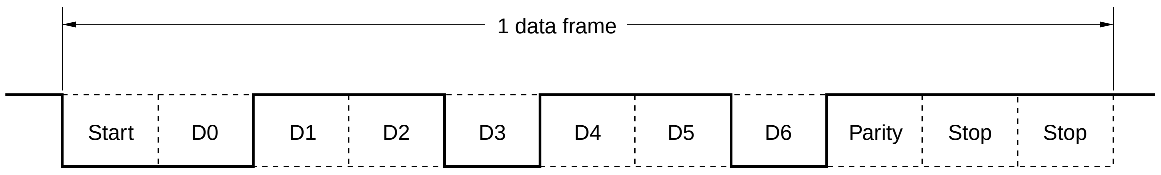
(b) Data length: 8 bits, MSB first, Parity: Even parity, Stop bit: 1 bit, Transfer data: 55H



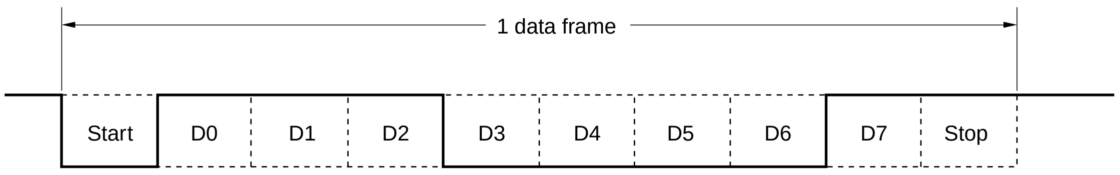
(c) Data length: 8 bits, MSB first, Parity: Even parity, Stop bit: 1 bit, Transfer data: 55H, TXD6n pin inverted output



(d) Data length: 7 bits, LSB first, Parity: Odd parity, Stop bit: 2 bits, Transfer data: 36H



(e) Data length: 8 bits, LSB first, Parity: None, Stop bit: 1 bit, Transfer data: 87H



(b) Parity types and operation

The parity bit is used to detect a bit error in communication data. Usually, the same type of parity bit is used on both the transmission and reception sides. With even parity and odd parity, a 1-bit (odd number) error can be detected. With zero parity and no parity, an error cannot be detected.

Caution: Fix the PS1n and PS0n bits to 0 when using the LIN protocol.

Even parity

- Transmission
  Transmitted data, including the parity bit, is controlled so that the number of bits that are "1" is even.
  The value of the parity bit is as follows.

  If transmitted data has an odd number of bits that are “1”: 1
  If transmitted data has an even number of bits that are “1”: 0

- Reception
  The number of bits that are “1” in the received data, including the parity bit, is counted. If it is odd, a parity error occurs.

Odd parity

- Transmission
  Unlike even parity, transmitted data, including the parity bit, is controlled so that the number of bits that are “1” is odd.

  If transmitted data has an odd number of bits that are “1”: 0
  If transmitted data has an even number of bits that are “1”: 1

- Reception
  The number of bits that are “1” in the received data, including the parity bit, is counted. If it is even, a parity error occurs.

0 parity

The parity bit is cleared to 0 when data is transmitted, regardless of the transmitted data.
The parity bit is not detected when the data is received. Therefore, a parity error does not occur regardless of whether the parity bit is “0” or “1”.

No parity

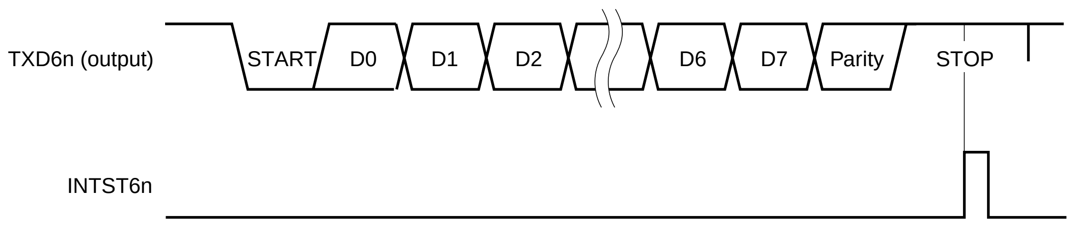
No parity bit is appended to the transmitted data.
Reception is performed assuming that there is no parity bit when data is received. Because there is no parity bit, a parity error does not occur.

(c) Normal transmission

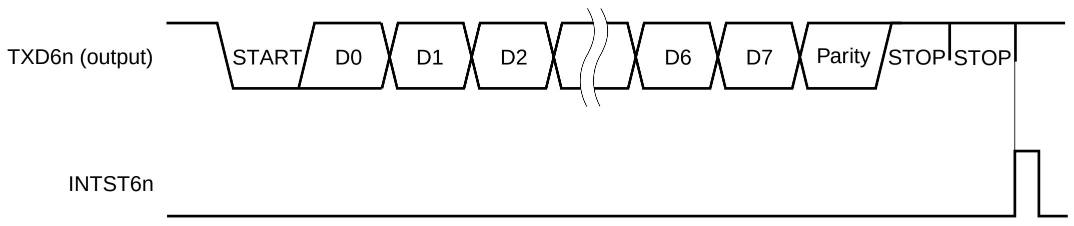
The TXD6n pin outputs a high level when bit 7 (POWERn) of asynchronous serial interface mode register (ASIMn) is set to 1. If bit 6 (TXEn) of ASIMn is then set to 1, transmission is enabled. Transmission can be started by writing transmitted data to transmit buffer register (TXBn). The start bit, parity bit, and stop bit are automatically appended to the data.
When transmission is started, the data in TXBn is transferred to transmit shift register (TXSn). After that, the data is sequential output from TXSn to the TXD6n pin, starting from the LSB/MSB. When transmission is completed, a transmission completion interrupt request (INTST6n) is generated.
Transmission is stopped until the data to be transmitted next is written to TXBn.
Figure 10-20 shows the timing of the transmission completion interrupt request (INTST6n). This interrupt occurs as soon as the last stop bit has been output.

Figure 10-20: Normal Transmission Completion Interrupt Request Timing

(a) Stop bit length: 1



(b) Stop bit length: 2



(d) Continuous transmission

When transmit shift register (TXSn) has started the shift operation, the next transmitted data can be written to transmit buffer register (TXBn). As a result, data can be transmitted without intermission even while an interrupt that has occurred after transmission of one data frame is being serviced, thus an efficient communication rate is realized. To transmit data continuously, however, transmission processing must be executed while referencing bits 1 (TXBFn) and 0 (TXSFn) of asynchronous serial interface transmission status register (ASIFn).

Caution: When using the LIN protocol, the continuous transmission function cannot be used. Make sure that asynchronous serial interface transmission status register (ASIFn) is 00H before writing transmitted data to transmit buffer register (TXBn).

Table 10-2: Write processing and writing to TXBn during execution of continuous transmission

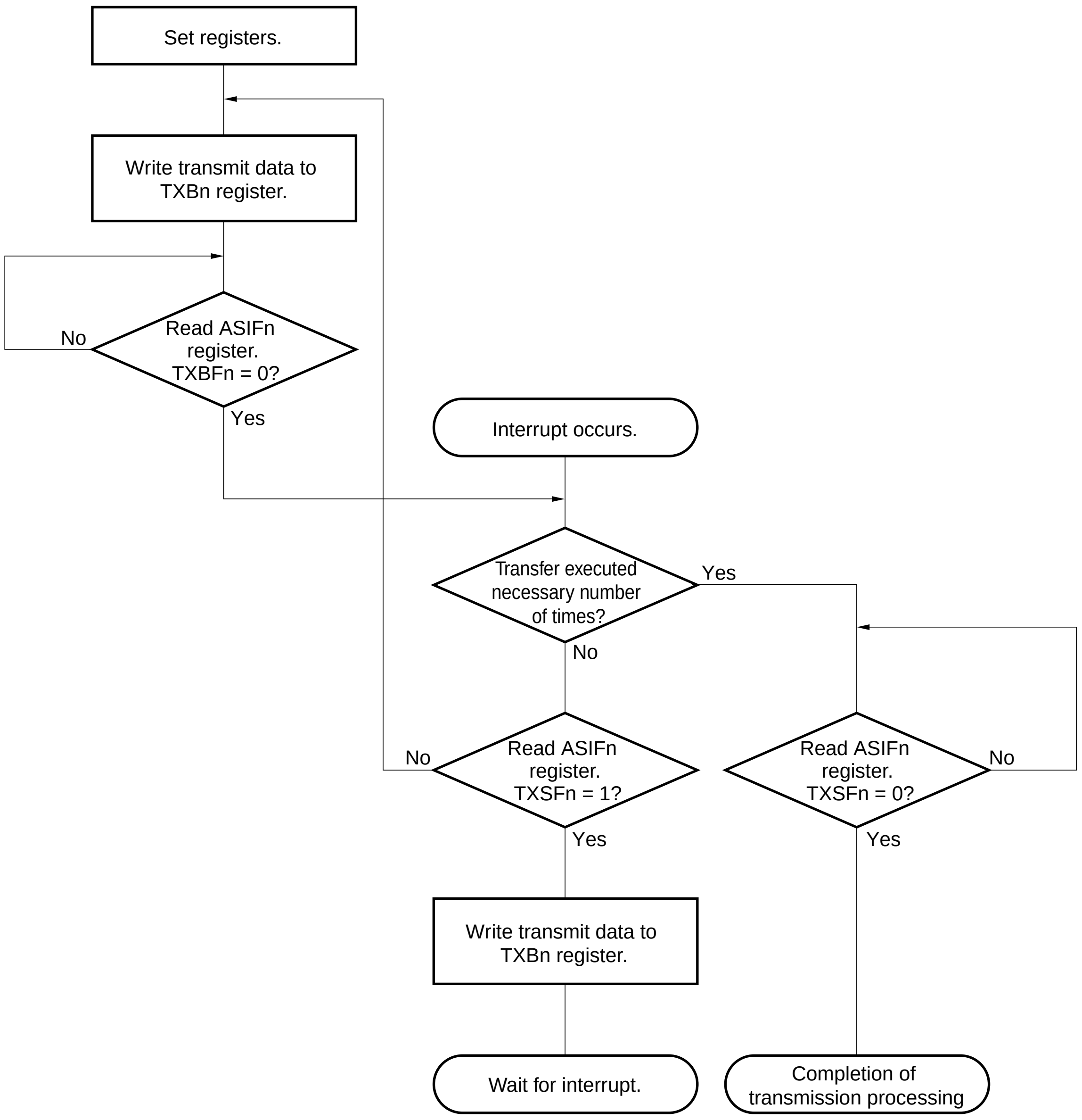
TXBFn	TXSFn	Write Processing During Execution of Continuous Transmission	Writing to TXBn During Execution of Continuous Transmission
0	0	Enables writing 2 bytes or transmission completion processing	Enables writing
0	1	Enables writing 1 byte	Enables writing
1	0	Enables writing 2 bytes or transmission completion processing	Disables writing
1	1	Enables writing 1 byte	Disables writing

Cautions:

1. **To continuously transmit data, write the data of the first byte to TXBn, check that the value of the TXBFn flag is 0, and then write the data of the second byte to TXBn. The operation is not guaranteed if data is written to TXBn while the TXBFn flag is 1.**
2. **While continuous transmission is being executed, check the value of the TXSFn flag after the transmission completion interrupt to determine the subsequent write processing to TXBn.**
   - **If TXSFn is 1: Continuous transmission is in progress. Data of 1 byte can be written.**
   - **If TXSFn is 0: Continuous transmission is completed. Data of 2 bytes can be written. To do so, observe Caution 1 above.**
3. **While continuous transmission is in progress, check that TXSFn is 0 after the transmission completion interrupt, and then execute clearing (POWERn = 0 or TXEn = 0). If clearing is executed while the TXSFn flag is 1, the transmitted data cannot be guaranteed.**

Figure 10-21 shows the processing flow of continuous transmission.

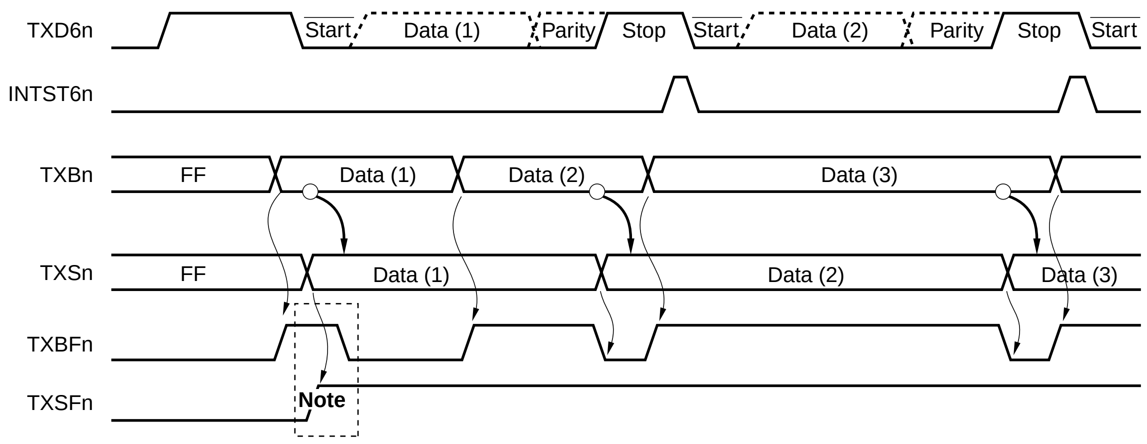
Figure 10-21: Processing Flow of Continuous Transmission


Set registers.
Write transmit data to TXBn register.
Read ASIFn register. TXBFn = 0?
No
Yes
Interrupt occurs.
Transfer executed necessary number of times?
Yes
No
Read ASIFn register. TXSFn = 1?
No
Yes
Read ASIFn register. TXSFn = 0?
No
Yes
Write transmit data to TXBn register.
Wait for interrupt.
Completion of transmission processing

Remark:		
	TXBn:	Transmit buffer register
	ASIFn:	Asynchronous serial interface transmission status register
	TXBFn:	Bit 1 of ASIFn (transmit buffer data flag)
	TXSFn:	Bit 0 of ASIFn (transmit shift register data flag)

Figure 10-22 shows the timing of starting continuous transmission, and Figure 10-23 shows the timing of ending continuous transmission.

Figure 10-22: Timing of Starting Continuous Transmission

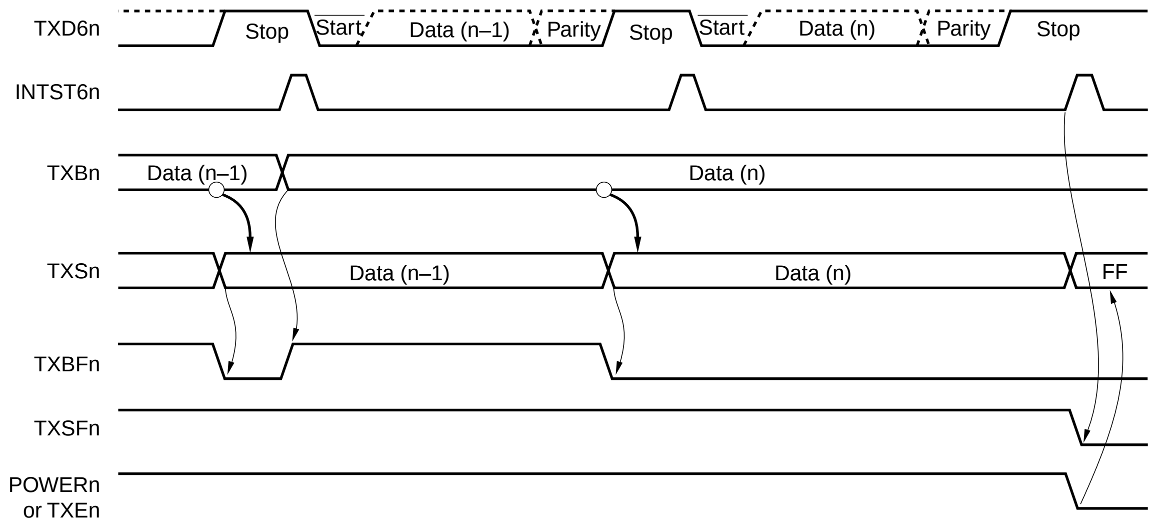


Note: When ASIFn is read, there is a period in which TXBFn and TXSFn = 1, 1. Therefore, judge whether writing is enabled using only the TXBFn bit.

Remark:

TXD6n:	TXD6n pin (output)
INTST6n:	Interrupt request signal
TXBn:	Transmit buffer register
TXSn:	Transmit shift register
ASIFn:	Asynchronous serial interface transmission status register
TXBFn:	Bit 1 of ASIFn
TXSFn:	Bit 0 of ASIFn

Figure 10-23: Timing of Ending Continuous Transmission



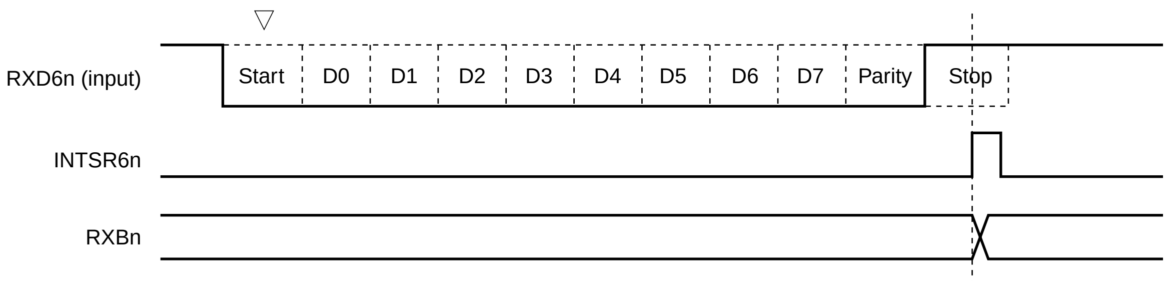
Remark:

TXD6n:	TXD6n pin (output)
INTST6n:	Interrupt request signal
TXBn:	Transmit buffer register
TXSn:	Transmit shift register
ASIFn:	Asynchronous serial interface transmission status register
TXBFn:	Bit 1 of ASIFn
TXSFn:	Bit 0 of ASIFn
POWERn:	Bit 7 of asynchronous serial interface mode register (ASIMn)
TXEn:	Bit 6 of asynchronous serial interface mode register (ASIMn)

(e) Normal reception

Reception is enabled and the RXD6n pin input is sampled when bit 7 (POWERn) of asynchronous serial interface mode register (ASIMn) is set to 1 and then bit 5 (RXEn) of ASIMn is set to 1.
The 8-bit counter of the dedicated baud rate generator starts counting when the falling edge of the RXD6n pin input is detected. When the set value of dedicated baud rate generator control register (BRGCn) has been counted, the RXD6n pin input is sampled again (in Figure 10-24). If the RXD6n pin is low level at this time, it is recognized as a start bit.
When the start bit is detected, reception is started, and serial data is sequential stored in the receive shift register (RXSn) at the set baud rate. When the stop bit has been received, the reception completion interrupt (INTSR6n) is generated and the data of RXSn is written to receive buffer register (RXBn). If an overrun error (OVEn) occurs, however, the received data is not written to RXBn.
Even if a parity error (PEn) or a framing error (FEn) occurs while reception is in progress, reception continues to the reception position of the stop bit, and an error interrupt (INTSR6n/INTSRE6n) is generated on completion of reception.

Figure 10-24: Reception Completion Interrupt Request Timing



Cautions: 1. Be sure to read receive buffer register (RXBn) even if a reception error occurs. Otherwise, an overrun error will occur when the next data is received, and the reception error status will persist.

2. Reception is always performed with the “number of stop bits = 1”. The second stop bit is ignored.

3. Be sure to read asynchronous serial interface reception error status register (ASISn) before reading RXBn.

(f) Reception error

Three types of errors may occur during reception: a parity error, framing error, or overrun error. If the error flag of asynchronous serial interface reception error status register (ASISn) is set as a result of data reception, a reception error interrupt request (INTSR6n/INTSRE6n) is generated.
Which error has occurred during reception can be identified by reading the contents of ASISn in the reception error interrupt servicing (INTSR6n/INTSRE6n) (refer to Table 10-3).
The contents of ASISn are reset to 0 when ASISn is read.

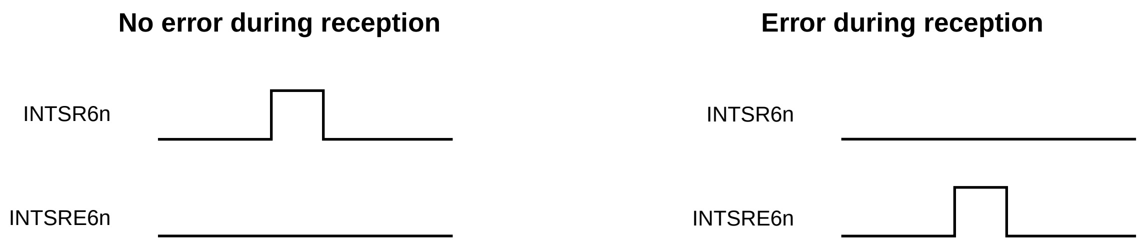
Table 10-3: Cause of Reception Error

Reception Error	Cause	Value of ASISn
Parity error	The parity specified for transmission does not match the parity of the received data.	04H
Framing error	Stop bit is not detected.	02H
Overrun error	Reception of the next data is completed before data is read from receive buffer register (RXBn).	01H

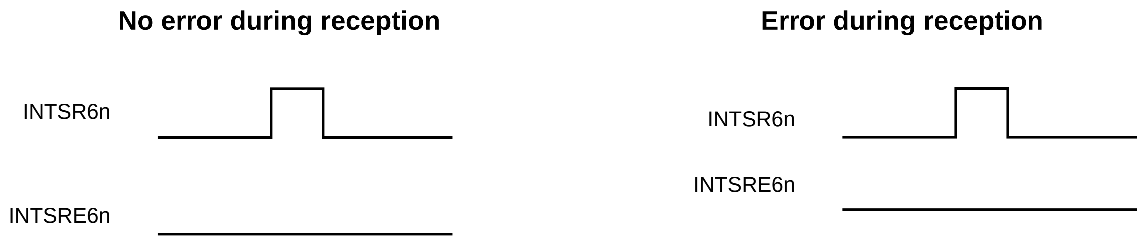
The error interrupt can be separated into INTSR6n and INTSRE6n by clearing bit 0 (ISRMn) of asynchronous serial interface mode register (ASIMn) to 0.

Figure 10-25: Reception Error Interrupt

(a) If ISRMn is cleared to 0 (INTSR6n and INTSRE6n are separated)



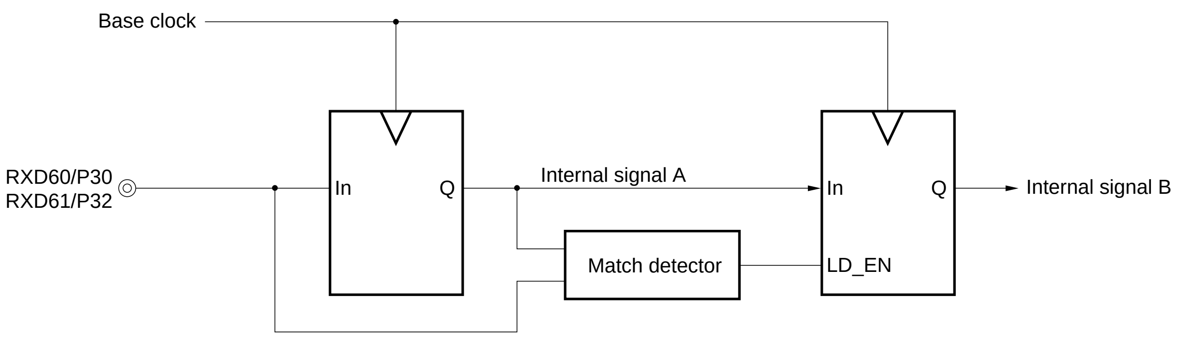
(b) If ISRMn is set to 1 (error interrupt is included in INTSR6n)



(g) Noise filter of received data

The RXD6n signal is sampled with the base clock output by the prescaler block.
If two sampled values are the same, the output of the match detector changes, and the data is sampled as input data.
Because the circuit is configured as shown in Figure 10-26, the internal processing of the reception operation is delayed by two clocks from the external signal status.

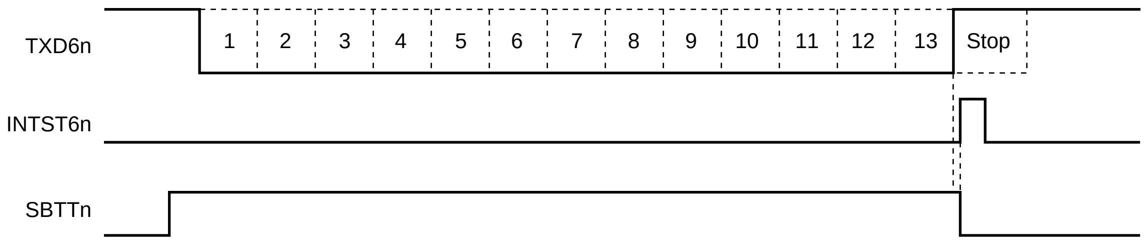
Figure 10-26: Noise Filter Circuit



(h) SBF transmission

When using the LIN protocol, the SBF (Synchronous Break Field) transmission control function is used for transmission. For the transmission operation of LIN protocol, refer to **Figure 10-1, "LIN Transmission Operation," on page 233**.
The TXD6n pin outputs a high level when bit 7 (POWERn) of asynchronous serial interface mode register (ASIMn) is set to 1. Transmission is enabled when bit 6 (TXEn) of ASIMn is set to 1 next time, and SBF transmission operation is started when bit 5 (SBTTn) of asynchronous serial interface control register (ASICLn) is set to 1.
After transmission has been started, the low level for bits 13 to 20 (set by bits 4 to 2 (SBL2n to SBL0n) of ASICLn) is output. When SBF transmission has been completed, a transmission completion interrupt request (INTST6n) is generated, and SBTTn is automatically cleared. After SBF transmission has been completed, the normal transmission mode is restored.
Transmission is stopped until the data to be transmitted next is written to transmit buffer register (TXBn) or SBTTn is set to 1.

Figure 10-27: SBF Transmission



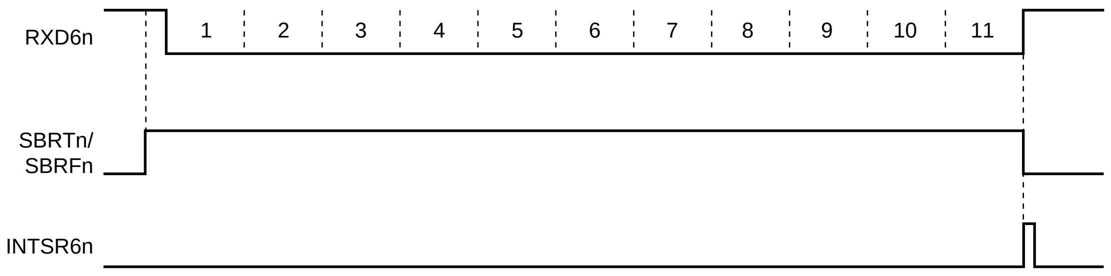
Remark: TXD6n: TXD6n pin (output)
INTST6n: Transmission completion interrupt request
SBTTn: Bit 5 of asynchronous serial interface control register (ASICLn)

(i) SBF reception

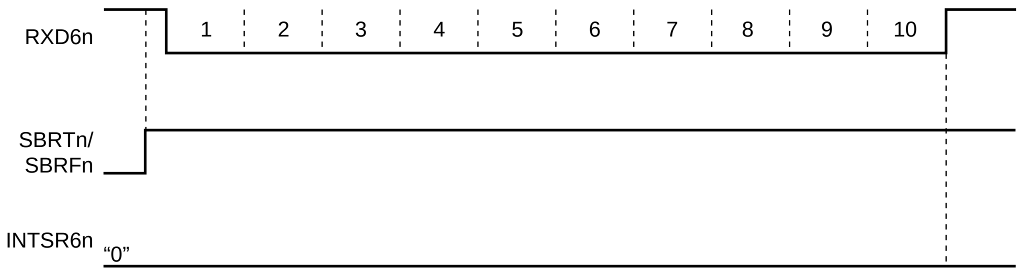
When the device is incorporated in LIN, the SBF (Synchronous Break Field) reception control function is used for reception. For the reception operation of LIN protocol, refer to **Figure 10-2, “LIN Reception Operation,” on page 234**.
Reception is enabled when bit 7 (POWERn) of asynchronous serial interface mode register (ASIMn) is set to 1 and then bit 5 (RXEn) of ASIMn is set to 1. SBF reception is enabled when bit 6 (SBRTn) of asynchronous serial interface control register (ASICLn) is set to 1. In the SBF reception enabled status, the RXD6n pin is sampled and the start bit is detected in the same manner as the normal reception enable status.
When the start bit has been detected, reception is started, and serial data is sequential stored in the receive shift register (RXSn) at the set baud rate. When the stop bit is received and if the width of SBF is 11 bits or more, a reception completion interrupt request (INTSR6n) is generated as normal processing. At this time, the SBRFn and SBRTn bits are automatically cleared, and SBF reception ends. Detection of errors, such as OVEn, PEn, and FEn (bits 0 to 2 of asynchronous serial interface reception error status register (ASISn)) is suppressed, and error detection processing of UART communication is not performed. In addition, data transfer between receive shift register (RXSn) and receive buffer register (RXBn) is not performed, and the reset value of FFH is retained. If the width of SBF is 10 bits or less, an interrupt does not occur as error processing after the stop bit has been received, and the SBF reception mode is restored. In this case, the SBRFn and SBRTn bits are not cleared.

Figure 10-28: SBF Reception

(a) Normal SBF reception (stop bit is detected with a width of more than 10.5 bits)



(b) SBF reception error (stop bit is detected with a width of 10.5 bits or less)



Remark:	RXD6n:	RXD6n pin (input)
	SBRTn:	Bit 6 of asynchronous serial interface control register (ASICLn)
	SBRFn:	Bit 7 of ASICLn
	INTSR6n:	Reception completion interrupt request

10.2.6 Dedicated baud rate generator

The dedicated baud rate generator consists of a source clock selector and an 8-bit programmable counter, and generates a serial clock for transmission/reception of UART6n.
Separate 8-bit counters are provided for transmission and reception.

(1) Configuration of dedicated baud rate generator

- Base clock (Clock)

  The clock selected by bits 3 to 0 (TPS3n to TPS0n) of clock selection register (CKSRn) is supplied to each module when bit 7 (POWERn) of the asynchronous serial interface mode register (ASIMn) is 1. This clock is called the base clock (Clock) and its frequency is called f_{XCLK}. Clock is fixed to the low level when POWERn = 0.

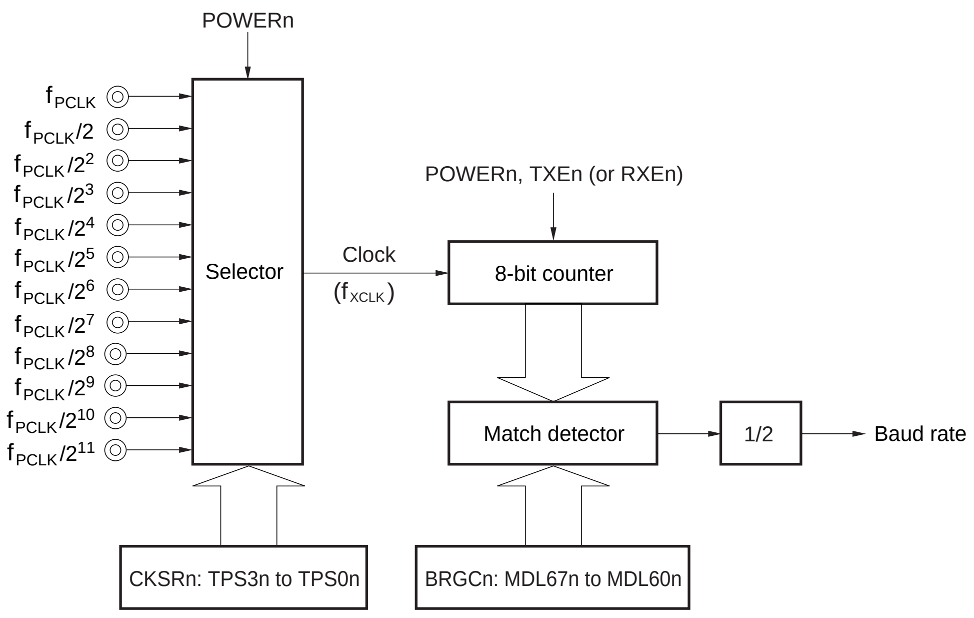
- Transmission counter

  This counter stops, cleared to 0, when bit 7 (POWERn) or bit 6 (TXEn) of asynchronous serial interface mode register (ASIMn) is 0.
  It starts counting when POWERn = 1 and TXEn = 1.
  The counter is cleared to 0 when the first data transmitted is written to transmit buffer register (TXBn).
  If data are continuously transmitted, the counter is cleared to 0 again when one frame of data has been completely transmitted. If there is no data to be transmitted next, the counter is not cleared to 0 and continues counting until POWERn or TXEn is cleared to 0.

- Reception counter

  This counter stops operation, cleared to 0, when bit 7 (POWERn) or bit 5 (RXEn) of asynchronous serial interface mode register (ASIMn) is 0.
  It starts counting when the start bit has been detected.
  The counter stops operation after one frame has been received, until the next start bit is detected.

Figure 10-29: Configuration of Dedicated Baud Rate Generator



Remark:

POWERn:	Bit 7 of asynchronous serial interface mode register (ASIMn)
TXEn:	Bit 6 of ASIMn
RXEn:	Bit 5 of ASIMn
CKSRn:	Clock selection register
BRGCn:	Dedicated Baud rate generator control register

(2) Generation of serial clock

A serial clock can be generated by using clock selection register (CKSRn) and dedicated baud rate generator control register (BRGCn).
Select the clock to be input to the 8-bit counter by using bits 3 to 0 (TPS3n to TPS0n) of CKSRn.
Bits 7 to 0 (BRG7 to BRG0) of BRGCn can be used to select the division value of the 8-bit counter.

(a) Clock selection register (CKSRn)

This register selects the base clock of serial interface UART6n. CKSRn can be set by an 8-bit memory manipulation instruction. $\overline{\text{RESET}}$ input clears this register to 00H.

Remark: CKSRn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1).

Figure 10-30: Clock Selection Register (CKSRn) Format

	7	6	5	4	3	2	1	0	Address	After reset
CKSR0	0	0	0	0	TPS30	TPS20	TPS10	TPS00	FFFF FA06H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
CKSR1	0	0	0	0	TPS31	TPS21	TPS11	TPS01	FFFF FA16H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

TPS3n	TPS2n	TPS1n	TPS0n	Base clock (f_{XCLK})
0	0	0	0	PCLK (32 MHz)
0	0	0	1	PCLK/2 (16 MHz)
^{2}	0	1	0	PCLK/2^2 (8 MHz)
0	0	1	1	PCLK/2^3 (4 MHz)
0	1	0	0	PCLK/2^4 (2 MHz)
0	1	0	1	PCLK/2^5 (1 MHz)
0	1	1	0	PCLK/2^6 (500 kHz)
0	1	1	1	PCLK/2^7 (250 kHz)
1	0	0	0	PCLK/2^8 (125 kHz)
1	0	0	1	PCLK/2^9 (62.5 kHz)
1	0	1	0	PCLK/2^{10} (31.25 kHz)
1	0	1	1	PCLK/2^{11} (15.625 kHz)
Other				Setting prohibited

Cautions: 1. Make sure POWERn = 0 when changing TPS3n to TPS0n.

2. The maximum base clock frequency (f_{XCLK}) to be selected with the Clock selector (TPSn bit) should not exceed 25Mhz. At 32Mhz operating frequency, first configuration of the clock selector is not allowed (CKSRn = 00H).

Remarks: 1. Figures in parentheses are for operation with external clock f_{XX} = 8 MHz

2. f_{XX}: X1 input clock oscillation frequency

(b) Dedicated Baud rate generator control register (BRGCn)

This register selects the base clock of serial interface UART6n.
BRGCn can be set by an 8-bit memory manipulation instruction.

$\overline{\text{RESET}}$ input sets this register to FFH.

Remark: BRGCn can be refreshed (the same value is written) by software during a communication operation (when bit 7 (POWERn) and bit 6 (TXEn) of ASIMn = 1 or bit 7 (POWERn) and bit 5 (RXEn) of ASIMn = 1).

Figure 10-31: Dedicated Baud Rate Generator Control Register (BRGCn) Format

	7	6	5	4	3	2	1	0	Address	After reset
BRGC0	MDL70	MDL60	MDL50	MDL40	MDL30	MDL20	MDL10	MDL00	FFFF FA07H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

	7	6	5	4	3	2	1	0	Address	After reset
BRGC1	MDL71	MDL61	MDL51	MDL41	MDL31	MDL21	MDL11	MDL01	FFFF FA17H	FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

MDL7n	MDL6n	MDL5n	MDL4n	MDL3n	MDL2n	MDL1n	MDL0n	k	Output clock selection of 8-bit counter
any value less than 04									Setting prohibited
0	0	0	0	0	1	0	0	4	$f_{XCLK}/4$
0	0	0	0	0	1	0	1	5	$f_{XCLK}/5$
0	0	0	0	0	1	1	0	6	$f_{XCLK}/6$
• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •	• • • • •
1	1	1	1	1	1	0	0	252	$f_{XCLK}/252$
1	1	1	1	1	1	0	1	253	$f_{XCLK}/253$
1	1	1	1	1	1	1	0	254	$f_{XCLK}/254$
1	1	1	1	1	1	1	1	255	$f_{XCLK}/255$

Cautions: 1. Make sure that bit 6 (TXEn) and bit 5 (RXEn) of the ASIMn register = 0 when changing the MDL7n to MDL0n bits.

2. The baud rate is the output clock of the 8-bit counter divided by 2.

Remarks: 1. f_{XCLK}: Frequency of base clock (Clock) selected by the TPS3n to TPS0n bits of CKSRn register

2. k: Value set by MDL7n to MDL0n bits (k = 4, 5,..., 255)

3. $\times$: Don't care

(c) Baud rate

The baud rate can be calculated by the following expression.

- Baud rate = f_{XCLK} / (2 x k) [bps], or k = INT (f_{XCLK} / (2 x target baud rate))

f_{XCLK}:	Frequency of base clock selected by TPS3n to TPS0n bits of CKSRn register
k:	Value set by MDL7n to MDL0n bits of BRGCn register (k = 4, 5,..., 255)

(d) Error of baud rate

The baud rate error can be calculated by the following expression.

- Error (%) = ((effective baud rate / target baud rate) - 1) $\times$ 100 [%]

Cautions: 1. Keep the baud rate error during transmission to within the permissible error range at the reception destination.

2. Make sure that the baud rate error during reception satisfies the range shown in (4) Permissible baud rate range during reception.

Example:

Frequency of base clock (Clock) = 16 MHz = 16,000,000 Hz
Target baud rate = 153,600 bps
The baud rate formula gives k = 52

Set value of MDL7n to MDL0n bits of BRGCn register = 00110100B (k = 52)
Effective baud rate = 16,000,000/(2 $\times$ 52) = 153,846 bps

Error = ((153846/153600)-1)$\times$100

= 0.160 [%]

(3) Example of setting baud rate

f_{XX} is the external quartz frequency, PLLEN = 1 and PDIV = 0.

Table 10-4: Set Data of Dedicated Baud Rate Generator

Baud Rate [bps]	f_{XX} = 8.0 MHz f_{XCLK} = 16 MH				f_{XX} = 6.0 MHz f_{XCLK} = 24 MHz			
	TPS3n to TPS0n	k	Effective Baud Rate	ERR[%]	TPS3n to TPS0n	k	Effective Baud Rate	ERR[%]
600	7	104	600.96	0.16	7	156	600.96	0.16
1200	6	104	1201.92	0.16	6	156	1201.92	0.16
2400	5	104	2403.85	0.16	5	156	2403.85	0.16
4800	4	104	4807.69	0.16	4	156	4807.69	0.16
9600	3	104	9615.38	0.16	3	156	9615.38	0.16
10400	3	96	10416.67	0.16	3	144	10416.67	0.16
19200	2	104	19230.77	0.16	2	156	19230.77	0.16
31250	2	64	31250	0	1	192	31250	0
38400	1	104	38461.54	0.16	1	156	38461.54	0.16
76800	0	104	76923.08	0.16	0	156	76923.08	0.16
115200	0	69	115942.02	0.64	0	104	115384.62	0.16
153600	0	52	153846.15	0.16	0	78	153846.15	0.16
230400	0	35	228571.43	-0.79	0	52	230769.23	0.16
312500	0	26	307692.31	-1.54	0	38	315789.47	1.05

Caution: The maximum permissible frequency (f_{XCLK}) of the base clock is 25 MHz.
When Fxx=8Mhz, the maximum permissible frequency (f_{XCLK}) of the base is 16Mhz.

Remarks: **1.** TPS3n to TPS0n: Bits 3 to 0 of clock selection register (CKSRn) (setting of base clock (f_{XCLK}))

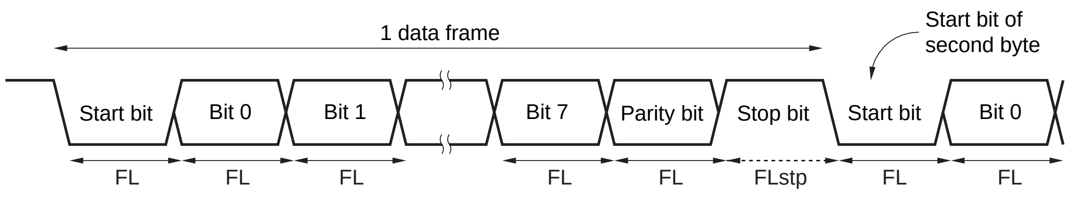
2. k: Value set by MDL7n to MDL0n bits of dedicated baud rate generator control register (BRGCn) (k = 4, 5,..., 255)

3. ERR: Baud rate error

(4) Transfer rate during continuous transmission

When data is continuously transmitted, the transfer rate from a stop bit to the next start bit is extended by two clocks from the normal value. However, the result of transfer is not affected because the timing is initialized on the reception side when the start bit is detected.

Figure 10-32: Transfer Rate During Continuous Transmission



Where the 1-bit data length is FL, the stop bit length is FLstp, and base clock frequency is f_{XCLK}, the following expression is satisfied.

$$FLstp = FL + 2/f_{XCLK}$$

Therefore, the transfer rate during continuous transmission is:

$$\text{Transfer rate} = 11 \times FL + 2/f_{XCLK}$$

10.3 Clocked Serial Interfaces 0, 1 (CSI00, CSI01)

10.3.1 Features

- High-speed transfer: Maximum 8 Mbps (at 32 MHz or 24MHz system clock)
- 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
- Data phase and clock polarity can be changed
- Eight clock signals can be selected (7 master clocks and 1 slave clock)
- 3-wire type
  - SO0n : Serial transmitted data output
  - SI0n : Serial transmitted 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
- 3 buffers for each of the 2 CSI0n channels are provided on chip:
  - first stage transmission buffer (SOTBFn/SOTBFnL)
  - transmission buffer (SOTBn/SOTBnL)
  - reception buffers (SIRBn/SIRBnL)
- Single transfer mode and repeat transfer mode can be specified

Remark: n = 0 to 1

Caution: In HELIOS the automatic direction control for the port pins is not used. The user is responsible for the direction setting of clock- pin, SO and SI by programming the direction bits in the respective port SFR's. There is no automatic direction control in peripheral mode of the port pin's.

10.3.2 Configuration

CSIn is controlled via the clocked serial interface mode register (CSIMn) (n = 0, 1).
Transmission/reception of data is performed with reading SIOn register (n = 0, 1).

(1) Clocked serial interface mode registers (CSIM0, CSIM1)

The CSIMn register is an 8-bit register that specifies the operation of CSIn.

(2) Clocked serial interface clock selection registers (CSIC0, CSIC1)

The CSICn register is an 8-bit register that controls the CSI0n serial transfer operation.

(3) Serial I/O shift registers (SIO0, SIO1)

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 LSB shift registers (SIO0L, SIO1L)

The SIOnL register is an 8-bit shift register that converts parallel data into serial data.
The SIOnL 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, SIRB1)

The SIRBn register is a 16-bit buffer register that stores received data.

(6) Clocked serial interface LSB reception buffer registers (SIRBL0, SIRBL1)

The SIRBnL register is an 8-bit buffer register that stores received data.

(7) Clocked serial interface transmission buffer registers (SOTB0, SOTB1)

The SOTBn register is a 16-bit buffer register that stores transmitted data.

(8) Clocked serial interface LSB transmission buffer registers (SOTB0L, SOTB1L)

The SOTBnL register is an 8-bit buffer register that stores transmitted data.

(9) Clocked serial interface initial transmission buffer registers (SOTBF, SOTBF1)

The SOTBFn register is a 16-bit buffer register that stores the initial transmitted data in the repeat transfer mode.

(10) Clocked serial interface LSB initial transmission buffer registers (SOTBF0L, SOTBF1L)

The SOTBFnL register is an 8-bit buffer register that stores initial transmitted data in the repeat transfer mode.

(11) Selector

The selector selects the serial clock to be used.

(12) 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.

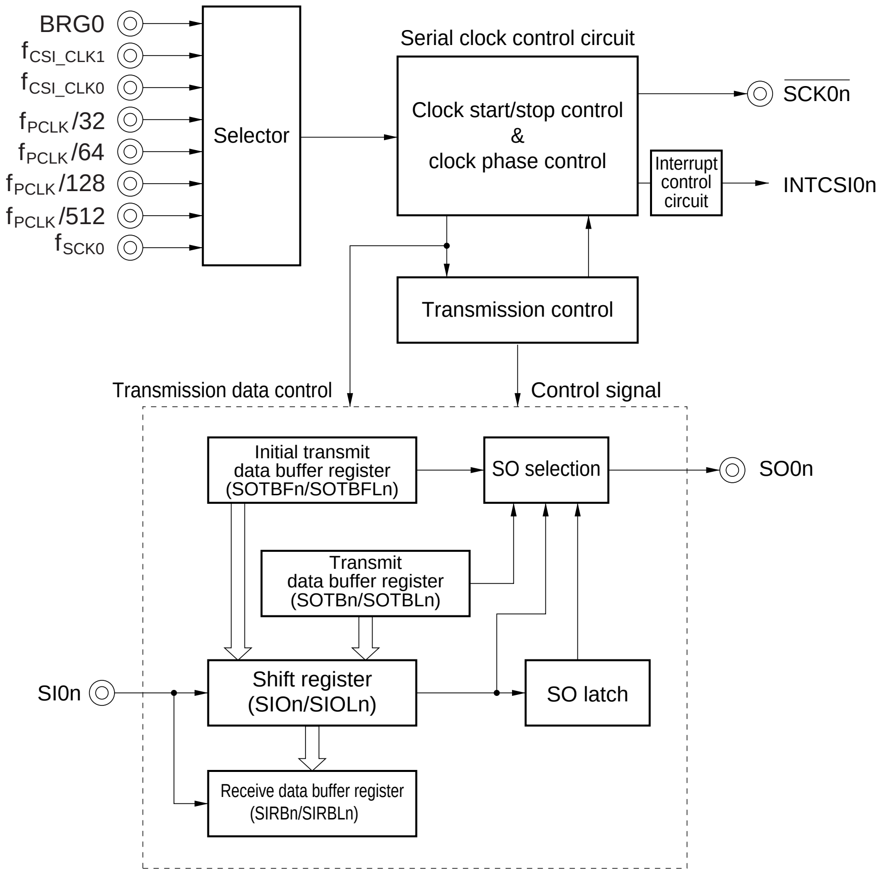
(13) Serial clock counter

Counts the serial clock output or input during transmission/reception operation, and checks whether 8-bit data transmission/reception has been performed.

(14) Interrupt control circuit

Controls the interrupt request timing.

Figure 10-33: Block Diagram of Clocked Serial Interfaces


BRG0
f_{CSI_CLK1}
f_{CSI_CLK0}
$f_{PCLK}/32$
$f_{PCLK}/64$
$f_{PCLK}/128$
$f_{PCLK}/512$
f_{SCK0}
Selector
Serial clock control circuit
Clock start/stop control & clock phase control
$\overline{\text{SCK0n}}$
Interrupt control circuit
INTCSI0n
Transmission control
Transmission data control
Control signal
Initial transmit data buffer register (SOTBFn/SOTBFLn)
SO selection
SO0n
Transmit data buffer register (SOTBn/SOTBLn)
SI0n
Shift register (SIOn/SIOLn)
SO latch
Receive data buffer register (SIRBn/SIRBLn)

Remark: n = 0, 1

10.3.3 Control registers

(1) Clocked serial interface mode registers 0, 1 (CSIM0, CSIM1)

The CSIMn register controls the CSI0n operation (n = 0, 1).

These registers can be read/written in 8-bit or 1-bit units (however, bit 1 is read-only).

Figure 10-34: Clocked Serial Interface Mode Registers (CSIM0, CSIM1)

	7	6	5	4	3	2	1	0	Address	Initial value
CSIM0	CSIE0	TRMD0	CCL0	DIRC00	CSIT0	AUTO0	0	CSOT0	FFFF FD00H	00H
CSIM1	CSIE1	TRMD1	CCL1	DIRC01	CSIT1	AUTO1	0	CSOT1	FFFF FD10H	00H

Bit Position	Bit Name	Function
7	CSIEn	Enables/disables CSIn operation. 0: Disable CSIn operation. 1: Enable CSIn operation. The internal CSIn circuit can be reset asynchronously by setting the CSIEn bit to 0. For the $\overline{\text{SCK0n}}$ and SO0n pin output status when the CSIEn bit = 0, refer to 10.3.5 ”Output pins” on page 301
6	TRMDn	Specifies transmission/reception mode. 0: Receive-only mode 1: Transmission/reception mode When the TRMDn 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 TRMDn bit = 1, transmission/reception is started by writing data to the SOTBn register.
5	CCLn	Specifies data length. 0: 8 bits 1: 16 bits
4	DIRC0n	Specifies transfer direction mode (MSB/LSB). 0: First bit of transfer data is MSB 1: First bit of transfer data is LSB
3	CSITn	Controls delay of interrupt request signal. 0: No delay 1: Delay mode (interrupt request signal is delayed 1/2 cycle). **Caution: The delay mode (CSITn bit = 1) is effective only in the master mode (CKS2 to CSK0 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	AUTOn	Specifies single transfer mode or repeat transfer mode. 0: Single transfer mode 1: Repeat transfer mode
0	CSOTn	Flag indicating transfer status. 0: Idle status 1: Transfer execution status **Caution: The CSOTn bit is cleared (0) by writing 0 to the CSIEn bit.**

Remark: n = 0 to 1

Overwriting the TRMDn, CCLn, DIRC0n, CSITn, and AUTOn bits of the CSIMn register can be done only when the CSOTn bit = 0. If these bits are overwritten at any other time, the operation cannot be guaranteed.

(2) Clocked serial interface clock selection registers 0, 1 (CSICn)

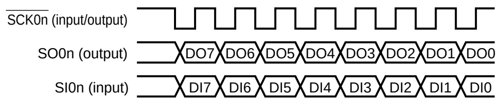
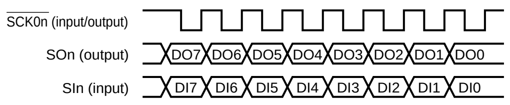
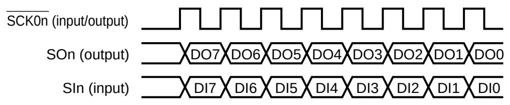
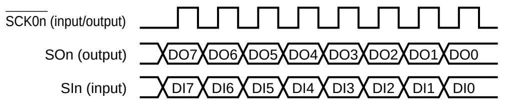
The CSICn register is an 8-bit register that controls the CSIn transfer operation (n = 0, 1).

This register can be read/written in 8-bit or 1-bit units.

Figure 10-35: Clocked Serial Interface Clock Selection Registers (CSIC0, CSIC1) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
CSIC0	0	0	0	CKP0	DAP0	CKS20	CKS10	CKS00	FFFF FD01H	00H
CSIC1	0	0	0	CKP1	DAP1	CKS21	CKS11	CKS01	FFFF FD11H	00H

Bit Position	Bit Name	Function
4, 3	CKPn, DAPn	Specifies operation mode: CKPn: clock phase selection bit DAPn: data phase selection bit

CKPn	DAPn	Operation Mode
0	0	 SCK0n (input/output) SO0n (output) SI0n (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
0	1	 SCK0n (input/output) SOn (output) SIn (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
1	0	 SCK0n (input/output) SOn (output) SIn (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
1	1	 SCK0n (input/output) SOn (output) SIn (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

Remark: n = 0, 1

Figure 10-35: Clocked Serial Interface Clock Selection Registers (CSIC0, CSIC1) (2/2)

Bit Position	Bit Name	Function				
2 to 0	CKS2n to CKS0n	Specifies input clock for CSI00				
		CKS2n	CKS1n	CKS0n	Input Clock	Mode
		0	0	0	from Baud Rate Generator BRG0	Master mode
		0	0	1	from PLL divider CSI_CLK1	Master mode
		0	1	0	from PLL divider CSI_CLK0	Master mode
		0	1	1	PCLK/32	Master mode
		1	0	0	PCLK/64	Master mode
		1	0	1	PCLK/128	Master mode
		1	1	0	PCLK/512	Master mode
		1	1	1	External clock ($\overline{SCK00}$)	Slave mode

Caution: The CSICn register can be overwritten only when the CSIE bit of the CSIM0n register = 0.

Notes: 1. PLCK is the peripheral clock and is connected to the CPU clock f_X.

2. CSI_CLK0 and CSI_CLK1 are set through the PCLCNT register (see Chapter 6 ”Clock Generator” on page 145).

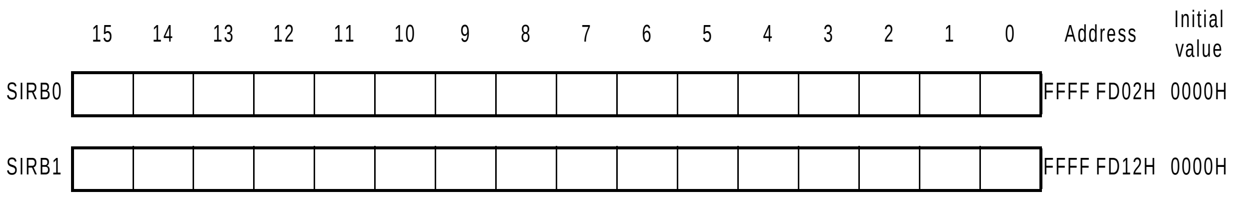
(3) Clocked serial interface reception buffer registers (SIRB0, SIRB1)

The SIRBn register is a 16-bit buffer register that stores received data.
When the receive-only mode is set (TRMDn bit of CSIMn register = 0, n = 0,1), 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 CSIEn bit of the CSIMn register.

Figure 10-36: Clocked Serial Interface Reception Buffer Registers (SIRB0, SIRB1)



Bit Position	Function
15 to 0	Store received data.

Cautions: 1. Read the SIRBn register only when the 16-bit data length has been set (CCLn bit of CSIMn register = 1).

2. When the single transfer mode has been set (AUTOn bit of CSIMn register = 0), perform read operation only in the idle state (CSOTn bit of CSIMn register = 0). If the SIRBn register is read during data transfer, the data cannot be guaranteed.

(4) Clocked serial interface LSB reception buffer registers (SIRBL0, SIRBL1)

The SIRBLn register is an 8-bit buffer register that stores received data (n = 0, 1).
When the receive-only mode is set (TRMDn bit of CSIMn register = 0), the reception operation is started by reading data from the SIRBLn register.
In addition to reset input, these registers can also be initialized by clearing (0) the CSIEn bit of the CSIMn register.
The SIRBLn register is the same as the lower bytes of the SIRBn register.

These registers are read-only, in 8-bit units.

Figure 10-37: Clocked Serial Interface Reception Buffer Registers (SIRBL0, SIRBL1)

	7	6	5	4	3	2	1	0	Address	Initial value
SIRBL0									FFFF FD02H	00H
SIRBL1									FFFF FD12H	00H

Bit Position	Function
7 to 0	Stores 8 LSB bits of the received data.

Cautions: 1. Read the SIRBLn register only when the 8-bit data length has been set (CCLn bit of CSIMn register = 0).

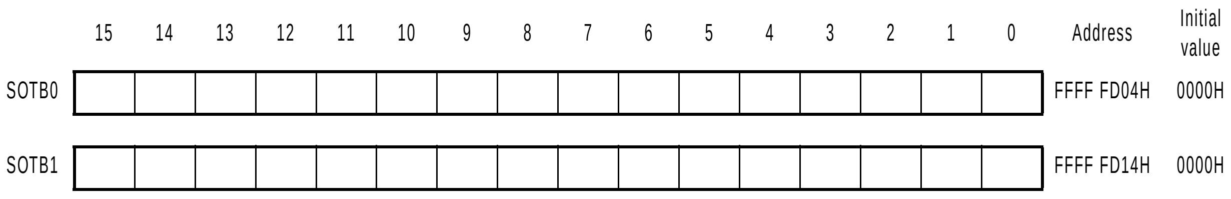
2. When the single transfer mode is set (AUTOn bit of CSIMn register = 0), perform read operation only in the idle state (CSOTn bit of CSIMn register = 0). If the SIRBLn register is read during data transfer, the data cannot be guaranteed.

(5) Clocked serial interface LSB transmission buffer registers (SOTB0, SOTB1)

The SOTBn register is a 16-bit buffer register that stores transmitted data (n = 0, 1).
When the transmission/reception mode is set (TRMDn bit of CSIMn register = 1), the transmission operation is started by writing data to the SOTBnL register.

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

Figure 10-38: Clocked Serial Interface Transmission Buffer Registers (SOTB0, SOTB1)



Bit Position	Function
15 to 0	Store transmitted data.

Cautions: 1. Access the SOTBn register only when the 16-bit data length is set (CCLn bit of CSIMn register = 1).

2. When the single transfer mode is set (AUTOn bit of CSIMn register = 0), perform access only in the idle state (CSOTn bit of CSIMn register = 0). If the SOTBn register is accessed during data transfer, the data cannot be guaranteed.

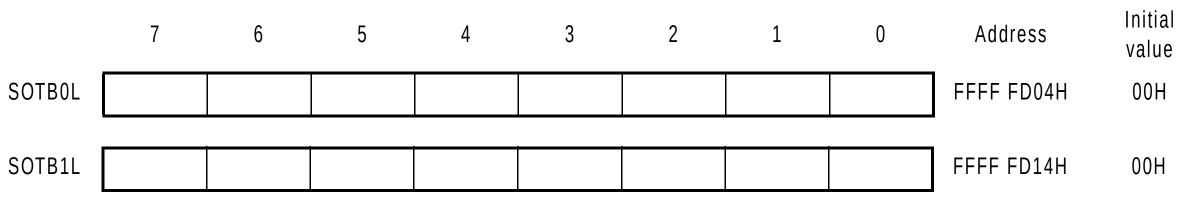
(6) Clocked serial interface LSB transmission buffer registers (SOTB0L, SOTB1L)

The SOTBnL register is an 8-bit buffer register that stores transmitted data (n = 0, 1).
When the transmission/reception mode is set (TRMDn bit of CSIMn register = 1), the transmission operation is started by writing data to the SOTBnL register.

These registers can be read/written in 8-bit units.

The SOTBnL register is the same as the lower bytes of the SOTBn register.

Figure 10-39: Clocked Serial Interface Transmission Buffer Registers (SOTBnL)



Bit Position	Function
7 to 0	Store transmitted data.

Cautions: 1. Access the SOTBnL register only when the 8-bit data length has been set (CCLn bit of CSIMn register = 0).

2. When the single transfer mode is set (AUTOn bit of CSIMn register = 0), perform access only in the idle state (CSOTn bit of CSIMn register = 0). If the SOTBnL register is accessed during data transfer, the data cannot be guaranteed.

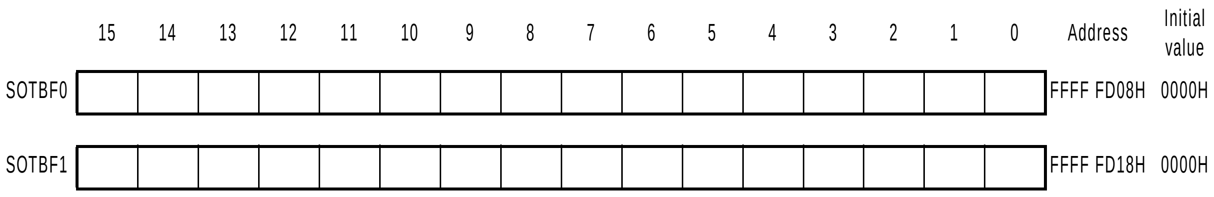
3. When SOTBnL is written by 8 bit access, then undefined data are written in the higher bits 15 to 8 of SOTBn.

(7) Clocked serial interface initial transmission buffer registers (SOTBF0, SOTBF1)

The SOTBFn register is a 16-bit buffer register that stores initial transmission data in the repeat transfer mode (n = 0, 1).
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 10-40: Clocked Serial Interface Initial Transmission Buffer Registers 0, 1 (SOTBFn)



Bit Position	Function
15 to 0	Stores initial transmission data in repeat transfer mode.

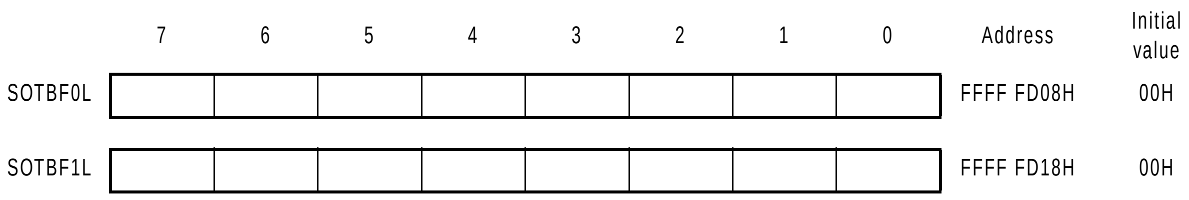
Caution: Access the SOTBFn register only when the 16-bit data length has been set (CCLn bit of CSIMn register = 1), and only in the idle state (CSOTn bit of CSIMn register = 0). If the SOTBFn register is accessed during data transfer, the data cannot be guaranteed.

(8) Clocked serial interface LSB initial transmission buffer registers (SOTBF0L, SOTBF1L)

The SOTBFnL register is an 8-bit buffer register that stores initial transmission data in the repeat transfer mode (n = 0, 1).
The transmission operation is not started even if data is written to the SOTBFnL register.
The SOTBFnL register is the same as the lower bytes of the SOTBFn register.

These registers can be read/written in 8-bit units.

Figure 10-41: Clocked Serial Interface Initial Transmission Buffer Registers (SOTBFnL)



Bit Position	Function
7 to 0	Store initial transmission data in repeat transfer mode.

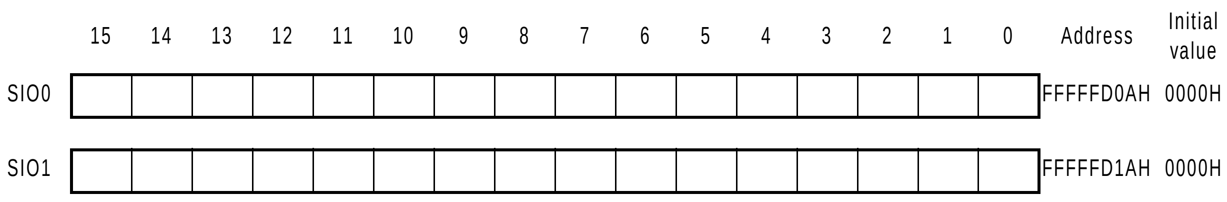
Cautions: 1. Access the SOTBFnL register only when the 8-bit data length has been set (CCLn bit of CSIM0 register = 0), and only in the idle state (CSOTn bit of CSIMn register = 0). If the SOTBFnL register is accessed during data transfer, the data cannot be guaranteed.

2. When SOTBFnL is written by 8 bit access, then undefined data are written in the higher bits 15 to 8 of SOTBFn.

(9) Serial I/O shift registers (SIO0, SIO1)

The SIOn register is a 16-bit shift register that converts parallel data into serial data (n = 0, 1).
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 CSIEn bit of the CSIMn register.

Figure 10-42: Serial I/O Shift Registers (SIO0, SIO1)



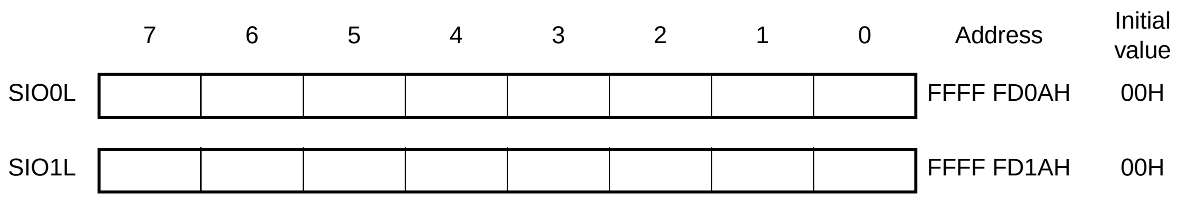
Bit Position	Function
15-0	Data is shifted in (reception) or shifted out (transmission) from the MSB or LSB side.

Caution: Access the SIOn register only when the 16-bit data length has been set (CCLn bit of CSIMn register = 1), and only in the idle state (CSOTn bit of CSIMn register = 0). If the SIOn register is accessed during data transfer, the data cannot be guaranteed.

(10) Serial I/O LSB shift registers (SIO0L, SIO1L)

The SIOnL register is an 8-bit shift register that converts parallel data into serial data (n = 0, 1).
The transfer operation is not started even if the SIOnL 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 CSIEn bit of the CSIMn register.
The SIOnL register is the same as the lower bytes of the SIOn register.

Figure 10-43: Serial I/O Shift Registers (SIO0L, SIO1L)



Bit Position	Function
7 to 0	Data is shifted in (reception) or shifted out (transmission) from the MSB or LSB side.

Caution: Access the SIOnL register only when the 8-bit data length has been set (CCLn bit of CSIMn register = 0), and only in the idle state (CSOTn bit of CSIMn register = 0). If the SIOnL register is accessed during data transfer, the data cannot be guaranteed.

10.3.4 Operation

(1) Single transfer mode

(a) Usage

In the receive-only mode (TRMDn bit of CSIMn register = 0), transfer is started by reading^Note 1^ the received data buffer register (SIRBn/SIRBnL) (n = 0, 1).
In the transmission/reception mode (TRMDn bit of CSIMn register = 1), transfer is started by writing^Note 2^ to the transmit data buffer register (SOTBn/SOTBnL).
In the slave mode, the operation must be enabled beforehand (CSIEn bit of CSIMn register = 1).
When transfer is started, the value of the CSOTn bit of the CSIMn register becomes 1 (transmission execution status).
Upon transfer completion, the transmission/reception completion interrupt (INTCSI0n) is set (1), and the CSOTn bit is cleared (0). The next data transfer request is then waited for.

Notes: 1. When the 16-bit data length (CCLn bit of CSIMn register = 1) has been set, read the SIRBn register. When the 8-bit data length (CCLn bit of CSIMn register = 0) has been set, read the SIRBnL register.

2. When the 16-bit data length (CCLn bit of CSIMn register = 1) has been set, write to the SOTBn register. When the 8-bit data length (CCLn bit of CSIMn register = 0) has been set, write to the SOTBnL register.

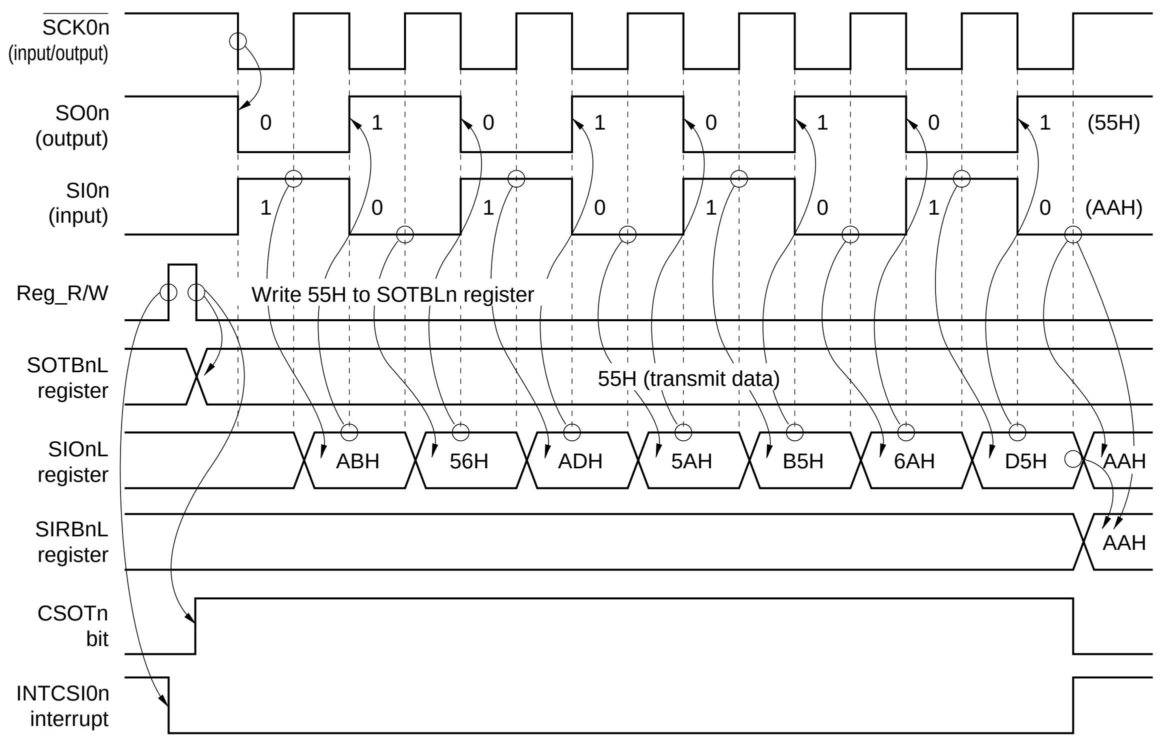
Caution: When the CSOTn bit of the CSIMn register = 1, do not manipulate the CSIn register.

As an example of the bidirectional communication the following example shows the sending of 55H and the receiving of AAH via the CSI. The following two timing charts shows the communication with different settings of the Data Phase Selection-bit (DAPn)

For detailed information of the DAPn and CKPn bit please refer to **10.4.4 Operation (b)”Clock phase selection” on page 313**.

Figure 10-44: Timing Chart in Single Transfer Mode (DAPn = 0) (1/2)

(a) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKPn bit = 0, DAPn bit = 0

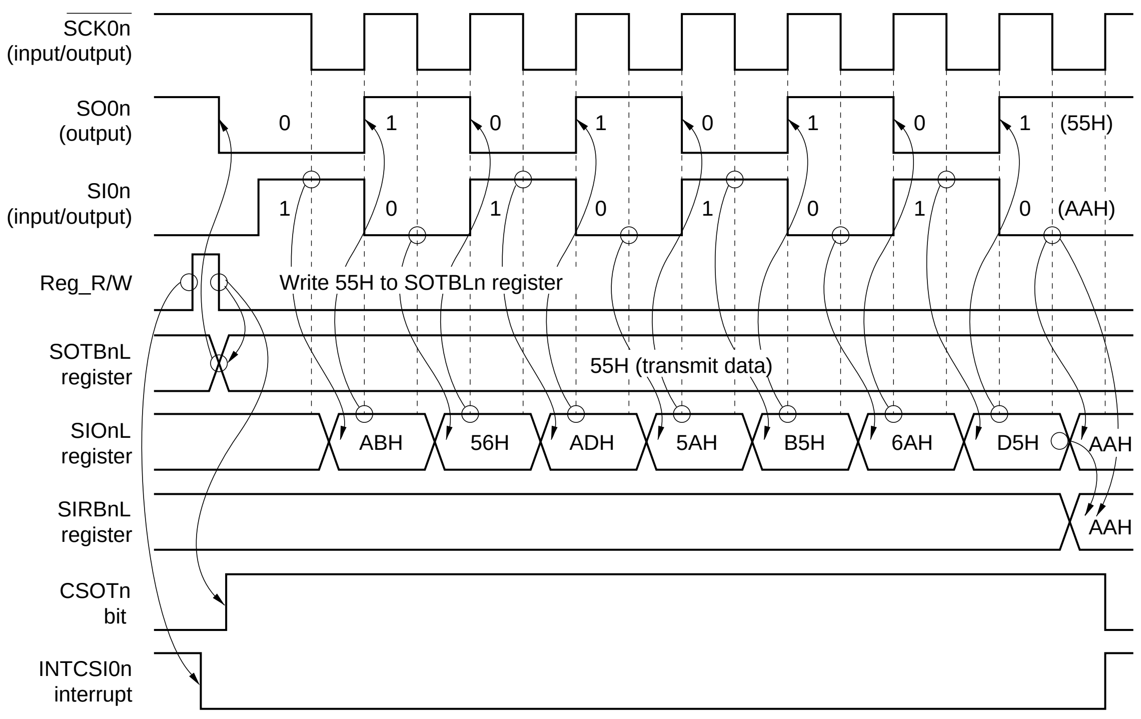


Remarks: 1. n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIRBn/SIRBnL) read or transmit data buffer register (SOTBn/SOTBnL) write was performed.

Figure 10-44: Timing Chart in Single Transfer Mode (DAPn = 1) (2/2)

(b) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKPn bit = 0, DAPn bit = 1



Remarks: 1. n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIRBn/SIRBnL) read or transmit data buffer register (SOTBn/SOTBnL) write was performed.

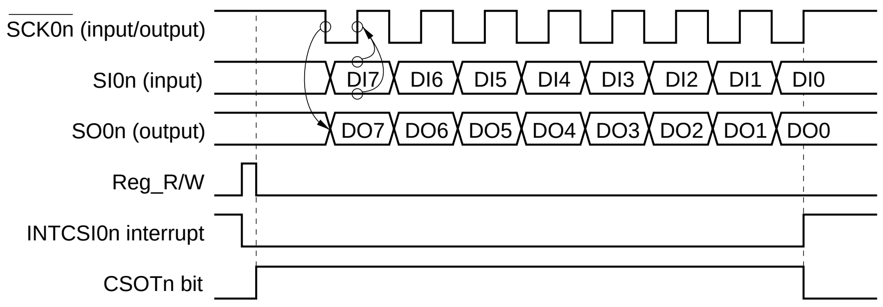
(b) Clock phase selection

The following shows the timing when changing the conditions for clock phase selection (CKPn bit of CSICn register) and data phase selection (DAPn bit of CSICn register) under the following conditions.

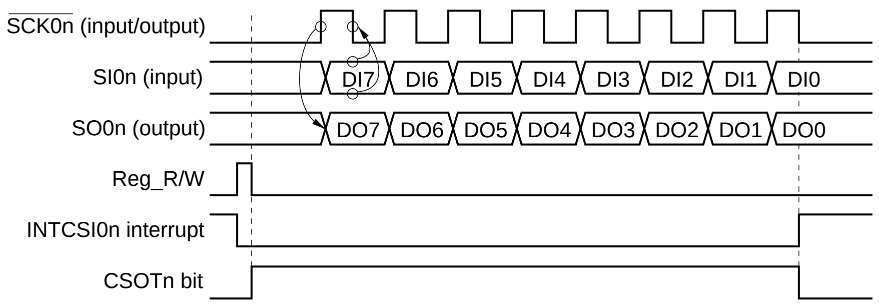
- Data length = 8 bits (CCLn bit of CSIMn register = 0)
- First bit of transfer data = MSB (DIRC0n bit of CSIMn register = 0)
- No interrupt request signal delay control (CSITn bit of CSIMn register = 0)

Figure 10-45: Timing Chart According to Clock Phase Selection (1/2)

(a) When CKPn bit = 0, DAPn bit = 0



(b) When CKPn bit = 1, DAPn bit = 0

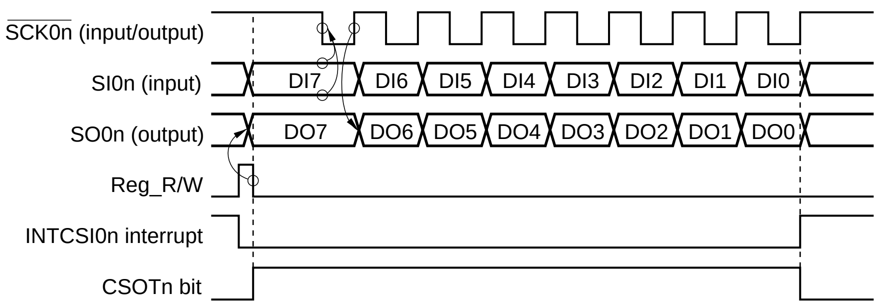


Remarks: 1. n = 0, 1

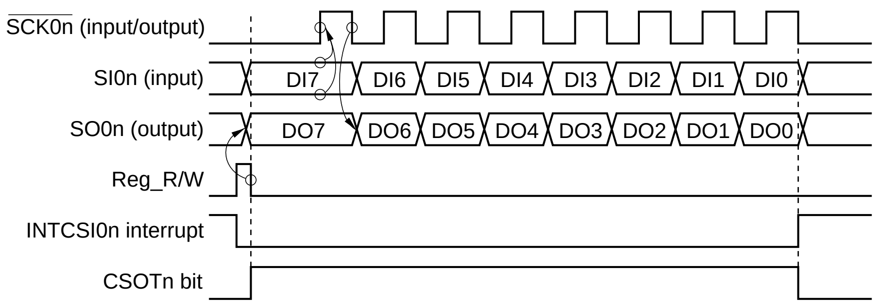
2. Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIRBn/SIRBnL) read or transmit data buffer register (SOTBn/SOTBnL) write was performed.

Figure 10-45: Timing Chart According to Clock Phase Selection (2/2)

(c) When CKPn bit = 0, DAPn bit = 1



(d) When CKPn bit = 1, DAPn bit = 1



Remarks: 1. n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIRBn/SIRBnL) read or transmit data buffer register (SOTBn/SOTBnL) write was performed. Transmission/reception completion interrupt request signals (INTCSI00, INTCSI01)

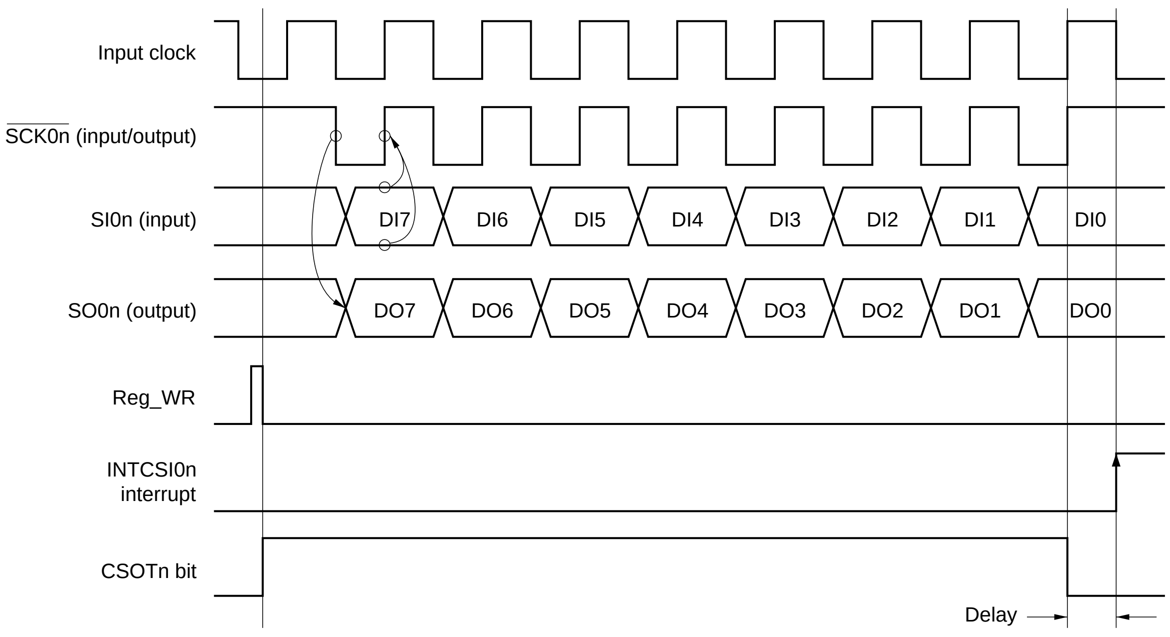
(c) Transmission/reception completion interrupt request signals (INTCSI00, INTCSI01)

INTCSI0n is set (1) upon completion of data transmission/reception.

Caution: The delay mode (CSITn 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 10-46: Timing Chart of Interrupt Request Signal Output in Delay Mode (1/2)

(a) When CKPn bit = 0, DAPn bit = 0

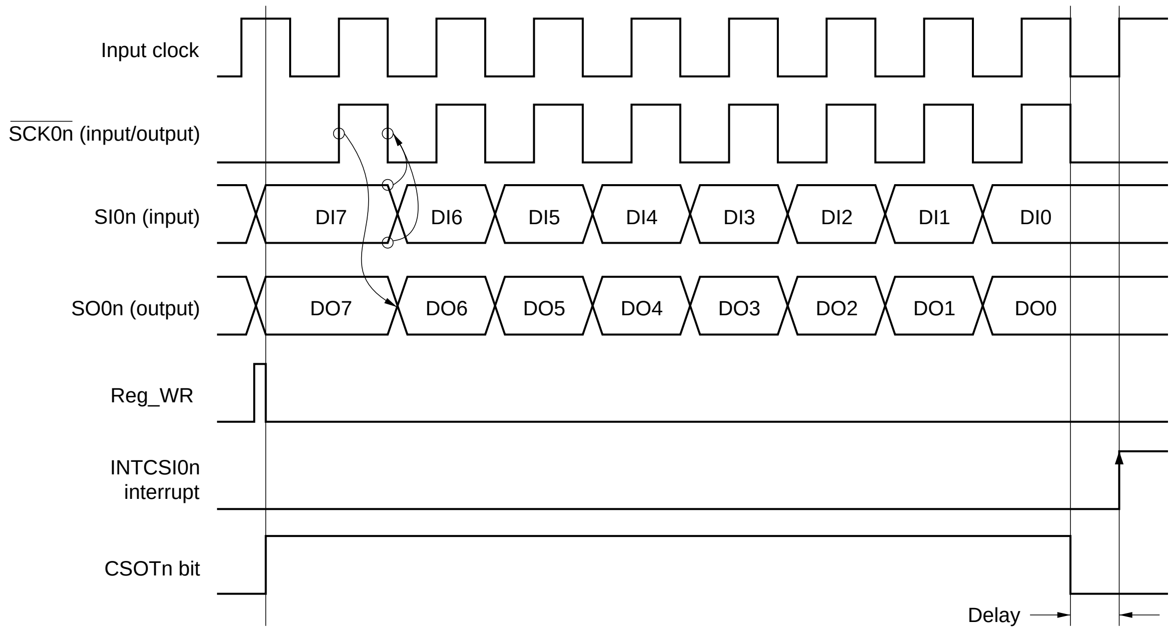


Remarks: 1. n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIRBn/SIRBnL) read or transmit data buffer register (SOTBn/SOTBnL) write was performed.

Figure 10-46: Timing Chart of Interrupt Request Signal Output in Delay Mode (2/2)

(b) When CKPn bit = 1, DAPn bit = 1



Remarks: 1. n = 0, 1

2. Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIRBn/SIRBnL) read or transmit data buffer register (SOTBn/SOTBnL) write was performed.

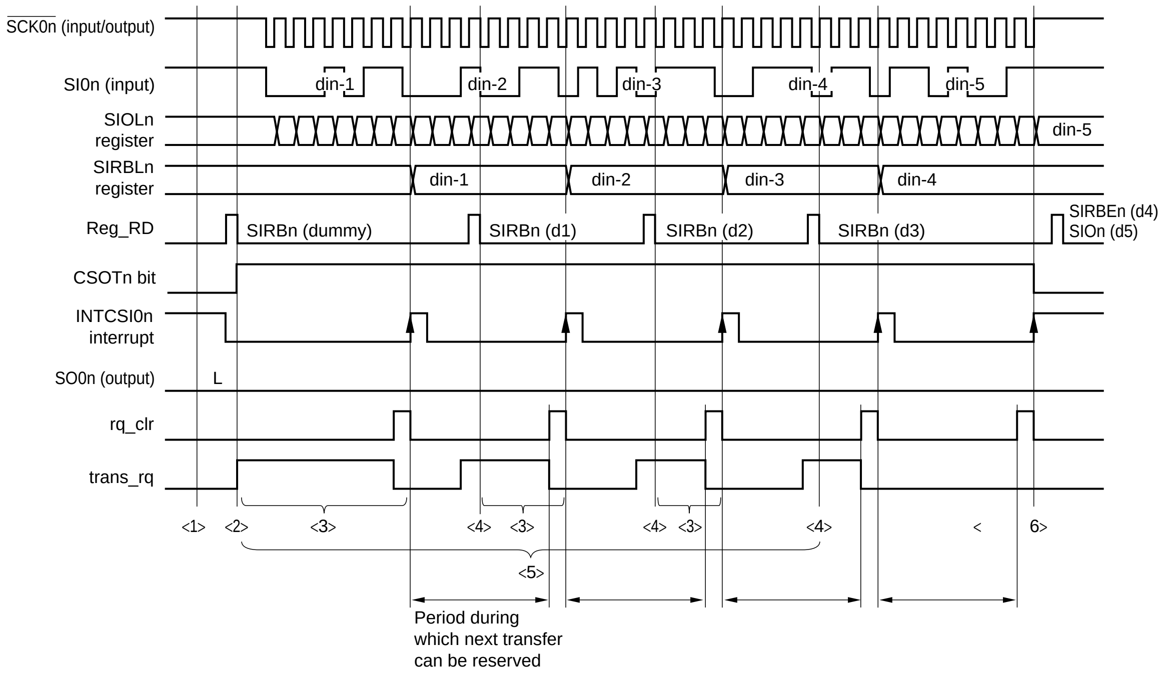
(2) Repeat transfer mode

(a) Usage (receive-only)

<1> Set the repeat transfer mode (AUTOn bit of CSIMn register = 1) and the receive-only mode (TRMDn 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, received 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 SIRBE0n register, and the n-th (last) data is loaded by reading the SIOn register.

Figure 10-47: Repeat Transfer (Receive-Only) Timing Chart



Remarks: **1.** n = 0, 1

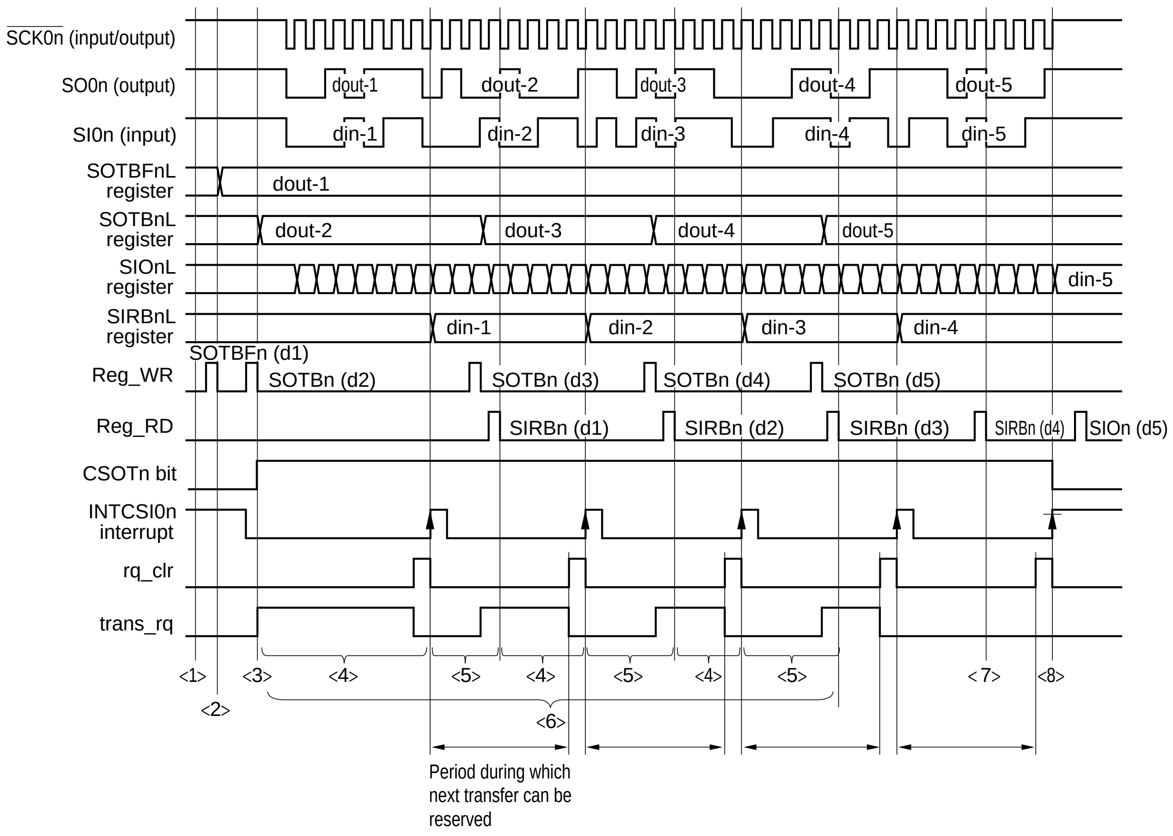
2. Reg_RD:Internal signal. This signal indicates that the received data buffer register (SIRBn/SIRBnL) has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

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 (AUTOn bit of CSIMn register = 1) and the transmission/reception mode (TRMDn 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 received 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 received data.
<8> Following the last transmission/reception completion interrupt request (INTCSI0n), read the SIOn register to load the n-th (last) received data.

Figure 10-48: Repeat Transfer (Transmission/Reception) Timing Chart



Remarks: 1. n = 0, 1

2. Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBnL) has been written.
Reg_RD:Internal signal. This signal indicates that the received data buffer register (SIRBn/SIRBnL) has been read.
rq_clr: Internal signal. Transfer request clear signal.
trans_rq: Internal signal. Transfer request signal.

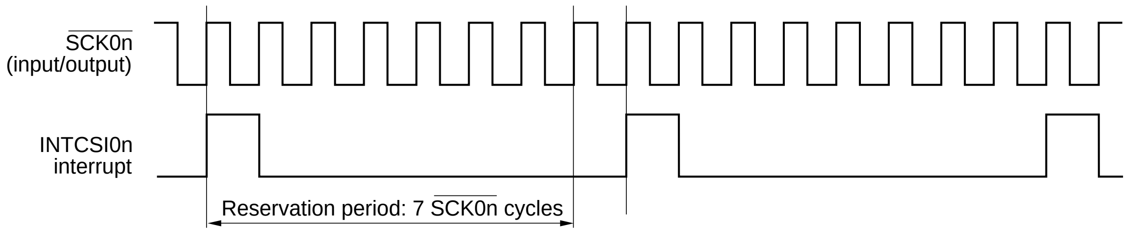
In the case of the repeat transfer mode, two transfer requests are set at the start of the first transfer. Following the transmission/reception completion interrupt request (INTCSI0n), transfer is continued if the SOTBn register can be written within the next transfer reservation period. If the SOTBn register cannot be written, transfer ends and the SIRBn register does not received the new value of the SIOn register. The last received 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 10-49.

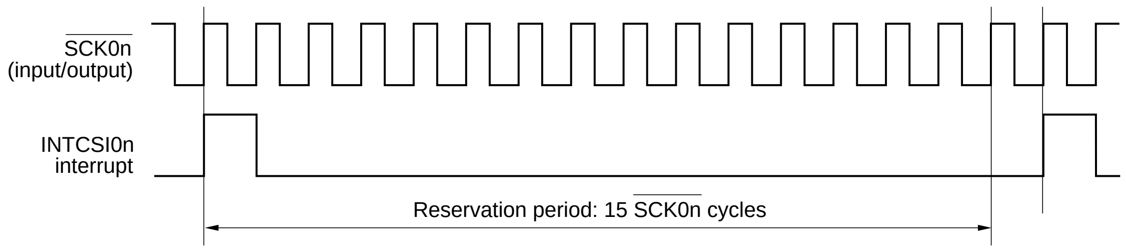
Figure 10-49: Timing Chart of Next Transfer Reservation Period (1/2)

(a) When data length: 8 bits, operation mode: CKPn bit = 0, DAPn bit = 0



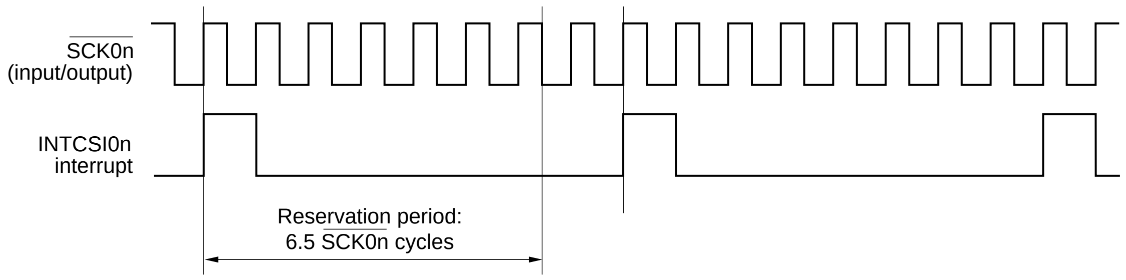
$\overline{\text{SCK0n}}$ (input/output)
INTCSI0n interrupt
Reservation period: 7 $\overline{\text{SCK0n}}$ cycles

(b) When data length: 16 bits, operation mode: CKPn bit = 0, DAPn bit = 0



$\overline{\text{SCK0n}}$ (input/output)
INTCSI0n interrupt
Reservation period: 15 $\overline{\text{SCK0n}}$ cycles

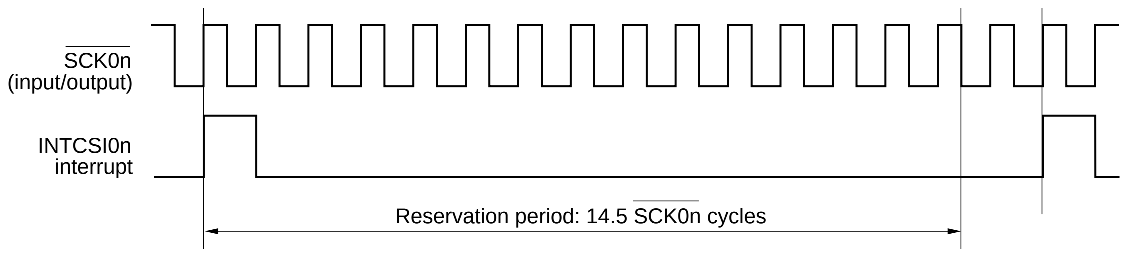
(c) When data length: 8 bits, operation mode: CKPn bit = 0, DAPn bit = 1



$\overline{\text{SCK0n}}$ (input/output)
INTCSI0n interrupt
Reservation period: 6.5 $\overline{\text{SCK0n}}$ cycles

Figure 10-49: Timing Chart of Next Transfer Reservation Period (2/2)

(d) When data length: 16 bits, operation mode: CKPn bit = 0, DAPn bit = 1



$\overline{\text{SCK0n}}$ (input/output)
INTCSI0n interrupt
Reservation period: 14.5 $\overline{\text{SCK0n}}$ cycles

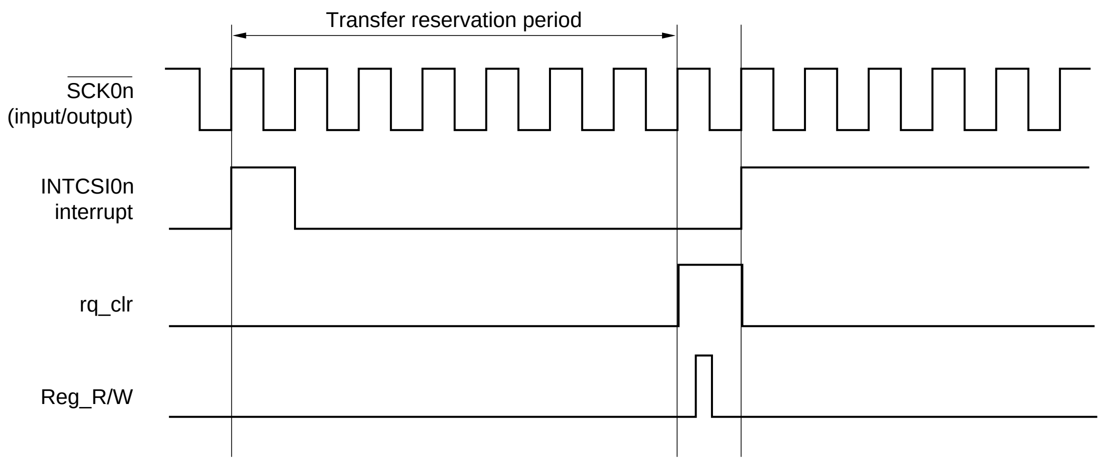
Remark: n = 0, 1

(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 10-50: Transfer Request Clear and Register Access Contention



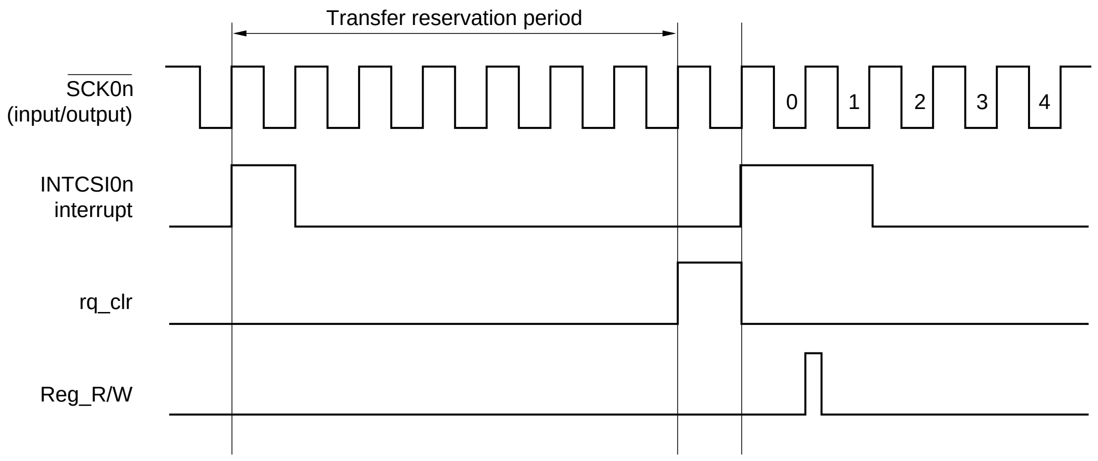
Transfer reservation period
$\overline{\text{SCK0n}}$ (input/output)
INTCSI0n interrupt
rq_clr
Reg_R/W

Remarks: 1. n = 0, 1

2. rq_clr: Internal signal. Transfer request clear signal.
Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBnL) 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 results in a transfer error (refer to Figure 10-51).
In the transmission/reception mode, the value of the SOTBFn register is re-transmitted, and illegal data is sent.

Figure 10-51: Interrupt Request and Register Access Contention



Remarks: 1. n = 0, 1

2. rq_clr: Internal signal. Transfer request clear signal.
Reg_WR:Internal signal. This signal indicates that the transmit data buffer register (SOTBn/SOTBnL) has been written.

10.3.5 Output pins

(1) $\overline{\text{SCK0n}}$ pin

When the CSIn operation is disabled (CSIEn bit of CSIMn register = 0), the $\overline{\text{SCK0n}}$ pin output status is as follows (n = 0 to 2).

CKPn	CKS2n	CKS1n	CKS0n	$\overline{\text{SCK0n}}$ Pin Output
0	Don’t care	Don’t care	Don’t care	Fixed to high level
1	1	1	1	Fixed to high level
1	Other than above			Fixed to low level

Remarks: 1. n = 0, 1

2. When any of bits CKPn and CKS2 to CKS0 of the CSICn register is overwritten, the $\overline{\text{SCK0n}}$ pin output changes.

(2) SO0n pin

When the CSIn operation is disabled (CSIEn bit of CSIMn register = 0), the SO0n pin output status is as follows (n = 0 to 2).

TRMDn	DAPn	AUTOn	CCLn	DIRC0n	SO0n Pin Output
0	X	X	X	X	Fixed at low level
1	0	X	X	X	SOn latch value (low level)
1	1	0	0	0	SOTBn7 value
1	1	0	0	1	SOTBn0 value
1	1	0	1	0	SOTBn15 value
1	1	0	1	1	SOTBn0 value
1	1	1	0	0	SOTBFn7 value
1	1	1	0	1	SOTBFn0 value
1	1	1	1	0	SOTBFn15 value
1	1	1	1	1	SOTBFn0 value

Remarks: 1. When any of bits TRMDn, CCLn, DIRC0n, AUTOn, and CSICn of the CSIMn register or DAPn bit of the CSICn register is overwritten, the SO0n pin output changes.

2. SOTBnm: Bit m of SOTBn register (m = 0, 7, 15)

3. SOTBFnm: Bit m of SOTBFn register (m = 0, 7, 15)

4. n = 0, 1

5. X = Don’t care

10.4 Clocked Serial Interface 10 (CSI10)

10.4.1 Features

- High-speed transfer: Maximum 8 Mbps (at 32 MHz and 24MHz system clock)
- Master mode or slave mode can be selected
- Transmission data length: 8 bits
- Transfer data direction can be switched between MSB first and LSB first
- Data phase and clock polarity can be changed
- Eight clock signals can be selected (7 master clocks and 1 slave clock)
- 3-wire type
  - SO10 : Serial transmitted data output
  - SI10 : Serial transmitted data input
  - $\overline{\text{SCK10}}$: Serial clock input/output
- Interrupt sources: 1 type
  - Transmission/reception completion interrupt (INTCSI0n)
- Transmission/reception mode and reception-only mode can be specified
- 1 transmission data buffer

10.4.2 Configuration

CSI10 is controlled via the clocked serial interface mode register (CSIM10).
Transmission/reception of data is performed with reading SIO10 register.

(1) Clocked serial interface mode registers (CSIM10)

The CSIM10 register is an 8-bit register that specifies the operation of CSI10.

(2) Clocked serial interface clock selection registers (CSIC10)

The CSIC10 register is an 8-bit register that controls the CSI10 serial transfer operation.

(3) Serial I/O shift register (SIO10)

The SIO10 register is a 8-bit shift register that converts parallel data into serial data.
The SIO10 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) Clocked serial interface transmission buffer register (SOTB10)

The SOTB10 register is a 8-bit buffer register that stores transmitted data.

(5) Selector

The selector selects the serial clock to be used.

(6) Serial clock control circuit

Controls the serial clock supply to the shift register. Also controls the clock output to the $\overline{\text{SCK10}}$ pin when the internal clock is used.

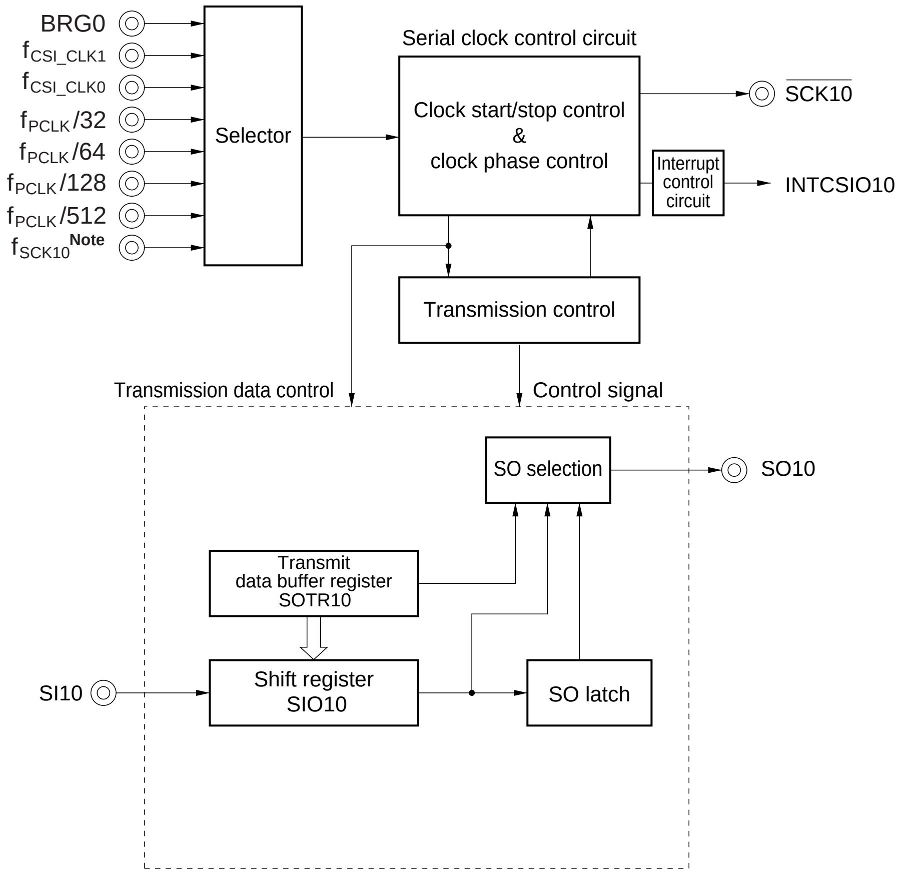
(7) Serial clock counter

Counts the serial clock output or input during transmission/reception operation, and checks whether 8-bit data transmission/reception has been performed.

(8) Interrupt control circuit

Controls the interrupt request timing.

Figure 10-52: Block Diagram of Clocked Serial Interface CSI10



Note: Please refer to the input frequency specification of CSI10.

10.4.3 Control registers

(1) Clocked serial interface mode register (CSIM10)

The **CSIM10** register controls the CSI10 operation.

This register can be read/written in 8-bit or 1-bit units (however, bit 5, 3, 2, 1 are read-only).

Figure 10-53: Clocked Serial Interface Mode Registers (CSIM10)

	7	6	5	4	3	2	1	0	Address	Initial value
CSIM10	CSIE10	TRMD10	0	DIRC10	0	0	0	CSOT10	FFFF FD20H	00H

Bit Position	Bit Name	Function
7	CSIE10	Enables/disables CSI10 operation. 0: Disable CSI10 operation. 1: Enable CSI10 operation. The internal CSI10 circuit can be reset asynchronously by setting the CSIE10 bit to 0. For the $\overline{\text{SCK10}}$ and SO10 pin output status when the CSIE10 bit = 0, refer to 10.4.5 ”Output pins” on page 315
6	TRMD10	Specifies transmission/reception mode. 0: Receive-only mode 1: Transmission/reception mode When the TRMD10 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 TRMD10 bit = 1, transmission/reception is started by writing data to the SOTBn register.
4	DIRC10	Specifies transfer direction mode (MSB/LSB). 0: First bit of transfer data is MSB 1: First bit of transfer data is LSB
0	CSOT10	Flag indicating transfer status. 0: Idle status 1: Transfer execution status **Caution: The CSOT10 bit is cleared (0) by writing 0 to the CSIE10 bit.**

Overwriting the TRMD10 and DIRC10 bits of the CSIM10 register can be done only when the CSOT10 bit = 0. If these bits are overwritten at any other time, the operation cannot be guaranteed.

(2) Clocked serial interface clock selection register (CSIC10)

The CSIC10 register is an 8-bit register that controls the CSI10 transfer operation.

This register can be read/written in 8-bit or 1-bit units.

Figure 10-54: Clocked Serial Interface Clock Selection Register (CSIC10) (1/2)

	7	6	5	4	3	2	1	0	Address	Initial value
CSIC10	0	0	0	CKP10	DAP10	CKS102	CKS101	CKS100	FFFF FD21H	00H

Bit Position	Bit Name	Function
4, 3	CKP10, DAP10	Specifies operation mode: CKP10: clock phase selection bit DAPn: data phase selection bit

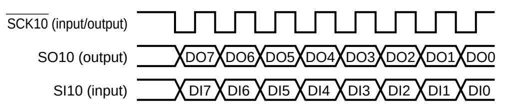
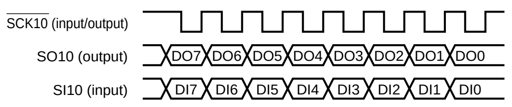
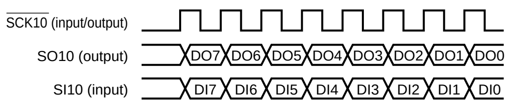
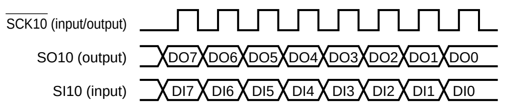
CKP10	DAP10	Operation Mode
0	0	 SCK10 (input/output) SO10 (output) SI10 (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
0	1	 SCK10 (input/output) SO10 (output) SI10 (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
1	0	 SCK10 (input/output) SO10 (output) SI10 (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0
1	1	 SCK10 (input/output) SO10 (output) SI10 (input) DO7 DO6 DO5 DO4 DO3 DO2 DO1 DO0 DI7 DI6 DI5 DI4 DI3 DI2 DI1 DI0

Figure 10-54: Clocked Serial Interface Clock Selection Register (CSIC10) (2/2)

Bit Position	Bit Name	Function
2 to 0	CKS102 to CKS100	Specifies input clock for CSI10

CKS102	CKS101	CKS100	Input Clock	Mode
0	0	0	from Baud Rate Generator BRG0	Master mode
0	0	1	from PLL divider CSI_CLK1	Master mode
0	1	0	from PLL divider CSI_CLK0	Master mode
0	1	1	PCLK/32	Master mode
1	0	0	PCLK/64	Master mode
1	0	1	PCLK/128	Master mode
1	1	0	PCLK/512	Master mode
1	1	1	External clock ($\overline{\text{SCK10}}$)	Slave mode

Caution: The CSIC10 register can be overwritten only when the CSIE10 bit of the CSIM10 register = 0.

Notes: 1. PLCK is the peripheral clock and is connected to the CPU clock f_X.

2. CSI_CLK0 and CSI_CLK1 are set through the PCLCNT register (see Chapter 6 ”Clock Generator” on page 145).

(3) Clocked serial interface LSB transmission buffer register (SOTB10)

The SOTB10 register is a 8-bit buffer register that stores transmitted data.
When the transmission/reception mode is set (TRMD10 bit of CSIM10 register = 1), the transmission operation is started by writing data to the SOTB10 register.

This register can be read/written in 8-bit units.

Figure 10-55: Clocked Serial Interface Transmission Buffer Register (SOTB10)

	7	6	5	4	3	2	1	0	Address	Initial value
SOTB10									FFFF FD24H	0000H

Bit Position	Function
7 to 0	Store transmitted data.

Caution: Perform access to this register only in the idle state (CSOT10 bit of CSIM10 register = 0). If the SOTB10 register is accessed during data transfer, the data cannot be guaranteed.

(4) Serial I/O shift register (SIO10)

The SIO10 register is a 8-bit shift register that converts parallel data into serial data.
The transfer operation is not started even if the SIO10 register is read.
This register is read-only, in 8-bit units.
In addition to reset input, this register can also be initialized by clearing (0) the CSIE10 bit of the CSIM10 register.

Figure 10-56: Serial I/O Shift Register (SIO10)

	7	6	5	4	3	2	1	0	Address	Initial value
SIO10									FFFFFD22H	0000H

Bit Position	Function
7-0	Data is shifted in (reception) or shifted out (transmission) from the MSB or LSB side.

Caution: Access the SIO10 register only in the idle state (CSOT10 bit of CSIM10 register = 0). If the SIO10 register is accessed during data transfer, the data cannot be guaranteed.

10.4.4 Operation

Single transfer mode is the only transfer mode available for CSI10.

(a) Usage

In the receive-only mode (TRMD10 bit of CSIM10 register = 0), transfer is started by reading the received data buffer register (SIO10).
In the transmission/reception mode (TRMD10 bit of CSIM10 register = 1), transfer is started by writing to the transmit data buffer register (SOTB10).
In the slave mode, the operation must be enabled beforehand (CSIE10 bit of CSIM10 register = 1).
When transfer is started, the value of the CSOT10 bit of the CSIM10 register becomes 1 (transmission execution status).
Upon transfer completion, the transmission/reception completion interrupt (INTCSI10) is set (1), and the CSOT10 bit is cleared (0). The next data transfer request is then waited for.

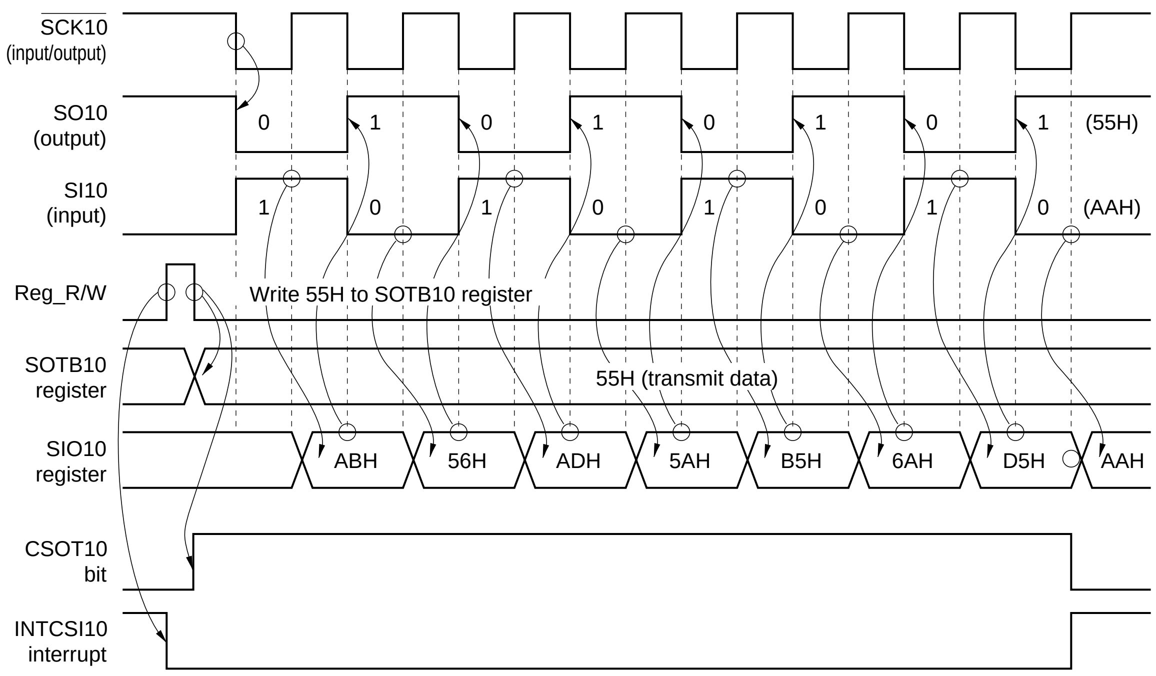
Caution: When the CSOT10 bit of the CSIM10 register = 1, do not manipulate the CSI10 register.

As an example of the bidirectional communication the following example shows the sending of 55H and the receiving of AAH via the CSI10. The following two timing charts shows the communication with different settings of the Data Phase Selection-bit (DAP10)

For detailed information of the DAP10 and CKP10 bit please refer to **10.4.4 Operation (b)”Clock phase selection” on page 313**.

Figure 10-57: Timing Chart in Single Transfer Mode (DAP10 = 0) (1/2)

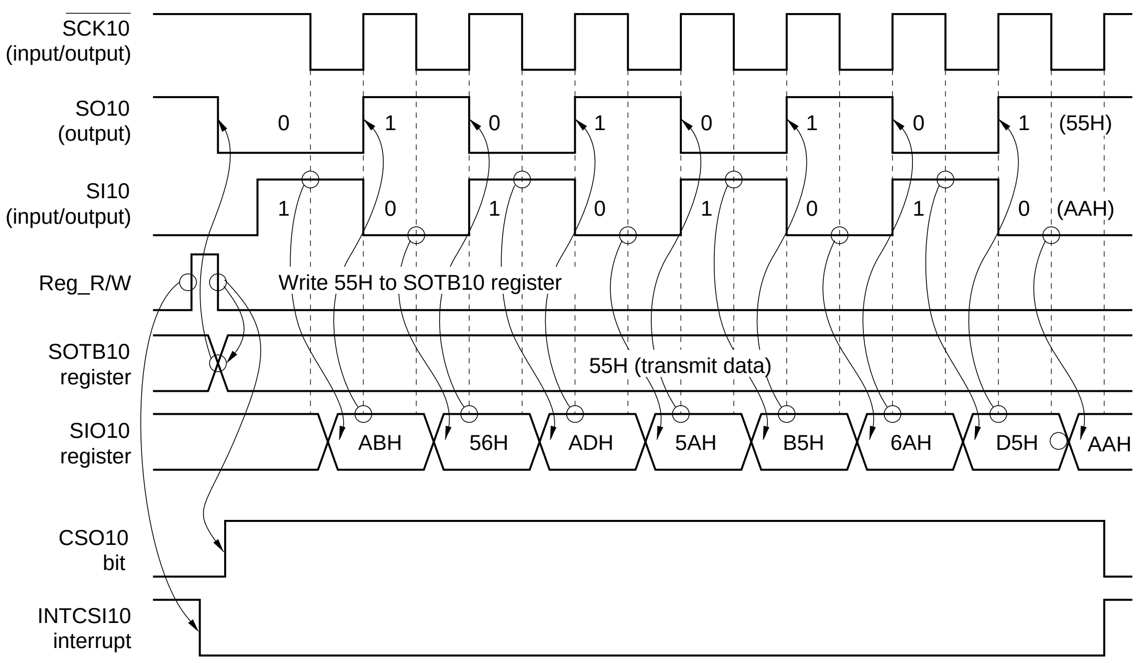
(a) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKP10 bit = 0, DAP10 bit = 0



Remark: Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIO10) read or transmit data buffer register (SOTB10) write was performed.

Figure 10-57: Timing Chart in Single Transfer Mode (DAP10 = 1) (2/2)

(b) In transmission/reception mode, data length: 8 bits, transfer direction: MSB first, no interrupt delay, single transfer mode, operation mode: CKP10 bit = 0, DAP10 bit = 1



Remark: Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIO10) read or transmit data buffer register (SOTB10) write was performed.

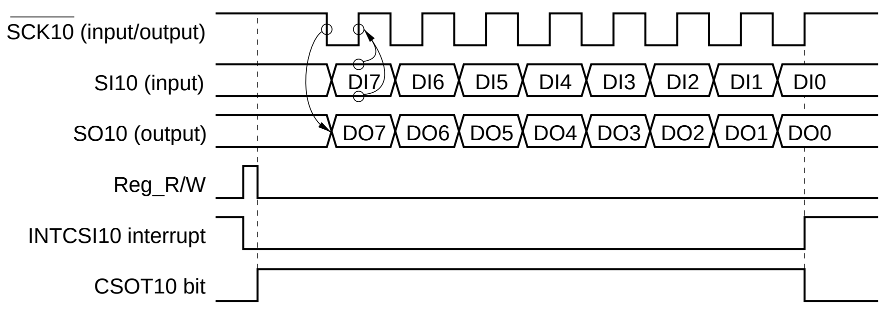
(b) Clock phase selection

The following shows the timing when changing the conditions for clock phase selection (CKP10 bit of CSIC10 register) and data phase selection (DAP10 bit of CSIC10 register) under the following conditions.

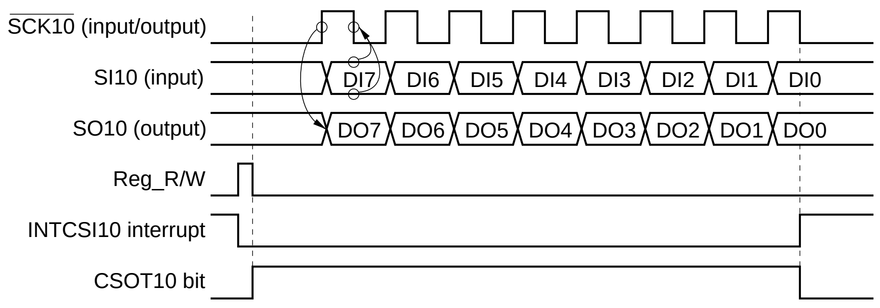
• First bit of transfer data = MSB (DIRC10 bit of CSIM10 register = 0)

Figure 10-58: Timing Chart According to Clock Phase Selection (1/2)

(a) When CKP10 bit = 0, DAP10 bit = 0



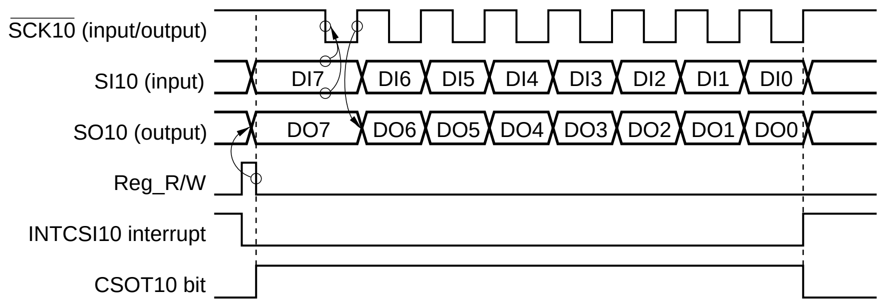
(b) When CKP10 bit = 1, DAP10 bit = 0



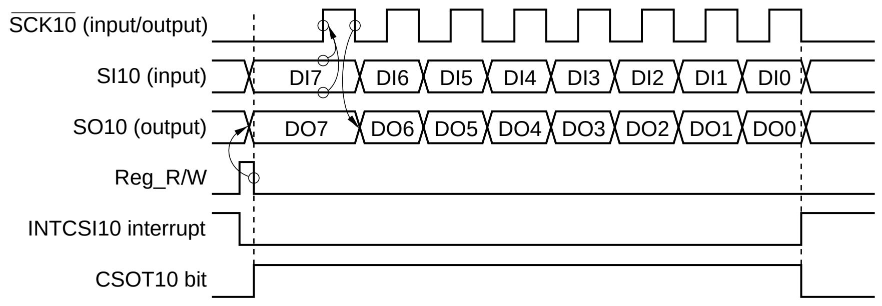
Remark: Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIO10) read or transmit data buffer register (SOTB10) write was performed.

Figure 10-58: Timing Chart According to Clock Phase Selection (2/2)

(c) When CKP10 bit = 0, DAP10 bit = 1



(d) When CKP10 bit = 1, DAP10 bit = 1



Remark: Reg_R/W:Internal signal. This signal indicates that received data buffer register (SIO10) read or transmit data buffer register (SOTB10) write was performed.

(c) Transmission/reception completion interrupt request signals (INTCSI10)

INTCSI10 is set (1) upon completion of data transmission/reception.

10.4.5 Output pins

(1) $\overline{\text{SCK10}}$ pin

When the CSI10 operation is disabled (CSIE10 bit of CSIM10 register = 0), the $\overline{\text{SCK10}}$ pin output status is as follows.

CKP10	CKS102	CKS101	CKS100	$\overline{\text{SCK10}}$ Pin Output
0	X	X	X	Fixed to high level
1	1	1	1	Fixed to high level
	Other than above			Fixed to low level

Remarks: 1. When any of bits CKP10 and CKS102 to CKS100 of the CSIC10 register is overwritten, the $\overline{\text{SCK10}}$ pin output changes.

2. X = Don't care

(2) SO10 pin

When the CSI10 operation is disabled (CSIE10 bit of CSIM10 register = 0), the SO10 pin output status is as follows.

TRMD10	DAP10	DIRC10	SO10 Pin Output
0	X	X	Fixed at low level
1	0	X	SO10 latch value (low level)
	1	0	SOTB107 value
		1	SOTB100 value

Remarks: 1. When any of bits TRMD10, DIRC10 and CSIE10 of the CSIM10 register or DAP10 bit of the CSIC10 register is overwritten, the SO10 pin output changes.

2. SOTB10m: Bit m of SOTB10 register (m = 0, 7)

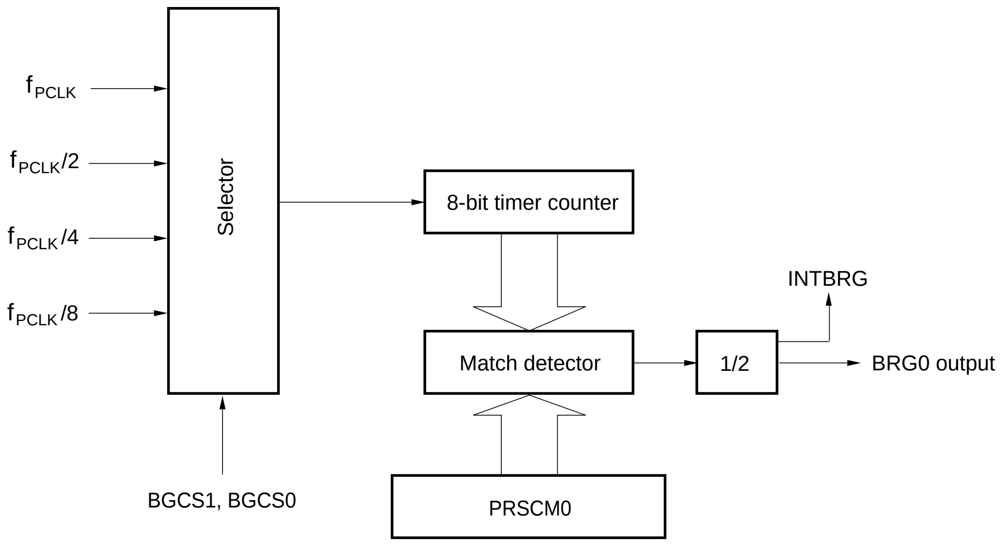
3. X = Don't care

10.5 Baud rate generator BRGC0

The baud rate generator has the capability to generate a transmission/reception clock for CSI00, CSI01and CSI10.

It consists of a programmable prescaler and a divider.

Figure 10-59: Baud Rate Generator 0 (BRG0) Block Diagram



(1) Control register

(a) Prescaler mode register (PRSM0)

The PRSM0 register controls the prescaler.
PRSM0 can be read or written in 1 or 8 bits units.

Figure 10-60: Prescaler Mode Register (PRSM0) Format

	7	6	5	4	3	2	1	0	Address	After reset
PRSM0	0	0	0	PRSMCE00	0	0	BGCS01	BGCS00	FFFF FCA0H	00H
	R	R	R	R/W	R	R	R/W	R/W		

PRSMCE0	Enables/disables operation of baud rate generator
0	Disables operation of the baud rate generator
1	Enables operation of the baud rate generator

BGCS01	BGCS00	Set the clock frequency for the divider
0	0	divider input frequency is PCLK
0	1	divider input frequency is PCLK / 2
1	0	divider input frequency is PCLK / 4
1	1	divider input frequency is PCLK / 8

Remark: PCLK is the peripheral clock, equal to the CPU clock in HELIOS

Cautions: 1. Changing the values of BGCS00 or BGCS01 is not allowed while PRSMCE00 is 1.

2. Bit 2 of PRSM0 should be kept at 0.

(b) Prescaler compare register (PRSCM0)

PRSCM0 holds the value of the frequency divisor.
PRSCM0 is an 8 bits register and can be read or written in 1- or 8-bits units.

Figure 10-61: Prescaler Compare Register (PRSCM0) Format

	7	6	5	4	3	2	1	0	Address	After reset
PRSCM0	PRSCM07	PRSCM06	PRSCM05	PRSCM04	PRSCM03	PRSCM02	PRSCM01	PRSCM00	FFFF FCA1H	00H
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W		

(2) Generation of baud rate

The bit rate value can be calculated with the following formulas:

The output frequency of the prescaler is f_{PRS} = PCLK / 2^{m+1}, where m is the value set in BGCS01 and BGCS00.

The bit rate is $f_{BRG} = f_{PRS} / (2 \times N)$, where N is the value set in PRSCM0.
(if PRSCM0 = 0, N = 256)

Caution: the bit rate delivered to CSI00, CSI01 and CSI10 should not be greater than 8 MHz.

Example

When f_{XX}= 8 MHz, PLL is enabled and PDIV = 0, PCLK = 32 MHz.

Setting BGCS00 and BGCS01 to 0 gives f_{PRS} = 16 MHz.

Setting the value of 1 in PRSCM0 results in a bit rate of f_{BRG} = 8 MHz

Chapter 11 FCAN Interface Function

11.1 Features

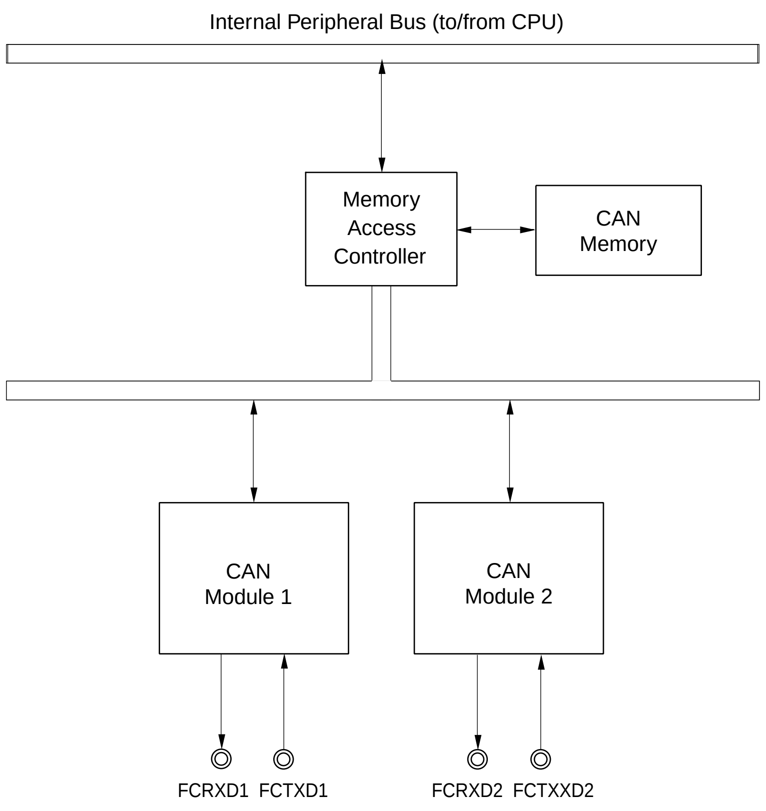
- Active support of extended format (ISO 11898, former CAN specification version 2.0B active), supporting transmission and reception of standard and extended frame format messages
- 2 CAN modules
- CAN bus speed up to 1 Mbit per second
- Direct message storage for minimum CPU burden
- Configurable number 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

11.2 Outline of the FCAN System

11.2.1 General

The FCAN (Full-CAN) system of the V850E/CA4 supports 2 independent CAN modules (CAN module 1, CAN module 2), 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.

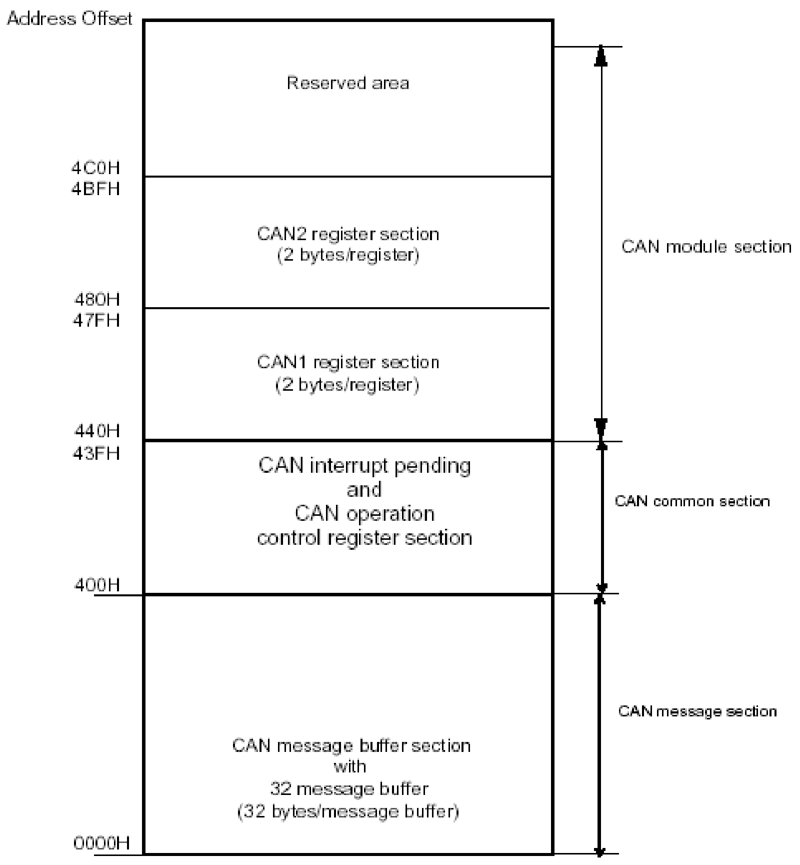
Figure 11-1: Functional Blocks of the FCAN Interface



11.2.2 CAN memory and register layout

All buffers and registers of the FCAN system are arranged within a memory layout of 4.5 KB.

Figure 11-2: Memory Area of the FCAN System



The sections within the FCAN memory layout contain areas, which are defined as illegal addresses or CANx temporary buffer (x = 1 to 2).

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 area address calculation

The CAN memory area shown above is located in the programmable peripheral area, at offset 2800H.

The value of the BPC register sets the physical address bits A14 to A29 of the programmable peripheral area.

So the first address of the CAN memory area can be obtained by adding 2800H to the BPC value multiplied by 2^{14}.

For a more easy reading of this FCAN chapter, all the addresses are shown as offsets from the start of the CAN memory area, called FCAN_BASE, and are independent of the setting of BPC, i.e. independent of the physical address, which is user defined.

Example

If BPC is set with 8040H (bit15 of BPC must be set to enable the programmable peripheral area),

- the programmable peripheral area starts at physical address 100000H
- the CAN area starts at FCAN_BASE = (BPC address) $\times$ 2^{14} + FCAN offset =102800H (offset 2800H from the programmable peripheral area)

  - the message section starts at 102800H (offset 0 from CAN area)
  - the interrupt pending and operation control section starts at 102C00H (offset 400H from CAN area)
  - the CAN1 module section starts at 102C40H (offset 440H from CAN area)
  - the CAN2 module sections starts at 102C80H (offset 480H from CAN area)

Caution: Before accessing any register or buffer of the FCAN system the base address must be fixed by the BPC register.

(2) 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 11-1: Configuration of the CAN Message Buffer Section

Address Offset Note	Name
00H to 1FH	Message buffer 0
20H to 3FH	Message buffer 1
40H to 5FH	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 = FCAN_BASE + address offset

Each message buffer has the same register layout (refer to Table 11-2).

Table 11-2: CAN Message Buffer Registers Layout

Address Offset Note 1, 2	Symbol Note 1	Name	Access Type			
			R/W	1-bit	8-bit	16-bit
(m × 20H) + 00H	M_EVTm0	Message event register 0	R/W		×	
(m × 20H) + 01H	M_EVTm1	Message event register 1	R/W		×	
(m × 20H) + 02H	M_EVTm2	Message event register 2	–			
(m × 20H) + 03H	M_EVTm3	Message event register 3	R/W		×	
(m × 20H) + 04H	M_DLCm	Message data length code register	R/W		×	
(m × 20H) + 05H	M_CTRLm	Message control register	R/W		×	
(m × 20H) + 06H	M_TIMEm	Message time stamp register	R/W			×
(m × 20H) + 08H	M_DATAm0	Message data byte 0	R/W		×	
(m × 20H) + 09H	M_DATAm1	Message data byte 1	R/W		×	
(m × 20H) + 0AH	M_DATAm2	Message data byte 2	R/W		×	
(m × 20H) + 0BH	M_DATAm3	Message data byte 3	R/W		×	
(m × 20H) + 0CH	M_DATAm4	Message data byte 4	R/W		×	
(m × 20H) + 0DH	M_DATAm5	Message data byte 5	R/W		×	
(m × 20H) + 0EH	M_DATAm6	Message data byte 6	R/W		×	
(m × 20H) + 0FH	M_DATAm7	Message data byte 7	R/W		×	
(m × 20H) + 10H	M_IDLm	Message identifier register (lower half-word)	R/W			×
(m × 20H) + 12H	M_IDHm	Message identifier register (upper half-word)	R/W			×
(m × 20H) + 14H	M_CONFm	Message configuration register	R/W		×	
(m × 20H) + 15H	M_STATm	Message status register	R		×	
(m × 20H) + 16H	SC_STATm	Message set/clear status register	W			×
(m × 20H) + 18H to (m × 20H) + 1FH	–	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 = FCAN_BASE + address offset

(3) CAN Interrupt Pending Registers Section

The layout of the interrupt pending register section is shown in Table 11-3.

Table 11-3: Relative Addresses of CAN Interrupt Pending Registers

Address Offset[Note]	Symbol	Name	Access Type				Comment
			R/W	1-bit	8-bit	16-bit	
404H	CCINTP	CAN interrupt pending register	R		×	×	
420H	CGINTP	CAN global interrupt pending register	R		×	×	
			W			×	bit-set function only
422H	C1INTP	CAN1 interrupt pending register	R		×	×	
			W		×	×	bit-clear function only
424H	C2INTP	CAN2 interrupt pending register	R		×	×	
			W		×	×	bit-clear function only

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = FCAN_BASE + address offset

(4) CAN Common Registers Section

The layout of the common register section is shown in Table 11-4.

Table 11-4: Relative Addresses of CAN Common Registers

Address Offset[Note]	Symbol	Name	Access Type				Comment
			R/W	1-bit	8-bit	16-bit	
400H	CSTOP	CAN stop register	R/W		×	×	
410H	CGST	CAN global status register	R	×	×	×	
			W			×	bit set/clear function
412H	CGIE	CAN global interrupt enable register	R	×	×	×	
			W			×	bit set/clear function
414H	CGCS	CAN main clock select register	R	×	×	×	
			W	×	×	×	only if GOM bit = 0
416H	CGTEN	CAN timer event enable register	R/W	×	×	×	
418H	CGTSC	CAN global time system counter	R	×	×	×	
			W			×	complete clear only
41AH	CGMSS	CAN message search start register	W			×	write only
	CGMSR	CAN message search result register	R	×	×	×	read only
41CH	CTBR	CAN test bus register	R/W		×	×	

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = FCAN_BASE + address offset

(5) CAN Module Registers Section

The appropriate register section of each CAN module is shown in Table 11-5 for CAN module 1, in Table 11-6 for CAN module 2.

Table 11-5: Relative Addresses of CAN Module 1 Registers

Address Offset[Note2]	Symbol	Name	Access Type				Comment
			R/W	1-bit	8-bit	16-bit	
440H	C1MASKL0	CAN1 mask 0 register L	R/W		×	×	lower half-word
442H	C1MASKH0	CAN1 mask 0 register H	R/W		×	×	upper half-word
444H	C1MASKL1	CAN1 mask 1 register L	R/W		×	×	lower half-word
446H	C1MASKH1	CAN1 mask 1 register H	R/W		×	×	upper half-word
448H	C1MASKL2	CAN1 mask 2 register L	R/W		×	×	lower half-word
44AH	C1MASKH2	CAN1 mask 2 register H	R/W		×	×	upper half-word
44CH	C1MASKL3	CAN1 mask 3 register L	R/W		×	×	lower half-word
44EH	C1MASKH3	CAN1 mask 3 register H	R/W		×	×	upper half-word
450H	C1CTRL	CAN1 control register	R		×	×	
			W			×	bit set/clear function
452H	C1DEF	CAN1 definition register	R		×	×	
			W			×	bit set/clear function
454H	C1LAST	CAN1 information register	R		×	×	read only
456H	C1ERC	CAN1 error counter register	R		×	×	read only
458H	C1IE	CAN1 interrupt enable register	R		×	×	
			W			×	bit set/clear function
45AH	C1BA	CAN1 bus activity register	R		×	×	
			W			×	bit set/clear function
45CH	C1BRP	CAN1 bit rate prescaler register	R		×	×	
			W		×	×	in initialisation state only (ISTAT = 1)
	C1DINF	CAN1 bus diagnostic information register	R		×	×	in diagnostic mode only
45EH	C1SYNC	CAN1 synchronization control register	R		×	×	
			W		×	×	

Note: The address of an interrupt pending register is calculated according to the following formula: effective address = FCAN_BASE + address offset.

Table 11-6: Relative Addresses of CAN Module 2 Registers

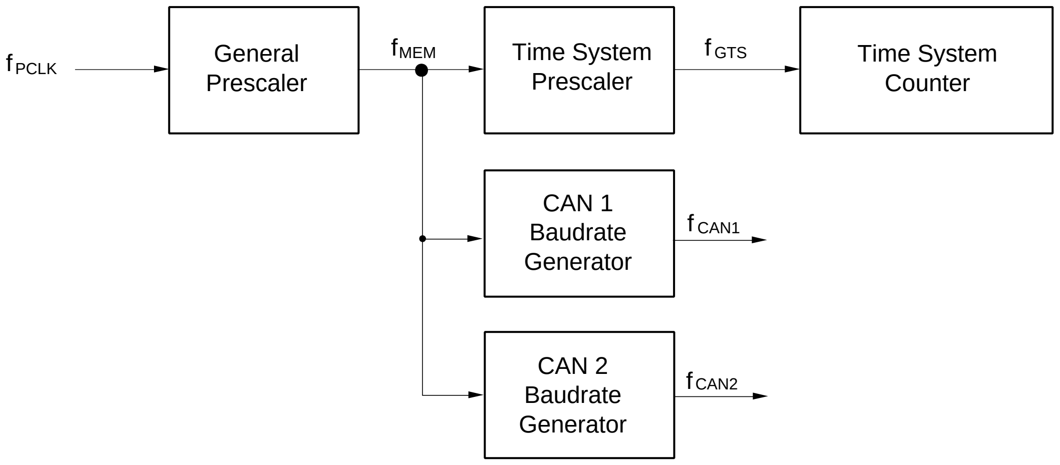
Address Offset[Note]	Symbol	Name	Ref. Page	Access Type				Comment
				R/W	1-bit	8-bit	16-bit	
480H	C2MASKL0	CAN2 mask 0 register L	178	R/W		×	×	lower half-word
482H	C2MASKH0	CAN2 mask 0 register H		R/W		×	×	upper half-word
484H	C2MASKL1	CAN2 mask 1 register L		R/W		×	×	lower half-word
486H	C2MASKH1	CAN2 mask 1 register H		R/W		×	×	upper half-word
488H	C2MASKL2	CAN2 mask 2 register L		R/W		×	×	lower half-word
48AH	C2MASKH2	CAN2 mask 2 register H		R/W		×	×	upper half-word
48CH	C2MASKL3	CAN2 mask 3 register L		R/W		×	×	lower half-word
48EH	C2MASKH3	CAN2 mask 3 register H		R/W		×	×	upper half-word
490H	C2CTRL	CAN2 control register	180	R		×	×	
				W			×	bit set/clear function
492H	C2DEF	CAN2 definition register	184	R		×	×	
				W			×	bit set/clear function
494H	C2LAST	CAN2 information register	188	R		×	×	read only
496H	C2ERC	CAN2 error counter register	189	R		×	×	read only
498H	C2IE	CAN2 interrupt enable register	190	R		×	×	
				W			×	bit set/clear function
49AH	C2BA	CAN2 bus activity register	192	R		×	×	
				W			×	bit-set/clear function
49CH	C2BRP	CAN2 bit rate prescaler register	195	R		×	×	
				W		×	×	in initialisation state only (ISTAT bit = 1)
	C2DINF	CAN2 bus diagnostic information register	200	R		×	×	in diagnostic mode only
49EH	C2SYNC	CAN2 synchronization control register	197	R		×	×	
				W		×	×	

Note: The address of a CAN module 2 register is calculated according to the following formula: effective address = FCAN_BASE + address offset

11.2.3 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 11-3: Clock Structure of the FCAN System

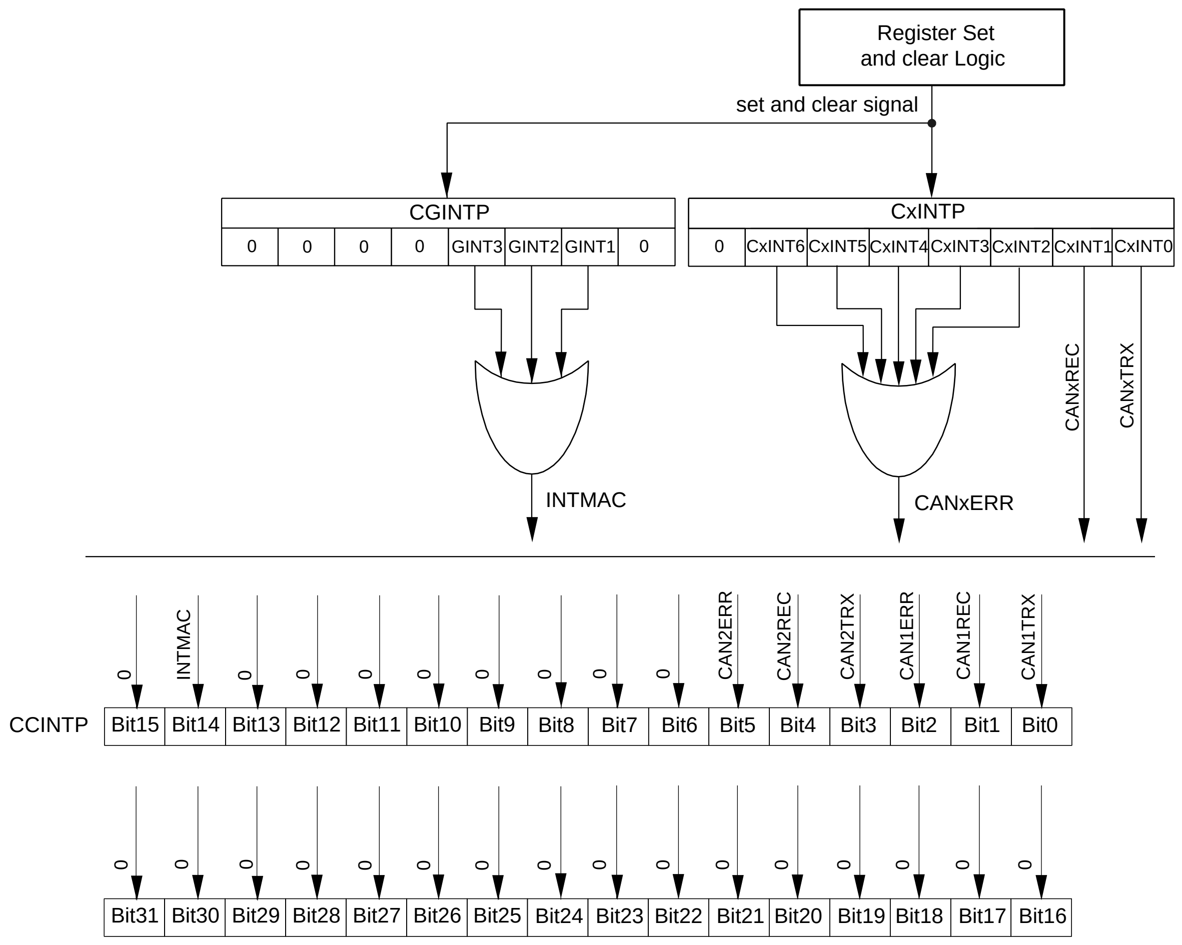


A functional block for a global time system is integrated in the FCAN system. That functional block is supplied by the global time system clock (f_{GTS}), which is derived from f_{MEM}. The time system prescaler scales f_{GTS} and is controlled by the CGCS register.
The time base of the global time system 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.

11.2.4 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/CA4 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 11-4: FCAN Interrupt Bundling of V850E/CA4


Register Set
and clear Logic
set and clear signal
CGINTP
0 0 0 0 GINT3 GINT2 GINT1 0
CxINTP
0 CxINT6 CxINT5 CxINT4 CxINT3 CxINT2 CxINT1 CxINT0
INTMAC
CANxERR
CANxREC
CANxTRX
0 INTMAC 0 0 0 0 0 0 0 0 CAN2ERR CAN2REC CAN2TRX CAN1ERR CAN1REC CAN1TRX
CCINTP
Bit15 Bit14 Bit13 Bit12 Bit11 Bit10 Bit9 Bit8 Bit7 Bit6 Bit5 Bit4 Bit3 Bit2 Bit1 Bit0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Bit31 Bit30 Bit29 Bit28 Bit27 Bit26 Bit25 Bit24 Bit23 Bit22 Bit21 Bit20 Bit19 Bit18 Bit17 Bit16

Remark: x = 1 to 2

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

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 section 11.3.3 ”CAN interrupt pending registers” on page 357.

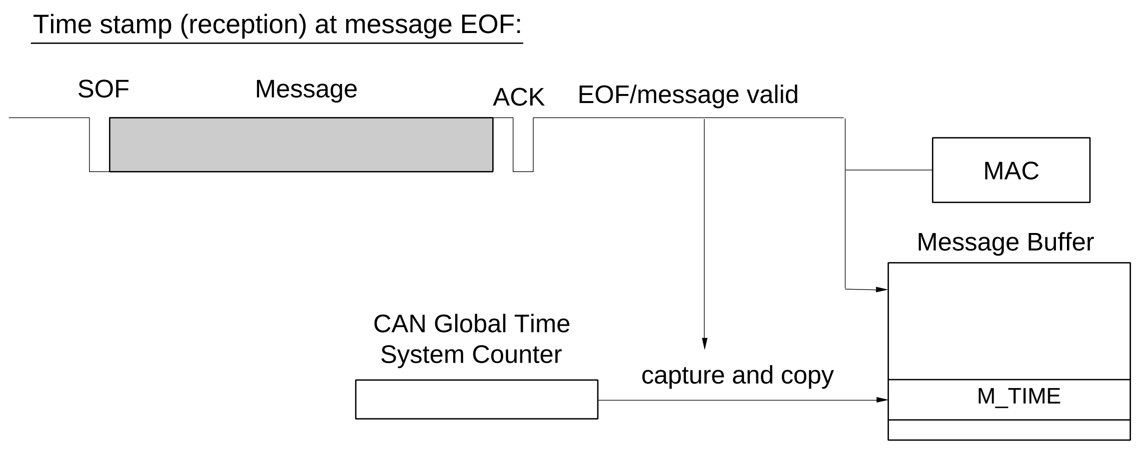
11.2.5 Time stamp

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.

The time stamp function is available at message reception (see Figure 11-5, “Time Stamp Capturing at Message Reception,” on page 331): the counter value of the CAN global time system counter (CGTSC) 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 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 11-5: 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 11-6, “Time Stamp Capturing at Message Transmission,” on page 332). The captured value from the CGTSC counter is written into particular data bytes of the transmit message’s data field. Table 11-7, “Transmitted Data on the CAN Bus (ATS = 1),” on page 332 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 11-6: Time Stamp Capturing at Message Transmission

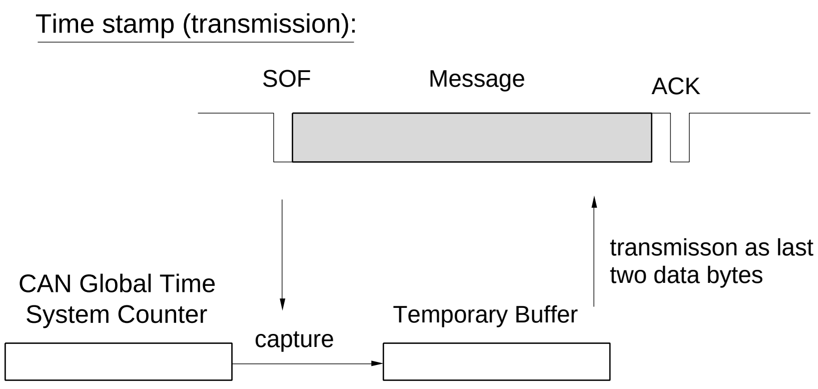

Time stamp (transmission):
SOF
Message
ACK
transmisson as last two data bytes
CAN Global Time System Counter
capture
Temporary Buffer

Table 11-7: 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

11.2.6 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 11-8, “Example for Automatic Transmission Priority Detection,” on page 334. 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 11-8: 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
3E0H	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				
E0H	7				
C0H	6				
A0H	5				
80H	4				
60H	3				
40H	2	CAN 1	TRX	3	229H
20H	1	CAN 1	TRX	3	120H
00H	0				

Notes: 1. The address of a message buffer entry is calculated according to the following formula:
effective address = FCAN_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 11-9: Example for Transmit Buffer Allocation When More Than 5 Buffers Linked to a CAN Module

Message Buffer Address Offset[Note1]	Message Buffer Number	Message Buffer Link	Message Buffer Type[Note2]	Identifier
3E0H	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			
E0H	7			
C0H	6	CAN1	TRX	000H [Note 3]
A0H	5			
80H	4			
60H	3			
40H	2	CAN1	TRX	004H [Note 3]
20H	1	CAN1	TRX	002H [Note 3]
00H	0			

Notes: 1. The address of a message buffer entry is calculated according to the following formula: effective address = FCAN_BASE + address offset

2. TRX = transmit message

3. 5 higher prior transmit messages assigned to messages buffer 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 11-10: Storage Priority for Reception of Data Frames** and
Table 11-11: Storage Priority for Reception of Remote Frames).

Table 11-10: 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 11-11: 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 11-12: 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 areas 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).

11.2.7 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

11.2.8 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 11-13).
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 11-13: Remote Frame Handling upon Reception into a Transmit Message Buffer 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 11-13: 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

Caution: It is recommended not to use the same buffer for transmission and reception of remote frames. One buffer for reception and another buffer for transmission should be used. A transmit buffer should be used for transmit of remote frames. A receive buffer should be used for reception of remote frames.

11.2.9 Outer priority inversion

It is not guaranteed that transmit requests will participate in the arbitration of the next CAN bus transmission slot, even if launched in time and the message ID has at least the highest priority of the first five TRQ-marked message buffers.

In other words an outer priority inversion may occur under some circumstances:

The priority inversion only takes place when in the time period in advance of the next arbitration on the bus the memory access controller (MAC) in the FCAN has to perform several tasks at once. Each task consumes a certain amount of clock cycles. The CPU activity (i.e. polling of a FCAN address), the number of tasks (i.e how many channels are active) and the task type (i.e RX search) defines the total number of clock cycles. If this number exceeds the intermission field on the CAN bus, the setup of a transmit message is done after the transmission of one of the other bus participants has started its transmission.

Due to the pure real-time behaviour and the dependencies of factors like FCAN macro frequency, baudrate on the CAN buses, number of FCAN modules and bus load it is not possible to predict the exact behaviour in all runtime conditions. A conceivable worst case scenario is for example when the RX search takes the maximum of time. This is the concern for a message object linked to mask 3 or the message will not be stored at all.

The outer priority inversion has been observed when all modules operate at the maximum macro frequency and at the maximum CAN-bus baud rate, 1MBaud, under special coincidences, such as worst case RX Search scenarios (i.e. message buffer linked to Mask3 or not matching at all).

The transmission of the affected TX message is only delayed for one frame on the bus; it is not lost.

There is basically no work around available. It can only be stated that

- the lower the baudrate the lower the possibility of outer priority inversion
- the lower the CAN macro frequency the higher the possibility of priority inversion
- the less CAN modules are used the less the possibility of priority inversion.

11.3 Control and Data Registers

11.3.1 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

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 Figure 11-7: on page 344.

Figure 11-7: 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

Bit Name	Function
SE_n	Sets the register bit n. 0: No change of register bit n 1: Register bit n is set (1)
CL_n	Clears the register bit n. 0: No change of register bit n 1: Register bit n is cleared (0)
SE_n, CL_n	Sets/clears the Register bit n.

SE_n	CL_n	Status of Register Bit n
0	1	Register bit n is cleared (0)
1	0	Register bit n is set (1)
Others		No change in register bit n value.

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

11.3.2 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 11-8: 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	400H	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 = FCAN_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 section 11.2.3 ”Clock structure” on page 329).
These register can be read/written in 1-bit, 8-bit and16-bit units.

Figure 11-9: 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	414H	7F05H

Bit Position	Bit Name	Function
15 to 8	CGTS7 to CGTS0	Specifies the 8-bit prescaler compare value for the global time system clock (f_{GTS}) (refer to Figure 11-11, “Configuration of FCAN Global Time System Clock,” on page 347).

CGTS7 to CGTS0 (k)	Prescaler (k + 1)	Global Time System Clock $f_{GTS} = f_{GTS1} / (k + 1)$
0	1	$f_{GTS} = f_{GTS1}$
1	2	$f_{GTS} = f_{GTS1} / 2$
2	3	$f_{GTS} = f_{GTS1} / 3$
⋮	⋮	⋮
255	256	$f_{GTS} = f_{GTS1} / 256$

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.

Bit Position	Bit Name	Function
7, 6	GTCS1, GTCS0	Selects the global time system basic clock (f_{GTS1}) from the memory clock (f_{MEM}) (refer to Figure 11-11, “Configuration of FCAN Global Time System Clock,” on page 347).

GTCS1	GTCS0	Global Time System Basic Clock (f_{GTS1})
0	0	$f_{GTS1} = f_{MEM} / 2$
0	1	$f_{GTS1} = f_{MEM} / 4$
1	0	$f_{GTS1} = f_{MEM} / 8$
1	1	$f_{GTS1} = f_{MEM} / 16$

Bit Position	Bit Name	Function
4	MCS	Selects input clock for the memory access clock prescaler (f_{MEM1}) (refer to Figure 11-10, “Configuration of FCAN System Main Clock,” on page 347). 0: f_{MEM1} = internal system clock (f_{CPU}) 1: f_{MEM1} = external clock input (f_{EXT})[Note] **Note:** V850E/CA4 doesn’t offer an external clock f_{MEM} supply pin. Therefore, the MCS must not be set at any time.

Note: The address of an interrupt pending register is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Figure 11-9: 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}) (refer to Figure 11-10, “Configuration of FCAN System Main Clock,” on page 347).

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	$f_{MEM} = f_{MEM1} / 16$

Caution: Changing the MCP value from 0 to any other value can be done just once after reset; changing the MCP value from any value different from 0 to any other value can be done at any time.

Figure 11-10: Configuration of FCAN System Main Clock

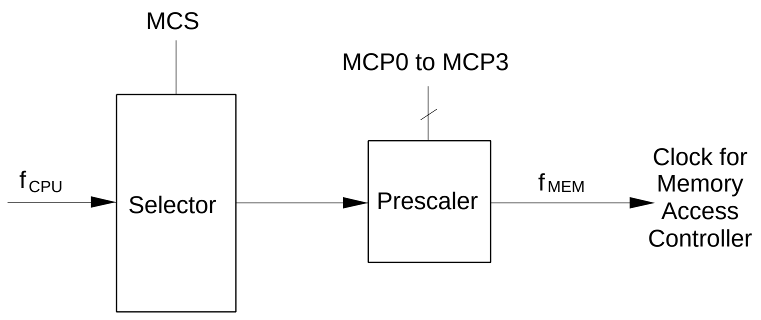
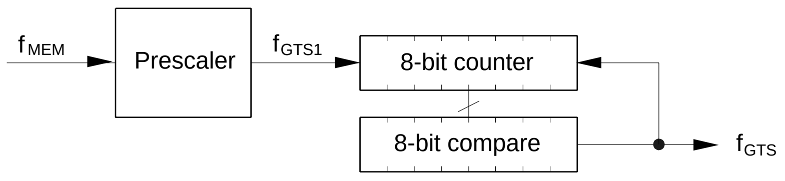


Figure 11-11: Configuration of FCAN Global Time System Clock



(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 section 11.3.1 ”Bit set/clear function” on page 343.)

Figure 11-12: 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	0	GOM	410H	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	0	ST_ GOM	CL_ MERR	0	0	0	CL_ EFSD	CL_ TSM	0	CL_ GOM	410H

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 initialisation 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.

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Figure 11-12: 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 behavior 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 2) is set (1)), the GOM flag remains set.**

Figure 11-12: CAN Global Status Register (CGST) (3/3)

Write

Bit Position	Bit Name	Function
11, 3	ST_EFSD, CL_EFSD	Sets/clears the EFSD bit.
10, 2	ST_TSM, CL_TSM	Sets/clears the TSM bit.
9, 1	ST_EVM, CL_EVM	Sets/clears the EVM bit.
8, 0	ST_GOM, CL_GOM	Sets/clears the GOM bit.
7	CL_MERR	Clears the MERR bit. 0: No change of MERR bit. 1: MERR bit is cleared (0).

ST_EFSD	CL_EFSD	Status of EFSD Bit
0	1	EFSD bit is cleared (0).
1	0	EFSD bit is set (1).
Others		No change in EFSD bit value.

ST_TSM	CL_TSM	Status of TSM Bit
0	1	TSM bit is cleared (0).
1	0	TSM bit is set (1).
Others		No change in TSM bit value.

ST_EVM	CL_EVM	Status of EVM Bit
0	1	EVM bit is cleared (0).
1	0	EVM bit is set (1).
Others		No change in EVM bit value.

ST_GOM	CL_GOM	Status of GOM Bit
0	1	GOM bit is cleared (0).Note 2
1	0	GOM bit is set (1).
Others		No change in GOM bit value.

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 chapter 9.3.1)

Figure 11-13: 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	0	0	0	0	0	G_IE2	G_IE1	0	412H	0000H

Write	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
CGIE	0	0	0	0	0	ST_ G_IE2	ST_ G_IE1	0	0	0	0	0	0	CL_ G_IE2	CL_ G_IE1	0	412H

Read

Bit Position	Bit Name	Function
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 = FCAN_BASE + address offset

Figure 11-13: CAN Global Interrupt Enable Register (CGIE) (2/2)

Write

Bit Position	Bit Name	Function
10, 2	ST_G_IE2, CL_G_IE2	Sets/clears the G_IE2 bit.

ST_G_IE2	CL_G_IE2	Status of G_IE2 Bit
0	1	G_IE2 bit is cleared (0).
1	0	G_IE2 bit is set (1).
Others		No change in G_IE2 bit value.

Bit Position	Bit Name	Function
9, 1	ST_G_IE1, CL_G_IE1	Sets/clears the G_IE1 bit.

ST_G_IE1	CL_G_IE1	Status of G_IE1 Bit
0	1	G_IE1 bit is cleared (0).
1	0	G_IE1 bit is set (1).
Others		No change in G_IE1 bit value.

(5) 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 sections 11.2.3 ”Clock structure” on page 329 and 11.2.5 ”Time stamp” on page 331).

This register can be read and written**Note 1** in 16-bit units only.

Figure 11-14: 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	418H	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 = FCAN_BASE + address offset

Remark: The CGTSC register can be read at any time.

(6) 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 11-15: 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	41AH	–

Bit Position	Bit Name	Function
15	CIDE	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).
14	CERQ	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.
13	CTRQ	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.
12	CMSK	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.
11	CDN	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.
9, 8	SMN1, SMN0	Specifies the CAN module number to search for. (see table below) **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.
5 to 0	STRT5 to STRT0	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.

SMN2	SMN1	SMN0	CAN Module Number
0	0	0	Search for message buffers not linked to any CAN module.
0	0	1	Search for message buffers linked to CAN module 1
0	1	0	Search for message buffers linked to CAN module 2

Note: The address of an interrupt pending register is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Remark: m = 00 to 31

(7) 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 11-16: 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
CGMSR	0	0	0	0	0	0	MM	AM	0[Note3]	0[Note3]	MFND5	MFND4	MFND3	MFND2	MFND1	MFND0	41AH	0000H

Bit Position	Bit Name	Function
9, 8	MM, AM	Indicates the match result of the preceding message search. MM | AM | Number of Hits × | 0 | No match 0 | 1 | 1 message meets the search criteria. 1 | 1 | Several message meet the search criteria.[Note 2]
5 to 0	MFND5 to MFND0	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.

Notes: 1. The register address is calculated according to the following formula:
effective address = FCAN_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.

(8) 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.

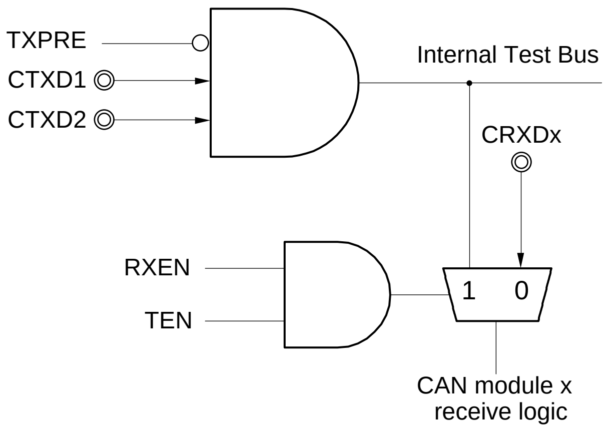
Figure 11-17: CAN Test Bus Register (CTBR)

	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	41CH	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 an 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 11-18: 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 and CAN module 2) 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

Caution: The internal test bus must only be used when none of the CAN modules are connected to a CAN bus.

11.3.3 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/CA4.
This register is read-only and can be read in 8-bit and16-bit units.

Figure 11-19: 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	0	0	0	406H	0000H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address Offset[Note 1]	Initial value
CCINTPL	0	INTMAC	0	0	0	0	0	0	0	0	CAN2ERR	CAN2REC	CAN2TRX	CAN1ERR	CAN1REC	CAN1TRX	404H	0000H

Bit Position	Bit Name Note 2, 3	Function
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
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
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
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 = FCAN_BASE + address offset

2. x = 1 to 2.

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 and C2INTP.

(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 11.3.1 ”Bit set/clear function” on page 343)
This register can be read in 8-bit and 16-bit units. It can be written in 16-bit units only.

Figure 11-20: 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	0	0	0	0	GINT3	GINT2	GINT1	0	420H	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	0	0	0	0	CL_ GINT3	CL_ GINT2	CL_ GINT1	0	420H

Read

Bit Position	Bit Name	Function
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 11-2: ”Memory Area of the FCAN System” on page 321). **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 = FCAN_BASE + address offset

Figure 11-20: CAN Global Interrupt Pending Register (CGINTP) (2/2)

Write

Bit Position	Bit Name	Function
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 and GINT2 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 11.3.1 ”Bit set/clear function” on page 343)
This register can be read and written in 8-bit and16-bit units.

Figure 11-21: 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	422H	0000H
C2INTP	0	0	0	0	0	0	0	0	0	C2INT6	C2INT5	C2INT4	C2INT3	C2INT2	C2INT1	C2INT0	424H	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	422H
C2INTP	0	0	0	0	0	0	0	0	0	CL_ C2INT6	CL_ C2INT5	CL_ C2INT4	CL_ C2INT3	CL_ C2INT2	CL_ C2INT1	CL_ C2INT0	424H

Read (1/2)

Bit Position	Bit Name **Note 2**	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 an error passive status on reception of CAN module x. 0: No Interrupt pending 1: Interrupt pending
2	CxINT2	Indicates an error passive or bus off status on transmission 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 = FCAN_BASE + address offset

2. x = 1 to 2.

Figure 11-21: CAN 1 to 4 Interrupt Pending Registers (C1INTP to C2INTP) (2/2)

Read (2/2)

Bit Position	Bit Name **Note**	Function
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).

Note: x = 1 to 2.

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.

11.3.4 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 11-22: 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	12H +(m x 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	10H +(m x 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 = FCAN_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 11-23: 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	14H + (m × 20H)	undef.

Bit Position	Bit Name	Function
5 to 3	MT2 to MT0	Specifies the message type and mask link.

MT2	MT1	MT0	Message type and mask link
0	0	0	Transmit message
0	0	1	Receive message, no mask linked
0	1	0	Receive message, mask 0 linked[Note 2]
0	1	1	Receive message, mask 1 linked[Note 2]
1	0	0	Receive message, mask 2 linked[Note 2]
1	0	1	Receive message, mask 3 linked[Note 1]
1	1	0	Reserved[Note 3]
1	1	1	Receive message in diagnostic mode (type 7)[Note 4]

Figure 11-23: 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	Reserved
1	0	0	Reserved
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 = FCAN_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.

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 11-24: Message Status Registers 00 to 31 (M_STAT00 to M_STAT31)

	7	6	5	4	3	2	1	0	Address Offset[Note1]	Initial value
M_STATm	0	0	0	0	0	DN	TRQ	RDY	15H + (m × 20H)	undef.

Bit Position	Bit Name	Function
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)

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Remark: m = 00 to 31

Processing of a transmit or receive message by TRQ and RDY flags is summarized in following Table:

Table 11-14: 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 11-25: 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_STAT m	0	0	0	0	0	ST_ DN	ST_ TRQ	ST_ RDY	0	0	0	0	0	CL_ DN	CL_ TRQ	CL_ RDY	16H +(m × 20H)	–

Bit Position	Bit Name	Function
10, 2	ST_DN, CL_DN	Sets/clears the DN bit of the M_STATm register.

ST_DN	CL_DN	Status of DN bit
0	1	DN bit is cleared (0).
1	0	DN bit is set (1).
Others		No change in DN bit value.

Bit Position	Bit Name	Function
9, 1	ST_TRQ, CL_TRQ	Sets/clears the TRQ bit of the M_STATm register.

ST_TRQ	CL_TRQ	Status of TRQ bit
0	1	TRQ bit is cleared (0).
1	0	TRQ bit is set (1).
Others		No change in TRQ bit value.

Bit Position	Bit Name	Function
8, 0	ST_RDY, CL_RDY	Sets/clears the RDY bit of the M_STATm register.

ST_RDY	CL_RDY	Status of RDY bit
0	1	RDY bit is cleared (0).
1	0	RDY bit is set (1).
Others		No change in RDY bit value.

Note: The register address is calculated according to the following formula:
effective address = FCAN_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 11-26: 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_DATAm0	D0_7	D0_6	D0_5	D0_4	D0_3	D0_2	D0_1	D0_0	08H + (m × 20H)	undef.
M_DATAm1	D1_7	D1_6	D1_5	D1_4	D1_3	D1_2	D1_1	D1_0	09H + (m × 20H)	undef.
M_DATAm2	D2_7	D2_6	D2_5	D2_4	D2_3	D2_2	D2_1	D2_0	0AH + (m × 20H)	undef.
M_DATAm3	D3_7	D3_6	D3_5	D3_4	D3_3	D3_2	D3_1	D3_0	0BH + (m × 20H)	undef.
M_DATAm4	D4_7	D4_6	D4_5	D4_4	D4_3	D4_2	D4_1	D4_0	0CH + (m × 20H)	undef.
M_DATAm5	D5_7	D5_6	D5_5	D5_4	D5_3	D5_2	D5_1	D5_0	0DH + (m × 20H)	undef.
M_DATAm6	D6_7	D6_6	D6_5	D6_4	D6_3	D6_2	D6_1	D6_0	0EH + (m × 20H)	undef.
M_DATAm7	D7_7	D7_6	D7_5	D7_4	D7_3	D7_2	D7_1	D7_0	0FH + (m × 20H)	undef.

Figure 11-26: 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 = FCAN_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) (refer to chapter 11.2.5 "Time stamp" on page 331)**
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 only in 8-bit units.

Figure 11-27: 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	04H + (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 = FCAN_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 11-28: 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_CTRLm	RMDE1	RMED0	ATS	IE	MOVR	R1	R0	RTR	805H + (m × 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 11.2.8 "Remote frame handling" on page 339)
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 11.2.8 "Remote frame handling" on page 339)
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 11.2.5 "Time stamp" on page 331)

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Remark: m = 00 to 31

Figure 11-28: Message Control Registers 00 to 31 (M_CTRL00 to M_CTRL31) (2/2)

Bit Position	Bit Name	Function
4	IE	Enables message buffer m related interrupts. 0: Interrupts related to message buffer m disabled. 1: Interrupts related to message buffer m enabled. If the message related interrupt is enabled, an interrupt is generated for any of the following conditions:

Condition	Related Interrupt
Data frame or remote frame is transmitted from transmit message buffer.	CANxTRX
Data frame or remote frame is received on receive message buffer.	CANxREC
Remote frame is received on transmit message without auto answering set (RMDE0 = 0).	

An interrupt is not generated, even if enabled, for any of the following conditions:

Condition
Remote frame is received on a transmit message with auto answering mode (RMDE0 = 1).
Remote frame is transmitted from receive message buffer.

Bit Position	Bit Name	Function
3	MOVR	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.
2	R1	Reserved bit (value of CAN bus bit r0 for receive message buffer)
1	R0	Reserved bit (value of CAN bus bit r1 for receive message buffer)
0	RTR	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.

Remarks: 1. m = 00 to 31

2. x = 1 to 2.

(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 11-29: 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	06H + (m x 20H)	undef.

Bit Position	Bit Name	Function
15 to 0	TS15 to TS0	16-bit time stamp value captured on message reception.

Note: The address of a message time stamp register is calculated according to the following formula: effective address = FCAN_BASE + address offset

Remarks: **1.** m = 00 to 31

2. x = 1 to 2.

(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 have no function in the V850E/CA4. 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.

These registers can be read/written in 8-bit units.

11.3.5 CAN Module Registers

(1) CAN 1 to 2 mask 0 to 3 registers L, H (CxMASKL0 to CxMASKL3, CxMASKH0 to CxMASKH3) (x = 1 to 2)

The CxMASKL0 to CxMASKL3, and CxMASKH0 to CxMASKH3 registers specify the four acceptance masks for each CAN module x (x = 1 to 2). (For more details refer to chapter 11.2.7 ”Mask handling” on page 338).
These registers can be read/written in 8-bit and 16-bit units.

Figure 11-30: CAN 1 to 2 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 11-15	undef.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
CxMASKLn	CMID15	CMID14	CMID13	CMID12	CMID11	CMID10	CMID9	CMID8	CMID7	CMID6	CMID5	CMID4	CMID3	CMID2	CMID1	CMID0	see Table 11-15	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).

Remarks: 1. n = 0 to 3 (mask number)

2. x = 1 to 2.

Table 11-15: Address Offsets of the CAN 1 to 2 Mask Registers

Symbol[Note 1,2]	Address Offset	
	x = 1	x = 2
CxMASKL0	440H	480H
CxMASKH0	442H	482H
CxMASKL1	444H	484H
CxMASKH1	446H	486H
CxMASKL2	448H	488H
CxMASKH2	44AH	48AH
CxMASKL3	44CH	48CH
CxMASKH3	44EH	48EH

Notes: 1. CAN module number: x = 1 to 2.

2. The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

(2) CAN 1 to 2 control registers (C1CTRL to C2CTRL)

The CxCTRL registers control the operating modes and indicate the operating status of the corresponding CAN module x (x = 1 to 2).

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 section 11.3.1 ”Bit set/clear function” on page 343).

Figure 11-31: CAN 1 to 2 Control Registers (C1CTRL to C2CTRL) (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	0	STOP	SLEEP	INIT	450H	0101H
C2CTRL	TECS1	TECS0	RECS1	RECS0	BOFF	TSTAT	RSTAT	ISTAT	TPE	DLEVR	DLEVT	OVM	0	STOP	SLEEP	INIT	490H	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	0	ST_STOP	ST_SLEEP	ST_INIT	CL_TPE	CL_DLEVR	CL_DLEVT	CL_OVM	1	CL_STOP	CL_SLEEP	CL_INIT	450H
C2CTRL	ST_TPE	ST_DLEVR	ST_DLEVT	ST_OVM	0	ST_STOP	ST_SLEEP	ST_INIT	CL_TPE	CL_DLEVR	CL_DLEVT	CL_OVM	1	CL_STOP	CL_SLEEP	CL_INIT	490H

Read (1/3)

Bit Position	Bit Name	Function
15, 14	TECS1, TECS0	Indicates the transmission error counter status. TECS1 | TECS0 | Transmission Error Counter Status 0 | 0 | Transmission error counter below warning level (< 96) 0 | 1 | Transmission error counter in warning level range (96 to 127) 1 | 0 | Reserved (not possible) 1 | 1 | Transmission error counter above warning level (≥ 128)
13, 12	RECS1, RECS0	Indicates the reception error counter status. RECS1 | RECS0 | Reception Error Counter Status 0 | 0 | Reception error counter below warning level (< 96) 0 | 1 | Reception error counter in warning level range (96 to 127) 1 | 0 | Reserved (not possible) 1 | 1 | Reception error counter in the error passive range (≥ 128)

Figure 11-31: CAN 1 to 2 Control Registers (C1CTRL to C2CTRL) (2/5)

Read (2/3)

Bit Position	Bit Name	Function
11	BOFF	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)
10	TSTAT	Indicates the transmission status. 0: No transmission activity on the CAN bus. 1: Transmission activity on the CAN bus.
9	RSTAT	Indicates the reception status. 0: No reception activity on the CAN bus. 1: Reception activity on the CAN bus.
8	ISTAT	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.
7	TPE	Indicates the transmit pin status. 0: CAN transmit pin is disabled (tri-state). 1: CAN transmit pin is enabled.
6	DLEVR	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.
5	DLEVT	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.
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.

Figure 11-31: CAN 1 to 2 Control Registers (C1CTRL to C4CTRL) (3/5)

Read (3/3)

Bit Position	Bit Name	Function
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 11-31: CAN 1 to 2 Control Registers (C1CTRL to C4CTRL) (4/5)

Write (1/2)

Bit Position	Bit Name	Function		
15, 7	ST_TPE, CL_TPE	Sets/clears the TPE bit.		
		ST_TPE	CL_TPE	Status of TPE bit
		0	1	TPE bit is cleared (0).
		1	0	TPE bit is set (1).
		Others		No change in TPE bit value.
14, 6	ST_DLEVR, CL_DLEVR	Sets/clears the DLEVR bit.		
		ST_DLEVR	CL_DLEVR	Status of DLEVR bit
		0	1	DLEVR bit is cleared (0).
		1	0	DLEVR bit is set (1).
		Others		No change in DLEVR bit value.
13, 5	ST_DLEVT, CL_DLEVT	Sets/clears the DLEVT bit.		
		ST_DLEVT	CL_DLEVT	Status of DLEVT bit
		0	1	DLEVT bit is cleared (0).
		1	0	DLEVT bit is set (1).
		Others		No change in DLEVT bit value.
12, 4	ST_OVM, CL_OVM	Sets/clears the OVM bit.		
		ST_OVM	CL_OVM	Status of OVM bit
		0	1	OVM bit is cleared (0).
		1	0	OVM bit is set (1).
		Others		No change in OVM bit value.
10, 2	ST_STOP, CL_STOP	Sets/clears the STOP bit.		
		ST_STOP	CL_STOP	Status of STOP bit
		0	1	STOP bit is cleared (0).
		1	0	STOP bit is set (1).
		Others		No change in STOP bit value.

Figure 11-31: CAN 1 to 4 Control Registers (C1CTRL to C4CTRL) (5/5)

Write (2/2)

Bit Position	Bit Name	Function
9, 1	ST_SLEEP, CL_SLEEP	Sets/clears the SLEEP bit.
8, 0	ST_INIT, CL_INIT	Sets/clears the INIT bit.

ST_SLEEP	CL_SLEEP	Status of SLEEP bit
0	1	SLEEP bit is cleared (0).
1	0	SLEEP bit is set (1).
Others		No change in SLEEP bit value.

ST_INIT	CL_INIT	Status of INIT bit
0	1	INIT bit is cleared (0).
1	0	INIT bit is set (1).
Others		No change in INIT bit value.

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset.

(3) CAN 1 to 2 definition registers (C1DEF to C2DEF)

The CxDEF registers define normal and diagnostic operation and indicate CAN bus error and states of the corresponding CAN module x (x = 1 to 2).

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 11.3.1 ”Bit set/clear function” on page 343).

Figure 11-32: CAN 1 to 2 Definition Registers (C1DEF to C2DEF) (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	452H	0000H
C2DEF	0	0	0	0	0	0	0	0	DGM	MOM	SSHT	PBB	BERR	VALID	WAKE	OVR	492H	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	452H
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	492H

Read (1/3)

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.

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Figure 11-32: CAN 1 to 2 Definition Registers (C1DEF to C2DEF) (2/4)

Read (2/3)

Bit Position	Bit Name	Function
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.**
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.

Figure 11-32: CAN 1 to 2 Definition Registers (C1DEF to C2DEF) (3/4)

Read (3/3)

Bit Position	Bit Name	Function
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 11-32: CAN 1 to 2 Definition Registers (C1DEF to C2DEF) (4/4)

Write

Bit Position	Bit Name	Function
15, 7	ST_DGM, CL_DGM	Sets/clears the DGM bit. ST_DGM | CL_DGM | Status of DGM bit 0 | 1 | DGM bit is cleared (0). 1 | 0 | DGM bit is set (1). Others | No change in DGM bit value.
14, 6	ST_MOM, CL_MOM	Sets/clears the MOM bit. ST_MOM | CL_MOM | Status of MOM bit 0 | 1 | MOM bit is cleared (0). 1 | 0 | MOM bit is set (1). Others | No change in MOM bit value.
13, 5	ST_SSHT, CL_SSHT	Sets/clears the SSHT bit. ST_SSHT | CL_SSHT | Status of SSHT bit 0 | 1 | SSHT bit is cleared (0). 1 | 0 | SSHT bit is set (1). Others | No change in SSHT bit value.
12, 4	ST_PPB, CL_PPB	Sets/clears the PPB bit. ST_PPB | CL_PPB | Status of PPB bit 0 | 1 | PPB bit is cleared (0). 1 | 0 | PPB bit is set (1). Others | No change in PPB bit value.
3	CL_BERR	Clears the BERR bit. 0: No change of BERR bit. 1: BERR bit is cleared (0).
2	CL_VALID	Clears the VALID bit. 0: No change of VALID bit. 1: VALID bit is cleared (0).
1	CL_WAKE	Clears the WAKE bit. 0: No change of WAKE bit. 1: WAKE bit is cleared (0).
0	CL_OVR	Clears the OVR bit. 0: No change of OVR bit. 1: OVR bit is cleared (0).

(4) CAN 1 to 2 information registers (C1LAST to C2LAST)

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 2).
These registers can be read-only in 8-bit and 16-bit units.

Figure 11-33: 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	454H	00FFH
C2LAST	0	0	0	0	LERR3	LERR2	LERR1	LERR0	LREC7	LREC6	LREC5	LREC4	LREC3	LREC2	LREC1	LREC0	494H	00FFH

Bit Position	Bit Name	Function
11 to 8	LERR3 to LERR0	Holds the code of the last CAN protocol error.

LERR3	LERR2	LERR1	LERR0	Code of Last CAN Protocol Error
0	0	0	0	No error (reset state only)
0	0	0	1	CAN bus bit error
0	0	1	0	CAN bus stuff error
0	0	1	1	CAN bus CRC error
0	1	0	0	CAN bus form error
0	1	0	1	CAN bus acknowledgement error
0	1	1	0	CAN bus arbitration lost [Note 2]
0	1	1	1	CAN module overrun error
1	0	0	0	CAN wake-up from CAN bus
Others than above				Reserved

Remark: The LERR3 to LERR0 bits cannot be cleared. Thus these bits remain unchanged until the next error occurs.

Bit Position	Bit Name	Function
7 to 0	LREC7 to LREC0	Holds the message buffer number of the last received message.

LREC7 to LREC0	Receive Message Buffer Number
0 to 31	Message buffer number of the last received message
32 to 255	Reserved (not possible)

Notes: 1. The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

2. This error code only occurs in single-shot mode (SSHT bit of the CxDEF register = 1).

(5) CAN 1 to 2 error counter registers (C1ERC to C2ERC)

The CxERC registers reflect the status of the transmit and the receive error counters of the corresponding CAN module x (x = 1 to 2).
These registers can be read-only in 8-bit and 16-bit units.

Figure 11-34: CAN 1 to 2 Error Counter Registers (C1ERC to C2ERC)

	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	456H	0000H
C2ERC	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	496H	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 = FCAN_BASE + address offset

(6) CAN 1 to 2 interrupt enable registers (C1IE to C2IE)

The CxIE registers enable the transmit, receive and error interrupts of the corresponding CAN module x (x = 1 to 2).

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 11.3.1 ”Bit set/clear function” on page 343).

Figure 11-35: CAN 1 to 2 Interrupt Enable Registers (C1IE to C2IE) (1/4)

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	458H	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	498H	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	458H
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	498H

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Figure 11-35: CAN 1 to 2 Interrupt Enable Registers (C1IE to C2IE) (2/4)

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 11-35: CAN 1 to 2 Interrupt Enable Registers (C1IE to C2IE) (3/4)

Write (1/2)

Bit Position	Bit Name	Function		
14, 6	ST_E_INT6, CL_E_INT6	Sets/clears the E_INT6 bit.		
		ST_E_INT6	CL_E_INT6	Status of E_INT6 bit
		0	1	E_INT6 bit is cleared (0).
		1	0	E_INT6 bit is set (1).
		Others		No change in E_INT6 bit value.
13, 5	ST_E_INT5, CL_E_INT5	Sets/clears the E_INT5 bit.		
		ST_E_INT5	CL_E_INT5	Status of E_INT5 bit
		0	1	E_INT5 bit is cleared (0).
		1	0	E_INT5 bit is set (1).
		Others		No change in E_INT5 bit value.
12, 4	ST_E_INT4, CL_E_INT4	Sets/clears the E_INT4 bit.		
		ST_E_INT4	CL_E_INT4	Status of E_INT4 bit
		0	1	E_INT4 bit is cleared (0).
		1	0	E_INT4 bit is set (1).
		Others		No change in E_INT4 bit value.
11, 3	ST_E_INT3, CL_E_INT3	Sets/clears the E_INT3 bit.		
		ST_E_INT3	CL_E_INT3	Status of E_INT3 bit
		0	1	E_INT3 bit is cleared (0).
		1	0	E_INT3 bit is set (1).
		Others		No change in E_INT3 bit value.
10, 2	ST_E_INT2, CL_E_INT2	Sets/clears the E_INT2 bit.		
		ST_E_INT2	CL_E_INT2	Status of E_INT2 bit
		0	1	E_INT2 bit is cleared (0).
		1	0	E_INT2 bit is set (1).
		Others		No change in E_INT2 bit value.
9, 1	ST_E_INT1, CL_E_INT1	Sets/clears the E_INT1 bit.		
		ST_E_INT1	CL_E_INT1	Status of E_INT1 bit
		0	1	E_INT1 bit is cleared (0).
		1	0	E_INT1 bit is set (1).
		Others		No change in E_INT1 bit value.

Figure 11-35: CAN 1 to 2 Interrupt Enable Registers (C1IE to C2IE)(4/4)

Write (2/2)

Bit Position	Bit Name	Function
8, 0	ST_E_INT0, CL_E_INT0	Sets/clears the E_INT0 bit.

ST_E_INT0	CL_E_INT0	Status of E_INT0 bit
0	1	E_INT0 bit is cleared (0).
1	0	E_INT0 bit is set (1).
Others		No change in E_INT0 bit value.

(7) CAN 1o 2 bus activity registers (C1BA to C2BA)

The CxBA registers indicate the status of the CAN bus activities of the corresponding CAN module x (x = 1 to 2).
These registers can be read-only in 8-bit and 16-bit units.

Figure 11-36: CAN 1 to 2 Bus Activity Registers (C1BA to C2BA) (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	45AH	00FFH
C2BA	0	0	0	CACT4	CACT3	CACT2	CACT1	CACT0	TMNO7	TMNO6	TMNO5	TMNO4	TMNO3	TMNO2	TMNO1	TMNO0	49AH	00FFH

Bit Position	Bit Name	Function
12 to 8	CACT4 to CACT0	Indicates the CAN module activity.

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

Remark: The CACT4 to CACT0 bits reflect the status of the CAN protocol layer.

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Figure 11-36: CAN 1 to 4 Bus Activity Registers (C1BA to C4BA) (2/2)

Bit Position	Bit Name	Function
7 to 0	TMNO7 to TMNO0	Indicates the message buffer, which is either waiting to be transmitted or in transmission progress.

TMNO7 to TMNO0	Number of Transmit Message Buffer
0 to 31	Current transmit message buffer (waiting for transmission, or in transmission progress)
32 to 254	Reserved (not possible)
255	No message waiting for transmission, or in transmission progress.

(8) CAN 1 to 2 bit rate prescaler registers (C1BRP to C2BRP)

The CxBRP registers specify the bit rate prescaler and CAN bus speed of the corresponding CAN module x (x = 1 to 2).
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 11-37: 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	45CH	0000H
C2BRP	TLM	0	0	0	0	0	0	0	0	BTYPE	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	49CH	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	45CH	
C2BRP	TLM	0	0	0	0	0	0	BTYPE	BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	49CH	

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 = FCAN_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 11-37: CAN 1 to 2 Bit Rate Prescaler Registers (C1BRP to C2BRP) (2/2)

Bit Position	Bit Name	Function
7 to 0 (TLM = 1) 5 to 0 (TLM = 0)	BRP7 to BRP0 (TLM = 1) BRP5 to BRP0 (TLM = 0)	Specifies the bit rate prescaler for the CAN protocol layer.

TLM = 0:

BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Bit Rate Prescaler $f_{BTL} = f_{MEM} / 2\times(k+1)$	k
0	0	0	0	0	0	$f_{BTL} = f_{MEM} / 2$	0
0	0	0	0	0	1	$f_{BTL} = f_{MEM} / 4$	1
0	0	0	0	1	0	$f_{BTL} = f_{MEM} / 6$	2
0	0	0	0	1	1	$f_{BTL} = f_{MEM} / 8$	3
0	0	0	1	0	0	$f_{BTL} = f_{MEM} / 10$	4
. . .						. . .	. . .
1	1	1	1	1	0	$f_{BTL} = f_{MEM} / 126$	62
1	1	1	1	1	1	$f_{BTL} = f_{MEM} / 128$	63

TLM = 1:

BRP7	BRP6	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	Bit Rate Prescaler $f_{BTL} = f_{MEM} / (k+1)$	k
0	0	0	0	0	0	0	0	$f_{BTL} = f_{MEM}$	0
0	0	0	0	0	0	0	1	$f_{BTL} = f_{MEM} / 2$	1
0	0	0	0	0	0	1	0	$f_{BTL} = f_{MEM} / 3$	2
0	0	0	0	0	0	1	1	$f_{BTL} = f_{MEM} / 4$	3
0	0	0	0	0	1	0	0	$f_{BTL} = f_{MEM} / 5$	4
. . .								. . .	. . .
1	1	1	1	1	1	1	0	$f_{BTL} = f_{MEM} / 255$	254
1	1	1	1	1	1	1	1	$f_{BTL} = f_{MEM} / 256$	255

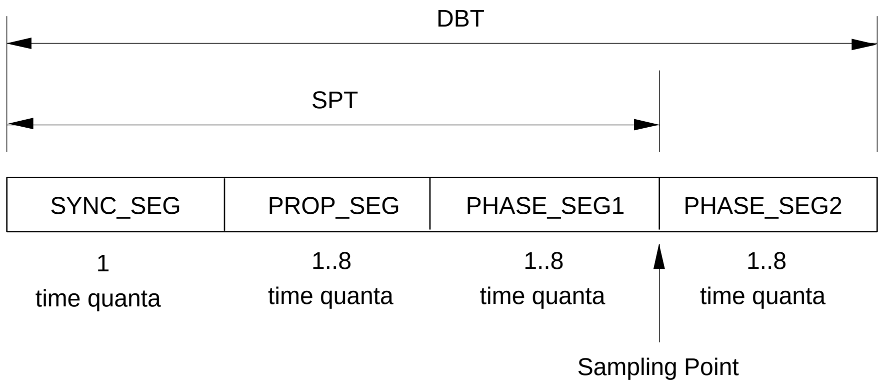
Remark: The BRP defines the period of the base clock f_{BTL} for the protocol layer of a CAN module. It determines the number of FCAN system clocks f_{MEM} per time quantum TQ. A time quantum TQ is the basic unit of a bit in a CAN frame:

$$TQ = \frac{1}{f_{BTL}}$$

(9) CAN 1 to 2 synchronization control registers (C1SYNC to C2SYNC)

A bit in a CAN frame is built by a programmable number of time quanta (TQ), as shown in the Figure 11-38 below.

Figure 11-38: 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 2)

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 2).
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 11-39: CAN 1 to 2 Synchronization Control Registers (C1SYNC to C2SYNC) (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	45EH	0218H
C2SYNC	0	0	0	SAMP	SJWR1	SJWR0	SPTR4	SPTR3	SPTR2	SPTR1	SPTR0	DBTR4	DBTR3	DBTR2	DBTR1	DBTR0	49EH	0218H

Bit Position	Bit Name	Function
12	SAMP	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.
11, 10	SJWR1, SJWR0	Specifies the synchronization jump width.
9 to 5	SPTR4 to SPTR0	Specifies the sampling point position.

SJWR1	SJWR0	Synchronization Jump Width
0	0	1 TQ
0	1	2 TQ
1	0	3 TQ
1	1	4 TQ

SPTR4	SPTR3	SPTR2	SPTR1	SPTR0	Sampling Point Position SPTR = (p + 1) TQ	p
0	0	0	0	0	Setting prohibited	
0	0	0	0	1		
0	0	0	1	0	3 TQ	2
0	0	0	1	1	4 TQ	3
0	0	1	0	0	5 TQ	4
		⋮			⋮	⋮
1	0	0	0	0	17 TQ	16
Other than above					Setting prohibited	

Note: The register address is calculated according to the following formula:
effective address = FCAN_BASE + address offset

Figure 11-39: CAN 1 to 2 Synchronization Control Registers (C1SYNC to C2SYNC) (2/2)

Bit Position	Bit Name	Function
4 to 0	DBTR4 to DBTR0	Specifies the number of TQ per bit.

DBTR4	DBTR3	DBTR2	DBTR1	DBTR0	Data Bit Time DBTR = (q + 1) TQ	q
0	0	0	0	0	Setting prohibited	–
		⋮				
0	0	1	0	1		
0	0	1	1	1	8 TQ	7
0	1	0	0	0	9 TQ	8
0	1	0	0	1	10 TQ	9
		⋮			⋮	⋮
1	1	0	0	0	25 TQ	24
Other than above					Setting prohibited	

Remarks: 1. CPU can read the CxSYNC register at any time (x = 1 to 2)

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 11.4 ”Operating Considerations” on page 400).

(10) CAN 1 to 2 bus diagnostic information registers (C1DINF to C2DINF)

The CxDINF registers reflect the last transmission on CAN bus of the corresponding CAN module x (x = 1 to 2).
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 11-40: 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	45CH	0000H
C2DINF	DINF15	DINF14	DINF13	DINF12	DINF11	DINF10	DINF9	DINF8	DINF7	DINF6	DINF5	DINF4	DINF3	DINF2	DINF1	DINF0	49CH	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.

11.4 Operating Considerations

11.4.1 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 2):

$$3\text{ TQ} \le SPT = (p+1)\text{ TQ} \le 17\text{ TQ}$$

$$2 \le p \le 16$$

$$p = \text{decimal value of bits SPTR4 to SPTR0}$$

(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:

$$8\text{ TQ} \le DBTR = (q+1)\text{ TQ} \le 25\text{ TQ}$$

$$7 \le q \le 24$$

$$q = \text{decimal value of bits DBTR4 to DBTR0}$$

(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.

11.4.2 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:

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)

11.4.3 Ensuring data consistency

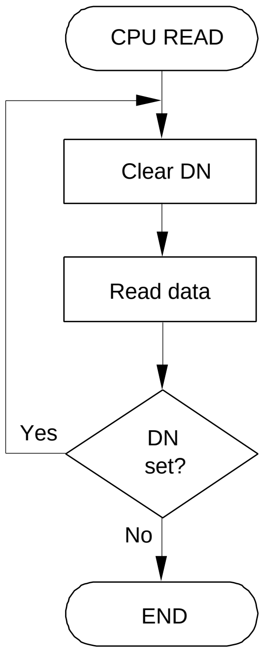
If the CPU reads data from the CAN message buffers, the consistency of data read has to be ensured. Therefore two mechanisms are provided:

- 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 11-41: 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.

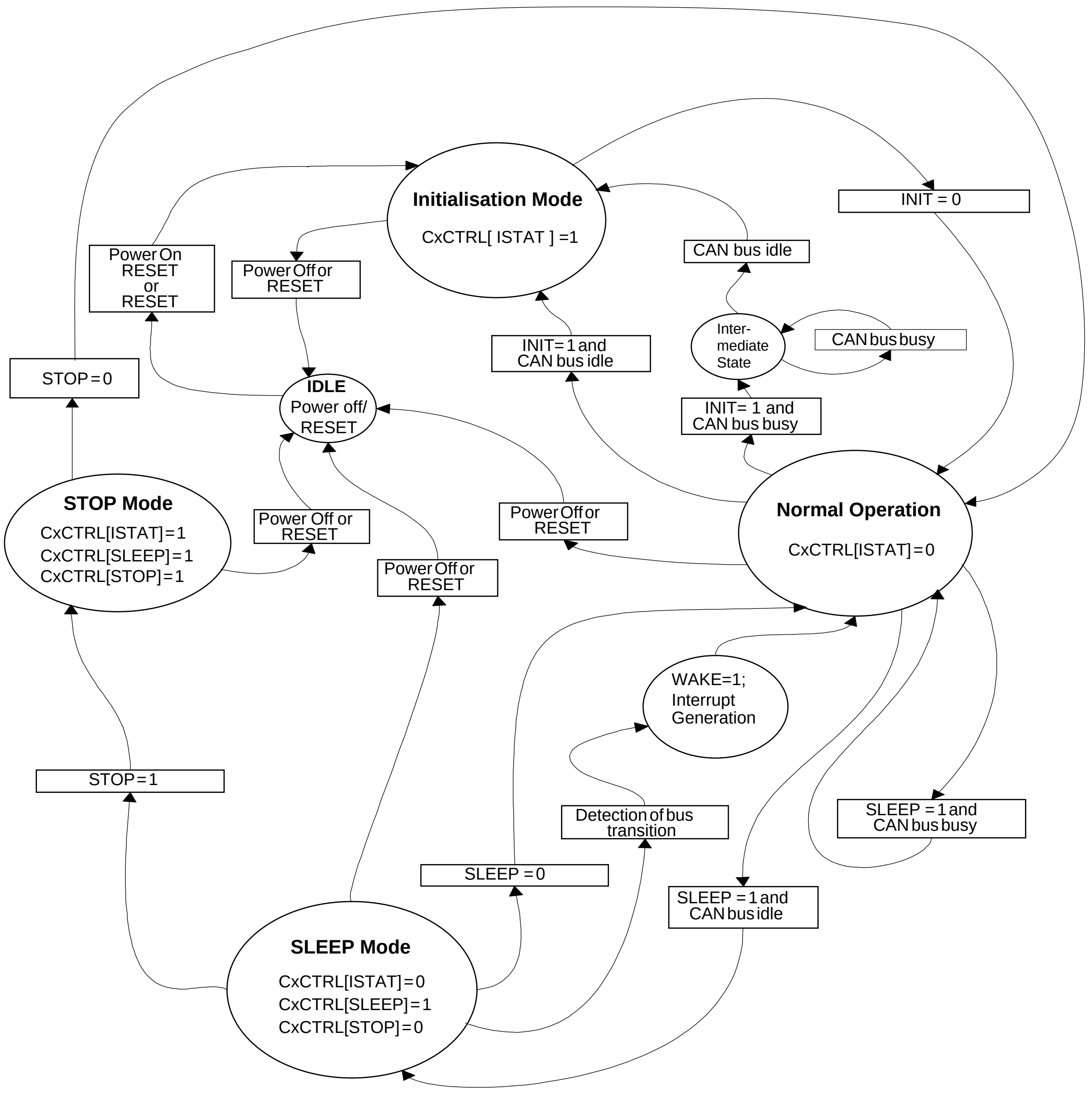
The copy of the message is automatically started whenever the data length code from the M_DLCm register is read by the CPU, and the data is copied from the message buffer into the temporary buffer. As long as the CPU reads 16-bit data from consecutive addresses (that means 16 -bit burst read sequence 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.

11.4.4 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 11-42.

*Figure 11-42: State Transition Diagram for CAN Module*s



Remark: x = 1 to 2.

11.4.5 Initialisation routines

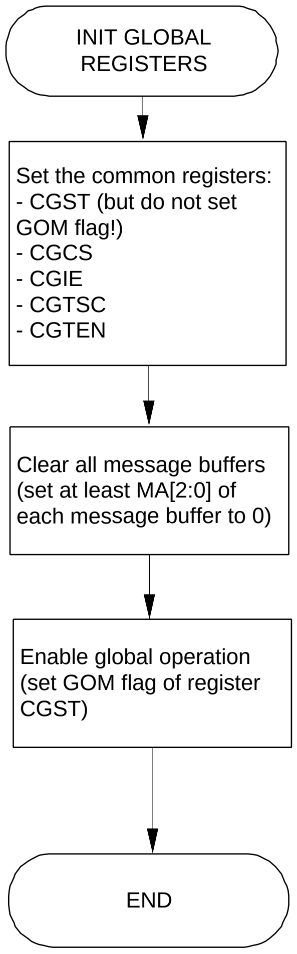
Below the necessary steps for correct start-up of the CAN interface are explained.

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 11-43.

Figure 11-43: General Initialisation Sequence for the CAN Interface


INIT GLOBAL REGISTERS
Set the common registers:
- CGST (but do not set GOM flag!)
- CGCS
- CGIE
- CGTSC
- CGTEN
Clear all message buffers (set at least MA[2:0] of each message buffer to 0)
Enable global operation (set GOM flag of register CGST)
END

Remark: Enabling the global operation does not automatically enable any CAN module. Each CAN module must be initialized and enabled separately.

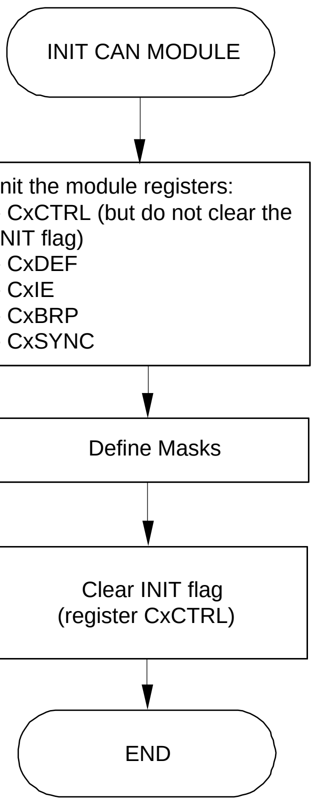
Example for C routine:

```
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 initialized by the sequence according to Figure 11-44.

Figure 11-44: Initialisation Sequence for a CAN module



Remark: x = 1 to 2.

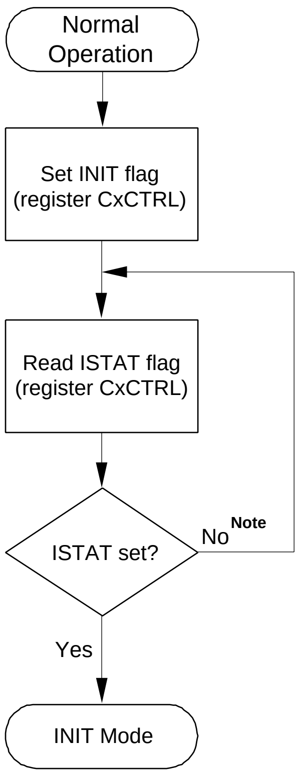
Example for C routine:

```
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 11-45: Setting CAN Module into Initialisation State


Normal Operation
Set INIT flag (register CxCTRL)
Read ISTAT flag (register CxCTRL)
ISTAT set?
No Note
Yes
INIT Mode

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 2.

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 2).

<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 12 A/D Converter

12.1 Features

- 10-bit resolution on-chip A/D converter
- Conversion time: 5 µsec.
- Analog inputs: 14 channels
- Separate on-chip A/D conversion result registers for each analog input
  - 10 bits × 14 registers
- A/D conversion trigger modes
  - A/D trigger mode
  - polling mode
- A/D conversion operating modes
  - select mode
  - scan mode
- Successive approximation technique
- Power fail detection feature
  - The result of the A/D conversion of one of the 14 channels can be selected for periodical comparison with the value of the A/D voltage detection mode register. Depending on the result an interrupt can generated either on undergoing the threshold or on surpassing the threshold.

12.2 Configuration

The A/D converter, which employs a successive approximation technique, performs A/D conversion operation using A/D scan mode registers 0 and 1 (ADSCM0, ADSCM1) and A/D conversion result registers ADCRm (m = 0 to 13).

(1) Input circuit

The input circuit selects an analog input (ANIm) according to the mode set in the ADSCM0 register and sends it to the sample and hold circuit (m = 0 to 13).

(2) Sample and hold circuit

The sample and hold circuit individually samples analog inputs sent sequentially from the input circuit and sends them to the comparator. It holds sampled analog inputs 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 an A/D conversion result register (ADCRm) (m = 0 to 13). When all specified A/D conversions terminate, there also is an A/D conversion termination interrupt (INTAD).

(6) A/D conversion result registers n (ADCRn) (n = 0 to 13)

ADCRn are 10-bit registers that hold A/D conversion results (n = 0 to 13). Whenever an A/D conversion terminates, the conversion result from the successive approximation register (SAR) is loaded.
When $\overline{\text{RESET}}$ is input, the contents of ADCRn are undefined.

(7) Controller

The controller selects an analog input, generates sample and hold circuit operation timing, controls conversion triggers, specifies the conversion operation time, and sets the low power consumption mode according to the mode set in the ADSCM0 or ADSCM1 register.

(8) ANIm pins (m = 0 to 13)

The ANIm pins are the 14-channel analog input pins to analog converter.

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.

(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 13). If not using the AV_{REF} pin, connect it to AV_{DD} or AV_{SS}**Note**.

Note: When connecting the AV_{REF} pin to AV_{SS} the power consumption will be reduced.

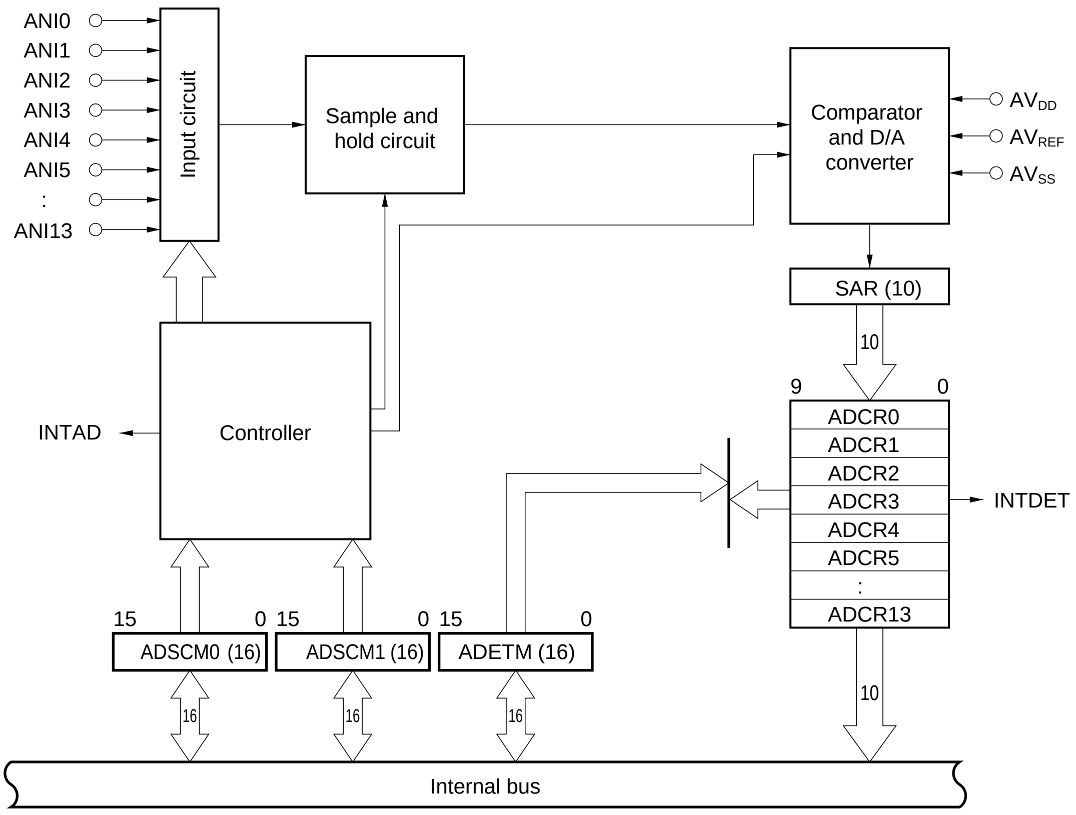
(10) AV_{SS} pin

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_{SS} 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_{DD} pin.

Figure 12-1: 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.
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.**

2. 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.

12.3 Control Registers

(1) A/D scan mode register 0 (ADSCM0)

The ADSCM0 register is a 16-bit register that selects analog input pins, specifies operation and trigger modes, and controls conversion operation.
It can be read or written in 1-bit, 8-bit or 16-bit units. However, writing to the ADSCM0 register during A/D conversion operation interrupts the conversion operation and the data is lost. The conversion operation restarts as specified.
When using 16-bit access, use ADSCM0.
When using 8-bit access, use ADSCM0L for the low-order byte, and ADSCM0H for the high-order byte.
When using 1-bit access, use the bit name.

Figure 12-2: A/D Scan Mode Register 0 (ADSCM0) (1/2)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADSCM0	CE	CS	AVREF_CUT	MS	PLM	0	0	0	SANI3	SANI2	SANI1	SANI0	ANIS3	ANIS2	ANIS1	ANIS0	FFFFF200H	0000H

Bit position	Bit name	Function
15	CE	Specifies enabling or disabling A/D conversion. 0: Disable 1: Enable
14	CS	Shows status of A/D converter. This bit is read-only. 0: Stopped 1: Operating
13	AVREF_CUT	Specifies supply of R2R network and comparator 0: powered 1: not powered
12	MS	Specifies operation mode of A/D converter. 0: Scan mode 1: Select mode
11	PLM	PLM: Specifies polling mode as trigger mode 0: trigger mode is A/D trigger mode 1: trigger mode is polling mode

Figure 12-2: A/D Scan Mode Register 0 (ADSCM0) (2/2)

Bit position	Bit name	Function
7 to 4	SANI3 to SANI0	The bits SANI3 to SANI0 specify the analog input pin mode for which the 1st conversion is performed in scan mode. These bits are ignored in select mode. (see table below) **Caution: Always set the conversion start analog input pin number that is set by bits SANI3-SANI0 to a smaller pin number than the conversion end analog input pin number that is set by bits ANIS3-ANIS0.**
3 to 0	ANIS3 to ANIS0	ANIS3 to ANIS0 specifies the analog input pin in select mode. In scan mode mode, it specifies the last analog input pin for which a conversion is issued. The range of consecutive conversions is defined by the setting of SANI3 to SANI0 and ANIS3 to ANIS0, which lead to a number of conversions defined by: n= ANIS3 to ANIS0 - SANI3 to SANI0 + 1. (see table below) **Note:** Setting for SANI3 to SANI0 has to be set to 0 to start the scan mode at analog input pin ANI0. **Remark:** SANI[3-0] < ANI[3-0]

Bits 7 to 4 (SANI3 to SANI0):

SANI3	SANI2	SANI1	SANI0	Scan Start Analog Input Pin
0	0	0	0	ANI0
0	0	0	1	ANI1
0	0	1	0	ANI2
0	0	1	1	ANI3
0	1	0	0	ANI4
0	1	0	1	ANI5
0	1	1	0	ANI6
0	1	1	1	ANI7
1	0	0	0	ANI8
1	0	0	1	ANI9
1	0	1	0	ANI10
1	0	1	1	ANI11
1	1	0	0	ANI12
1	1	0	1	ANI13
Other than above				Setting prohibited

Caution: Always set the conversion start analog input pin number that is set by bits SANI3-SANI0 to a smaller pin number than the conversion end analog input pin number that is set by bits ANIS3-ANIS0.

Bits 3 to 0 (ANIS3 to ANIS0):

ANIS3	ANIS2	ANIS1	ANIS0	in Select Mode	In Scan Mode
0	0	0	0	ANI0	ANI0 [Note]
0	0	0	1	ANI1	ANI0 [Note] → ANI1
0	0	1	0	ANI2	SANI[3-0] → ANI2
0	0	1	1	ANI3	SANI[3-0] → ANI3
0	1	0	0	ANI4	SANI[3-0] → ANI4
0	1	0	1	ANI5	SANI[3-0] → ANI5
0	1	1	0	ANI6	SANI[3-0] → ANI6
0	1	1	1	ANI7	SANI[3-0] → ANI7
1	0	0	0	ANI8	SANI[3-0] → ANI8
1	0	0	1	ANI9	SANI[3-0] → ANI9
1	0	1	0	ANI10	SANI[3-0] → ANI10
1	0	1	1	ANI11	SANI[3-0] → ANI11
1	1	0	0	ANI12	SANI[3-0] → ANI12
1	1	0	1	ANI13	SANI[3-0] → ANI13
Other than above				Setting prohibited	

Note: Setting for SANI3 to SANI0 has to be set to 0 to start the scan mode at analog input pin ANI0.

Remark: SANI[3-0] < ANI[3-0]

(2) A/D scan mode register 1 (ADSCM1)

The ADSCM1 register is a 16-bit register that sets the conversion time of the A/D converter.
It can be read or written in 1-bit, 8-bit or 16-bit units. However, when the data is written to the ADSCM1 register during the A/D conversion operations, the conversion has to be stopped.

Figure 12-3: A/D Scan Mode Register 1 (ADSCM1) (1/2)

(a) When using 16-bit access to ADSCM1, use ADSCM1

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADSCM1	ADPS	0	0	0	0	FR2	FR1	FR0	0	0	0	0	0	0	0	0	FFFFF202H	0000H

(b) When using 8-bit access to ADSCM1, use ADSCM1L for the LSB byte and ADSCM1H for the MSB byte

	7	6	5	4	3	2	1	0	Address	Initial value
ADSCM1H	ADPS	0	0	0	0	FR2	FR1	FR0	FFFFF203H	0000H

	7	6	5	4	3	2	1	0	Address	Initial value
ADSCM1L	0	0	0	0	0	0	0	0	FFFFF202H	0000H

Figure 12-3: A/D Scan Mode Register 1 (ADSCM1) (2/2)

Bit Position	Bit Name	Function
7	ADPS	A/D converter power saving [Note] 0: Disable power saving mode 1: Enable power saving mode **Note:** Conversion with ADPS bit set requires stabilization time for analog circuit. Conversion in power saving mode is only allowed in select mode and select mode with polling mode. After conversion, within 5 µs (after post-stabilization time) the analog circuit turns in low power mode again. Writing CE bit must be delayed 5 µs to prevent the analog circuit from illegal action. In select-polling mode stabilization time is required for first conversion only. Conversion request during post-stabilization time is valid but result can not be guaranteed. **Remark:** Power saving functions are only engaged, if this bit is set, after the select or select-polling mode has been activated.
2 to 0	FR2 to FR0	Specifies the conversion time (T_{CONV}). (see table below) **Remark:** f_{CPU}: Internal system clock.

FR2	FR1	FR0	Conversion Clocks (TSTAGE)	Conversion Time [µs]	
				f_{CPU} = 24 MHz	f_{CPU} = 32 MHz
0	0	0	336	14.0	10.375
0	0	1	240	10.0	7.50
0	1	0	168	7.0	5.25
0	1	1	120	5.0	-
1	0	0	96	-	-
1	0	1	72	-	-
1	1	0	48	-	-
1	1	1	Setting prohibited	-	-

Caution: Do not write to the ADSCM1 register during A/D conversion operation. If a write is performed, conversion operation is suspended and subsequently terminates.

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 operation stage can be adjusted. By the selection bits FR2 to FR0 in the ADSCM1 register the SAR compare time T_{CONV} can be set in the range of 120/f_{CPU} to 336/f_{CPU}.

$$T_{CONV} = T_{CONV} / f_{CPU}$$

However, the settings modifying the convertion time T_{CONV} must keep the following relation.

$$T_{CONV} > 5\ \mu s$$

(3) A/D voltage detection mode register (ADETM)

The ADETM register is a 16-bit register that sets voltage detection mode. In voltage detection mode a reference voltage value is compared with the analog input pin for which voltage detection is being performed, and an interrupt is set in response to the comparison result.
This register can be read or written in 1-bit, 8-bit, or 16-bit units.
When using 16-bit access, use ADETM0;
When using 8-bit access, use ADETM0L for the low-order byte, and ADETM0H for the high-order byte.
When using 1-bit access, use the bit name.

Figure 12-4: A/D Voltage Detection Mode Register (ADETM)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADETM	DETEN	DETLH	DET ANI3	DET ANI2	DET ANI1	DET ANI0	DET CMP9	DET CMP8	DET CMP7	DET CMP6	DET CMP5	DET CMP4	DET CMP3	DET CMP2	DET CMP1	DET CMP0	FFFFF204H	0000H

Bit position	Bit name	Function
15	DETEN	Specifies voltage detection mode. 0: Operate in normal mode 1: Operate in voltage detection mode
14	DETLH	Sets voltage comparison detection. 0: Generate INTDET interrupt, if reference voltage value > analog input pin voltage. 1: Generate INTDET interrupt, if reference voltage value < analog input pin voltage.
13 to 10	DETANI3 to DETANI0	Selects analog input pin to compare to reference voltage value set by bits DETCMP9-DETCMP0 when in voltage detection mode.

DETANI3	DETANI2	DETANI1	DETANI0	Voltage Detection Analog Input Pin
0	0	0	0	ANI0
0	0	0	1	ANI1
0	0	1	0	ANI2
0	0	1	1	ANI3
0	1	0	0	ANI4
0	1	0	1	ANI5
0	1	1	0	ANI6
0	1	1	1	ANI7
1	0	0	0	ANI8
1	0	0	1	ANI9
1	0	1	0	ANI10
1	0	1	1	ANI11
1	1	0	0	ANI12
1	1	0	1	ANI13
Other than above				Setting prohibited

Bit position	Bit name	Function
9 to 0	DETCMP9 to DETCMP0	Sets reference voltage value to compare with analog input pin selected by bits DETANI3 - DETANI0.

Caution: Do not write to the an ADETM register during A/D conversion operation. If a write is performed, conversion is suspended and it subsequently terminates. Also, the polling mode needs to be re-engaged by writing the CE and PLM bits of the ADSCM0 register.

(4) A/D conversion result registers 0 to 13 (ADCR0 to ADCR13)

The ADCRm registers are 10-bit registers that hold the results of A/D conversions (m = 0 to 13).
These registers can only be read in 16-bit units.
When reading 10 bits of data of an A/D conversion result from an ADCRm register, only the lower 10 bits are valid and the upper 6 bits always read 0.

Figure 12-5: A/D Conversion Result Registers 0 to 13 (ADCR0 to ADCR13)

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	Address	Initial value
ADCRm	0	0	0	0	0	0	ADCRm9	ADCRm8	ADCRm7	ADCRm6	ADCRm5	ADCRm4	ADCRm3	ADCRm2	ADCRm1	ADCRm0	see Table 14-1	un-defined

Table 12-1: Correspondence between ADCRm (m = 0 to 13) Register Names and Addresses

Register name	Address
ADCR0	FFFFF210H
ADCR1	FFFFF212H
ADCR2	FFFFF214H
ADCR3	FFFFF216H
ADCR4	FFFFF218H
ADCR5	FFFFF21AH
ADCR6	FFFFF21CH
ADCR7	FFFFF21EH
ADCR8	FFFFF220H
ADCR9	FFFFF222H
ADCR10	FFFFF224H
ADCR11	FFFFF226H
ADCR12	FFFFF228H
ADCR13	FFFFF22AH

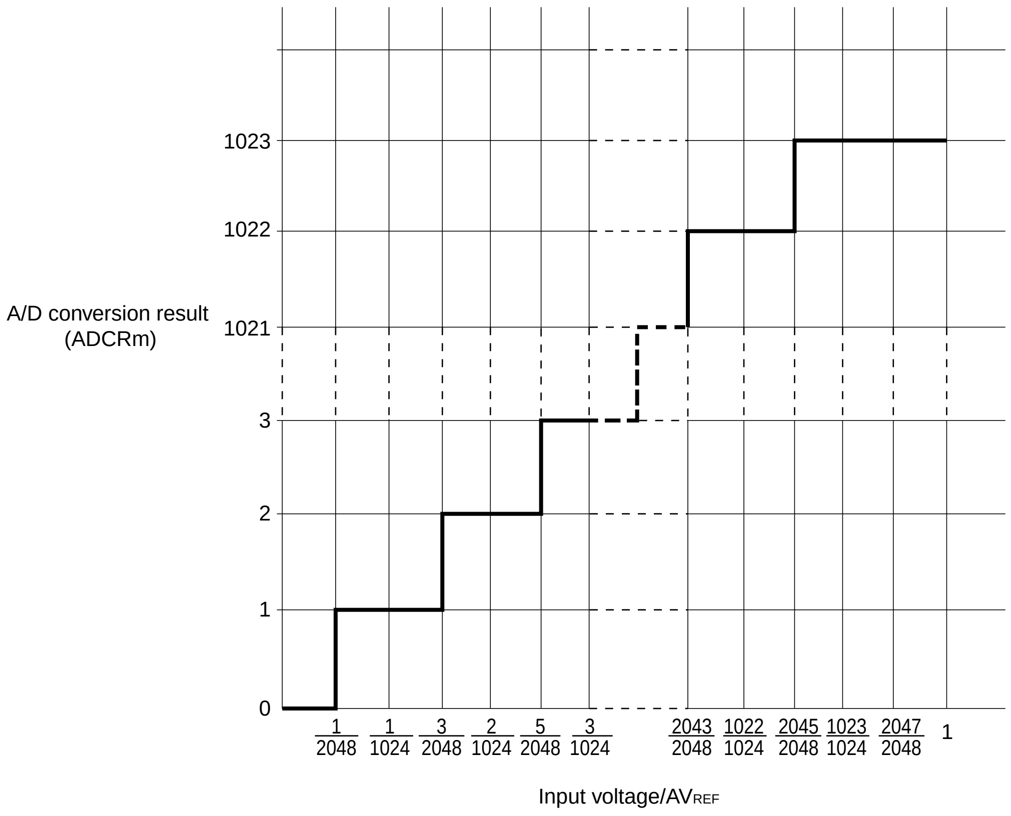
The correspondence between each analog input pin and the ADCRm registers is shown in Table 12-2.

Table 12-2: Correspondence Between each Analog Input Pin and ADCRm Registers

Analog Input Pin	A/D Conversion Result Register
ANI0	ADCR0
ANI1	ADCR1
ANI2	ADCR2
ANI3	ADCR3
ANI4	ADCR4
ANI5	ADCR5
ANI6	ADCR6
ANI7	ADCR7
ANI8	ADCR8
ANI9	ADCR9
ANI10	ADCR10
ANI11	ADCR11
ANI12	ADCR12
ANI13	ADCR13

The following figure shows the relationship between analog input voltage and A/D conversion results.

Figure 12-6: Relationship Between Analog Input Voltages and A/D Conversion Results



Remark: m = 0 to 13

12.4 Interrupt Requests

The A/D converter generates two kinds of interrupts.

- A/D conversion termination interrupt (INTAD)
- Voltage detection interrupt (INTDET)

(1) A/D conversion termination interrupt (INTAD)

In A/D conversion enabled status, an A/D conversion termination interrupt is generated when a specified number of A/D conversions have terminated.

(2) Voltage detection interrupt (INTDET)

In voltage detection mode (DETEN bit of ADETM register = 1), the value of the ADCRm register of the relevant analog input pin is compared to the reference voltage set in the DETCMP[9:0] bits of the ADETM register and a voltage detection interrupt is generated in response to the value of the DETLH bit of the ADETM register (m = 0 to 13).

12.5 A/D Converter Operation

12.5.1 A/D converter basic operation

A/D conversion is performed using the following procedure.

(1) Set the analog input selection and the operation mode and trigger mode specifications using the ADSCM0 **Note**. Setting (1) the CE bit of the ADSCM0 register when in A/D scan mode starts the A/D conversion.

(2) When the A/D conversion starts, the analog input is compared with the voltage generated by the D/A converter.

(3) When the 10-bit comparison terminates, the conversion result is started in the ADCRm register. When the specified number of A/D conversions have terminated, an A/D conversion termination interrupt (INTAD) is generated.

Note: If the contents of the ADSCM0 register are changed during A/D conversion operation, the A/D conversion operation preceding the change stops and a conversion result is not stored in the ADSCRm register. Conversion operation is initialized and conversion starts from the beginning.

Remark: m = 0 to 13

12.5.2 Operation modes and trigger modes

Several conversion operations can be specified for A/D converter by specifying operation modes and trigger modes. Operation modes and trigger modes are set using the ADSCM0 register.
The relationship between operation modes and trigger modes is shown below.

Trigger Mode	Operation Mode	Setting of ADSCM0
AD trigger	Select	×××10000××××××××B
	Scan	×××00000××××××××B
Polling	Select	×××11000××××××××B
	Scan	×××01000××××××××B

(1) Trigger modes

The following trigger modes that serve as the start timing of A/D conversion processing are available: A/D trigger mode and polling mode.
These trigger modes are set using the ADSCM0 register.

(a) A/D trigger mode

In this mode, the A/D conversion is started by setting the CE bit of the register ADSCM0. This starts the conversion timing for the analog input(s) ANIm.
To restart the AD conversion after the INTAD interrupt (conversion finished; CS = 0), the CE bit has to be set again to engage the next conversion.

(b) Polling mode

In this mode, the A/D conversion is started by setting the CE bit of the register ADSCM0. This starts the conversion timing for the analog input(s) ANIm.
A restart the AD conversion after the INTAD interrupt (conversion finished; CS = 1), is not necessary.
The specified analog input is converted serially until the CE bit is set to 0. An INTAD interrupt occurs each time, when a conversion terminates.

Remark: m = 0 to 13

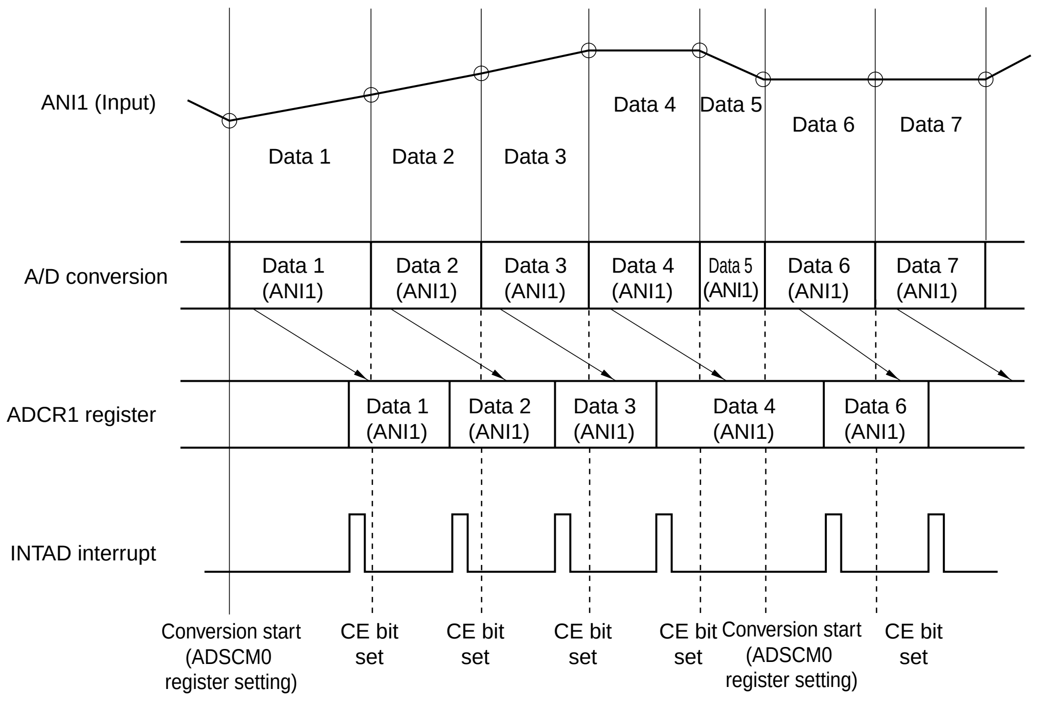
(2) Operation modes

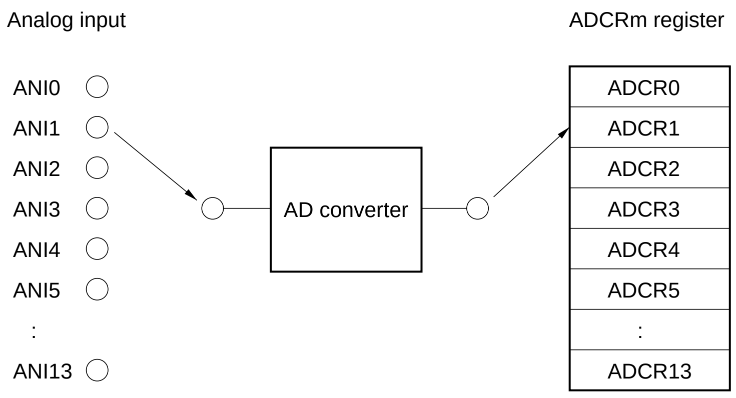
The two operation modes are select mode and scan mode. These modes are set using the ADSCM0 register.

(a) Select mode

In select mode the A/D converts one analog input specified in the ADSCM0 register. The conversion result is stored in the ADCRm register corresponding to the analog input (ANIm) (m = 0 to 13).

Figure 12-7: Example of Select Mode Operation Timing (ANI1)

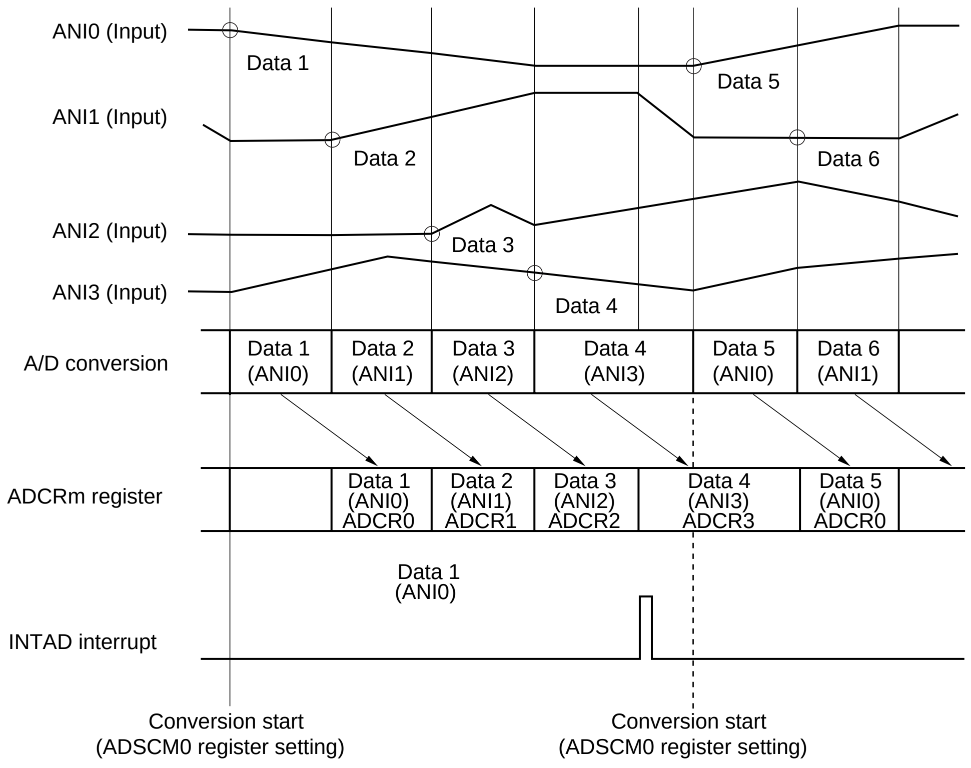


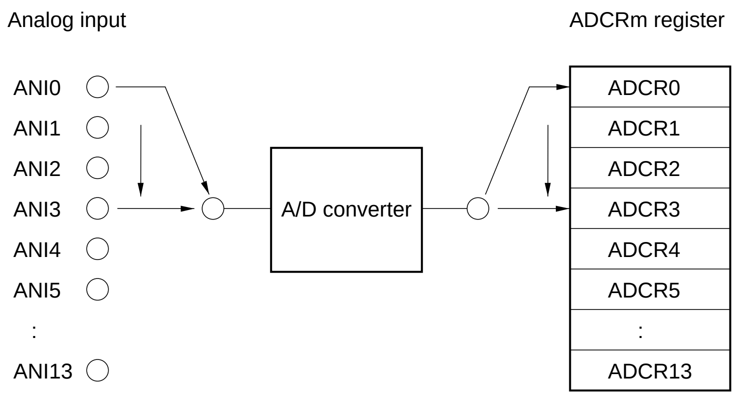


(b) Scan mode

The scan mode sequentially selects and converts pin input voltage from the A/D conversion start analog input pin through the A/D conversion termination analog input pin specified in the ADSCM0 register.
It stores the A/D conversion result in the ADCRm register corresponding to the analog input (m = 0 to 13). When the specified analog input conversion terminates, there is an A/D conversion termination interrupt (INTAD).

Figure 12-8: Example of Scan Mode Operation Timing (4-Channel Scan (ANI0 to ANI3))





12.6 Operation in A/D Trigger Mode

Setting the CE bit of the ADSCM0 register to 1 starts A/D conversion immediately.

12.6.1 Operation in select mode

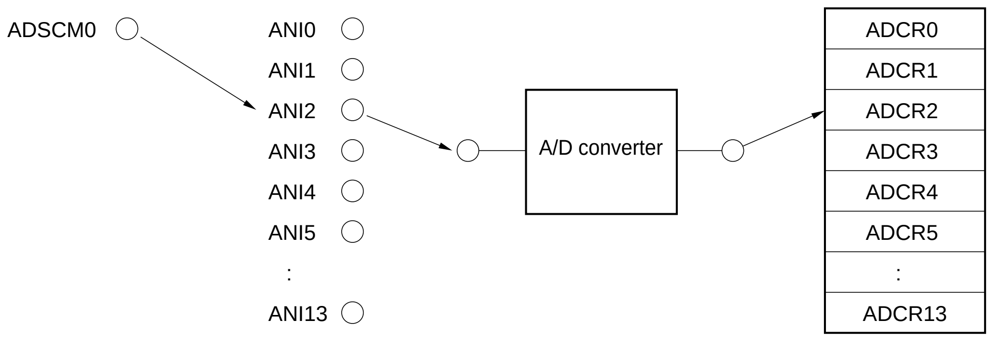
One analog input specified in the ADSCM0 register is A/D converted at a time and the result is stored in an ADCRm register. Analog inputs correspond one-to-one with ADCRm register (m = 0 to 13).
An A/D conversion termination interrupt (INTAD) is generated for each A/D conversion termination (CS bit = 0).

Analog input	A/D conversion result register
ANIm	ADCRm

To restart A/D conversion, set the CE bit of the ADSCM0 register.
This is optimal for an application that reads a result for each A/D conversion.

Remark: m = 0 to 13

Figure 12-9: Example of Select Mode (A/D Trigger Select) Operation (ANI2)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) Generate INTAD interrupt

12.6.2 Operation in scan mode

The pins from the first analog input pin through the last analog input pin, specified in the ADSCM0 register, are sequentially selected and A/D converted. The A/D conversion result is stored in the ADCRm register corresponding to the analog input (m = 0 to 13). When conversion stops through the last analog input pin, an A/D conversion interrupt (INTAD) is generated, which terminates A/D conversion
(CS bit of ADSCM0 register = 0).

Analog input	A/D conversion result register
ANIn**Note 1**	ADCRn
:	:
ANIm**Note 2**	ADCRm

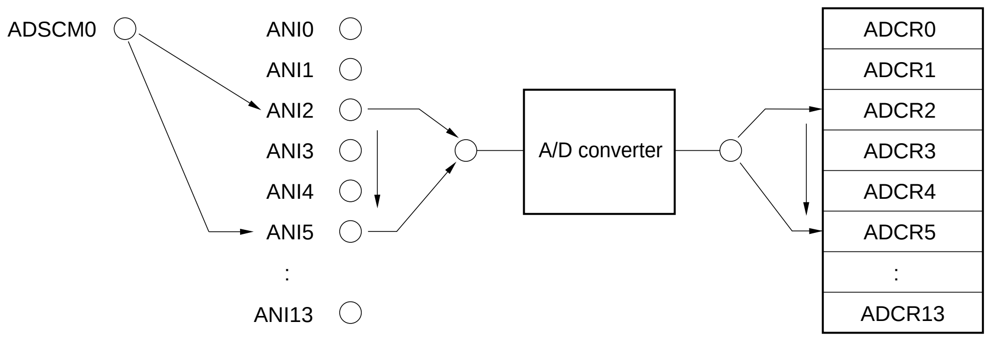
To restart A/D conversion, set the CE bit of the ADSCM0 register. This is optimal for an application that regularly monitors multiple analog inputs.

Notes: 1. n = SANI [3:0] = 0 to 12

2. m = ANIS [3:0] = 1 to 13

Caution: Always set SANI[3:0] < ANIS[3:0]
Exception: When bits SANI[3:0] = ANIS[3:0] = 0, just the analog input ANI0 is scanned.

Figure 12-10: Example of Scan Mode (A/D Trigger Scan) Operation (ANI2-ANI5)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) A/D conversion of ANI3
(5) Store conversion result in ADCR3
(6) A/D conversion of ANI4
(7) Store conversion result in ADCR4
(8) A/D conversion of ANI5
(9) Store conversion result in ADCR5
(10) Generate INTAD interrupt

12.7 Operation in A/D Trigger Polling Mode

Setting the CE bit of the ADSCM0 register to 1 starts the A/D conversion immediately.
Both select mode and scan mode can be continued with A/D trigger polling mode. Since the CS bit of the ADSCM0 register remains 1 after an INTAD interrupt in this mode, it is not necessary to set the CE bit to restart the A/D conversion.

12.7.1 Operation in select mode

The analog input specified in the ADSCM0 register is A/D converted. The conversion result is stored in the ADCRm register (m = 0 to 13).
One analog input is A/D converted at a time and the result is stored in one ADCRm register. Analog inputs correspond one-to-one with ADCRm register.
An A/D conversion termination interrupt (INTAD) is generated for each A/D conversion termination.
A/D conversion operation is repeated until the CE bit = 0 (CS bit = 1).

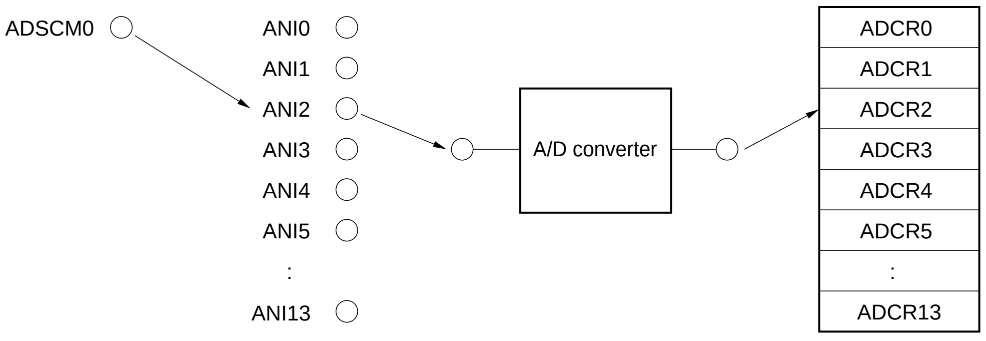
Analog input	A/D conversion result register
ANIm	ADCRm

In A/D trigger polling mode, it is not necessary to set the CE bit of the ADSCM0 register to restart the A/D conversion operation Note.
This is optimal for applications that regularly read A/D conversion values.

Note: If the ADCRm register is not read before the next A/D conversion has been finished, its contents is overwritten.

Remark: m = 0 to 13

Figure 12-11: Example of Select Mode (A/D Trigger Polling Select) Operation (ANI2)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) Generate INTAD interrupt
(5) Return to (2)

12.7.2 Operation in scan mode

The pins from the first analog input pin through the last analog input pin, specified in the ADSCM0 register, are sequentially selected and A/D converted. The A/D conversion result is stored in the ADCRm register corresponding to the analog input (m = 0 to 13). When conversion stops through the last analog input pin, an A/D conversion termination interrupt (INTAD) is generated.
A/D conversion operation repeats until the CE bit = 0 (CS bit = 1).

Analog input	A/D conversion result register
ANIn**Note 1**	ADCRn
:	:
ANIm**Note 2**	ADCRm

It is not necessary to set the CE bit of the ADSCM0 register as the A/D conversion restarts automatically **Note 3**.
This is optimal for applications that regularly read A/D conversion values.

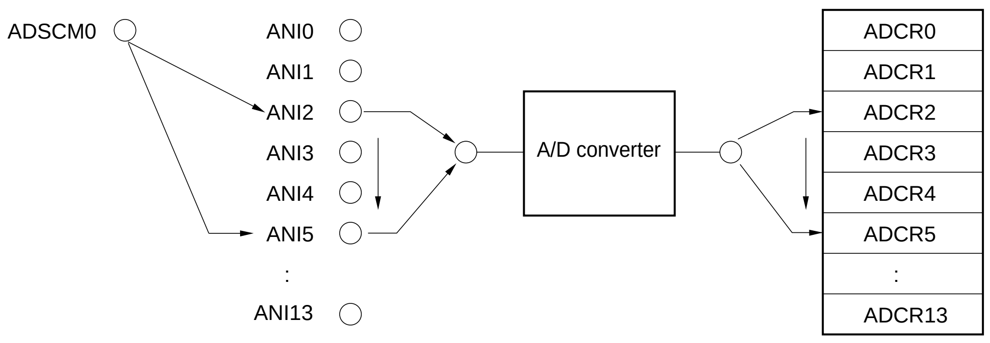
Notes: 1. n = SANI [3:0] = 0 to 12

2. m = ANIS [3:0] = 1 to 13

3. If the ADCRm register is not read before the next A/D conversion has been finished, its contents is overwritten.

Caution: Always set SANI[3:0] < ANIS[3:0]
Exception: When bits SANI[3:0] = ANIS[3:0] = 0, just the analog input ANI0 is scanned.

Figure 12-12: Example of Scan Mode (A/D Trigger Polling Scan) Operation (ANI2 to ANI5)



(1) CE bit of ADSCM0 = 1 (Enabled)
(2) A/D conversion of ANI2
(3) Store conversion result in ADCR2
(4) A/D conversion of ANI3
(5) Store conversion result in ADCR3
(6) A/D conversion of ANI4
(7) Store conversion result in ADCR4
(8) A/D conversion of ANI5
(9) Store conversion result in ADCR5
(10) Generate INTAD interrupt
(11) Return to (2)

12.8 Precautions

12.8.1 Stopping conversion operation

If the CE bit of the ADSCM0 register is cleared during conversion operation, all conversion operations are stopped, and a conversion result is not stored in the ADCRm register (m = 0 to 13).

12.8.2 Operation in standby modes

(1) HALT mode

The A/D converter continues the operation. When recover from HALT mode the ADSCM0, ADSCM1 and ADCRm registers maintain their values (m = 0 to 13).

(2) WATCH mode, IDLE mode, STOP mode

Since clock supply to A/D converter stops, conversion operation is not performed.
If released by $\overline{\text{RESET}}$, NMI, or maskable interrupt input, the ADSCM0, ADSCM1 and ADCRm registers maintain their values.
However, if IDLE mode or STOP mode is set during conversion operation, conversion operation stops. If released by NMI input, conversion resumes but the conversion result written in the ADCRm register becomes undefined (m = 0 to 13).

Remark: Connecting the AV_{REF} pin to AV_{SS} in IDLE mode, or STOP mode will further reduce the power consumption.

[MEMO]

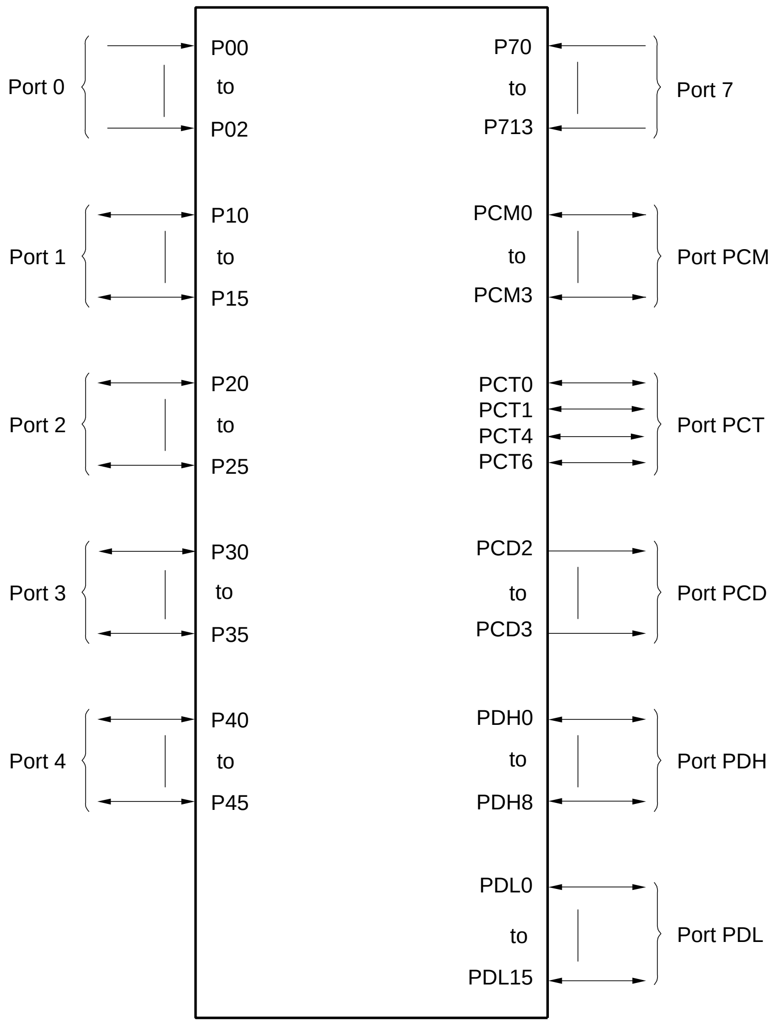
Chapter 13 Port Feature

13.1 Compositions of the port

The ports of the HELIOS have the following features:

- Number of input port: 14
- Number of I/O port: 62
- Multiplexed with I/O pins of other peripheral functions.

Figure 13-1: Ports Block Diagram



13.2 Port 0

Port 0 is a 3-bit input/output port that can be set in the input mode in 1-bit units.
To enable FCTXD1, FCRXD1, PCL, the register PM must be set to the output and PMC must be set to the function mode.

13.2.1 Port 0 register (P0)

This register can be read in 1/8-bit units.

Figure 13-2: Port 0 Register (P0) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
P0	0	0	0	0	0	P02	P01	P00	FFFF F400H	R/W	undefined.
	R	R	R	R	R	R/W	R/W	R/W			

In addition to the function as a general input/output port, this port can be also be used to input external interrupt request.

Port 0	Control mode	Remarks
P00	INTP00/FCRXD1	External interrupt input and serial receive data input for FCAN2
P01	INTP01/FCTXD1	External interrupt input and serial transmit data output for FCAN2
P02	INTP02/PCL	External interrupt input and programma-ble clock output

13.2.2 Port 0 mode register (PM0)

This register can be read or written in 1/8-bit units.

Figure 13-3: Port Mode Register (PM0) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PM0	0	0	0	0	0	PM02	PM01	PM00	FFFF F420H	R/W	0FH
	R	R	R	R	R	R/W	R/W	R/W			

PM0x	Port mode
0	Output port mode (Terminal outputs 0)
1	Input port mode

13.2.3 Port 0 mode control register (PMC0)

This register can be read or written in 1/8-bit units.

Figure 13-4: Port 0 Mode Control Register (PMC0) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMC0	0	0	0	0	0	PM0C2	PMC01	PMC00	FFFF F440H	R/W	00H
	R	R	R	R	R	R/W	R/W	R/W			

PMC00	Port mode
0	Port or external interrupt mode
1	FCRXD1 is enabled (Port, External interrupt is still valid)

PMC01	Port mode
0	Port or external interrupt mode
1	FCTXD1 is enabled (Port, External interrupt is still valid)

PMC02	Port mode
0	Port or external interrupt mode
1	PCL is enabled (Port, External interrupt is still valid)

13.3 Port 1

Port1 is a 6-bit input/output port that can be set in the input or output mode in 1-bit units.

13.3.1 Port 1 register (P1)

This register can be read or written in 1/8-bit units.

Figure 13-5: Port 1 Register (P1) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
P1	0	0	P15	P14	P13	P12	P11	P10	FFFF F402H	R/W	undefined.
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

In addition to the function as a general input/output port, this port can be also be used to input/output for TMG0.

Port 1	Control mode	Remarks
P10	TIG00/INTP10	External interrupt input and Input for TMG0
P11	TIG01/TOG01	Input and output for TMG0
P12	TIG02/TOG02	Input and output for TMG0
P13	TIG03/TOG03/SI10	Input and output for TMG0 and SI10
P14	TIG04/TOG04/SO10	Input and output for TMG0 and SO10
P15	TIG05/INTP15/SCK10	External interrupt input and Input for TMG0 and SCK10

13.3.2 Port 1 mode register (PM1)

This register can be read or written in 1/8-bit units.

Figure 13-6: Port Mode Register (PM1) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PM1	0	0	PM15	PM14	PM13	PM12	PM11	PM10	FFFF F422H	R/W	3FH
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PM1x	Port mode
0	Output port mode
1	Input port mode

13.3.3 Port 1 mode control register (PMC1)

This register can be read or written in 1/8-bit units.

Figure 13-7: Port Mode Control Register (PMC1) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMC1	0	0	PMC15	PMC14	PMC13	PMC12	PMC11	PMC10	FFFF F442H	R/W	00H
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PMC10	Port mode control
0	Input/output port mode
1	External interrupt request input (INTP10), Input for TMG0

PMC11	Port mode control
0	Input/output port mode
1	Input or output for TMG0

PMC12	Port mode control
0	Input/output port mode
1	Input or output for TMG0

PMC13	Port mode control
0	Input/output port mode
1	Input or output for TMG0/SI10

PMC14	Port mode control
0	Input/output port mode
1	Input or output for TMG0/SO10

PMC15	Port mode control
0	Input/output port mode
1	External interrupt request input (INTP15), Input for TMG0. Input or output for SCK10

13.4 Port 2

Port 2 is a 6-bit input/output port that can be set in the input or output mode in 1-bit units.

13.4.1 Port 2 register (P2)

This register can be read or written in 1/8-bit units.

Figure 13-8: Port 2 Register (P2) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
P2	0	0	P25	P24	P23	P22	P21	P20	FFFF F404H	R/W	undefined.
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

In addition to the function as a general input/output port, this port can be also be used to input/output for TMG1.

Port 2	Control mode	Remarks
P20	TIG10/INTP20	External interrupt input and Input for TMG1
P21	TIG11/TOG11	Input and output for TMG1
P22	TIG12/TOG12	Input and output for TMG1
P23	TIG13/TOG13	Input and output for TMG1
P24	TIG14/TOG14	Input and output for TMG1
P25	TIG15/INTP25	External interrupt input and Input for TMG1

13.4.2 Port 2 mode register (PM2)

This register can be read or written in 1/8-bit units.

Figure 13-9: Port Mode Register (PM2) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PM2	0	0	PM25	PM24	PM23	PM22	PM21	PM20	FFFF F424H	R/W	3FH
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PM2x	Port mode
0	Output port mode
1	Input port mode

13.4.3 Port 2 mode control register (PMC2)

This register can be read or written in 1/8-bit units.

Figure 13-10: Port Mode Control Register (PMC2) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMC2	0	0	PMC25	PMC24	PMC23	PMC22	PMC21	PMC20	FFFF F444H	R/W	00H
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PMC20	Port mode control
0	Input/output port mode
1	External interrupt request input (INTP20), Input for TMG1

PMC21	Port mode control
0	Input/output port mode
1	Input or output for TMG1

PMC22	Port mode control
0	Input/output port mode
1	Input or output for TMG1

PMC23	Port mode control
0	Input/output port mode
1	Input or output for TMG1

PMC24	Port mode control
0	Input/output port mode
1	Input or output for TMG1

PMC25	Port mode control
0	Input/output port mode
1	External interrupt request input (INTP25), Input for TMG1

13.5 Port 3

Port 3 is a 6-bit input/output port that can be set in the input or output mode in 1-bit units.

13.5.1 Port 3 register (P3)

This register can be read or written in 1/8-bit units.

Figure 13-11: Port 3 Register (P3) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
P3	0	0	P35	P34	P33	P32	P31	P30	FFFF F406H	R/W	undefined.
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

In addition to the function as a general input/output port, this port can be also be used to input/output for serial interface.

Port 3	Control mode	Remarks
P30	RXD60/INTP30	External interrupt input and serial receive data input for UART60
P31	TXD60	Serial transmit data input for UART60
P32	RXD61/INTP32	External interrupt input and serial receive data input for UART61
P33	TXD61	Serial transmit data input for UART61
P34	FCRXD0/INTP34	External interrupt input and serial receive data input for FCAN1
P35	FCTXD0	Serial transmit data output for FCAN1

13.5.2 Port 3 mode register (PM3)

This register can be read or written in 1/8-bit units.

Figure 13-12: Port Mode Register (PM3) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PM3	0	0	PM35	PM34	PM33	PM32	PM31	PM30	FFFF F426H	R/W	3FH
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PM3x	Port mode
0	Output port mode
1	Input port mode

13.5.3 Port 3 mode control register (PMC3)

This register can be read or written in 1/8-bit units.

Figure 13-13: Port 3 Mode Control Register (PMC3) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMC3	0	0	PMC35	PMC34	PMC33	PMC32	PMC31	PMC30	FFFF F446H	R/W	00H
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PMC30	Port mode control
0	Input/output port mode
1	Serial receive data input for UART60, External interrupt (INTP30)

PMC31	Port mode control
0	Input/output port mode
1	Serial transmit data output for UART60

PMC32	Port mode control
0	Input/output port mode
1	Serial receive data input for UART61, External interrupt (INTP32)

PMC33	Port mode control
0	Input/output port mode
1	Serial transmit data output for UART61

PMC34	Port mode control
0	Input/output port mode
1	Serial receive data input for FCAN1, External interrupt (INTP34)

PMC35	Port mode control
0	Input/output port mode
1	Serial transmit data output for FCAN1

13.6 Port 4

Port 4 is a 6-bit input/output port that can be set in the input or output mode in 1-bit units.

13.6.1 Port 4 register (P4)

This register can be read or written in 1/8-bit units.

Figure 13-14: Port 4 Register (P4) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
P4	0	0	P45	P44	P43	P42	P41	P40	FFFF F408H	R/W	undefined.
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

In addition to the function as a general input/output port, this port can also be used to input/output for serial interface.

Port 4	Control mode	Remarks
P40	SI00	Serial transmit data input(3-wire) for SIO00
P41	SO00	Serial transmit data output(3-wire) for SIO00
P42	$\overline{\text{SCK00}}$	Serial clock I/O(3-wire) for SIO00
P43	SI01	Serial receive data input(3-wire) for SIO01
P44	SO01	Serial transmit data output(3-wire) for SIO01
P45	$\overline{\text{SCK01}}$	Serial clock I/O(3-wire) for SIO01

13.6.2 Port 4 mode register (PM4)

This register can be read or written in 1/8-bit units.

Figure 13-15: Port Mode Register (PM3) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PM4	0	0	PM45	PM44	PM43	PM42	PM41	PM40	FFFF F428H	R/W	3FH
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PM4x	Port mode
0	Output port mode
1	Input port mode

13.6.3 Port 4 mode control register (PMC4)

This register can be read or written in 1/8-bit units.

Figure 13-16: Port 4 Mode Control Register (PMC4) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMC4	0	0	PMC45	PMC44	PMC43	PMC42	PMC41	PMC40	FFFF F448H	R/W	00H
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

PMC40	Port mode control
0	Input/output port mode
1	Serial receive data input (3-wire) for SIO00

PMC41	Port mode control
0	Input/output port mode
1	Serial transmit data output (3-wire) for SIO00

PMC42	Port mode control
0	Input/output port mode
1	Serial clock I/O (3-wire) for SIO00

PMC43	Port mode control
0	Input/output port mode
1	Serial receive data input (3-wire) for SIO01

PMC44	Port mode control
0	Input/output port mode
1	Serial transmit data output (3-wire) for SIO01

PMC45	Port mode control
0	Input/output port mode
1	Serial clock I/O (3-wire) for SIO1

13.7 Port 7

Port 7 is a 14-bit input only port that can be set in the input mode in lower 8-bit unit or upper 6-bit unit or both.

13.7.1 Port 7 register (P7)

This register can be read in 16/8-bit units.
When using 16 bits access, use the name P7.
When using 8 bit access, use P7L for the low-order byte, and P7H for the high-order byte.

Figure 13-17: Port 7 Register (P7) Format

	15	14	13	12	11	10	9	8	Address	R/W	After Reset
P7	0	0	P713	P712	P711	P710	P79	P78	FFFF F40EH	R	undefined.
	R	R	R	R	R	R	R	R			
	7	6	5	4	3	2	1	0			
	P77	P76	P75	P74	P73	P72	P71	P70			
	R	R	R	R	R	R	R	R			

(a) Port 7 register low-order byte (P7L) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
P7L	0	0	P713	P712	P711	P710	P79	P78	FFFF F40EH	R	undefined.
	R	R	R	R	R	R	R	R			

(b) Port 7 register high-order byte (P7H) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
P7H	P77	P76	P75	P74	P73	P72	P71	P70	FFFF F40FH	R	undefined.
	R	R	R	R	R	R	R	R			

Caution: Do not read port 7 During AD conversion.

13.7.2 Port 7 mode control register (PMC7)

This register can be read or written in 1/8-bit units.

Figure 13-18: Port 7 Mode Control Register (PMC7) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMC7	0	0	0	0	0	0	PMC7H	PMC7L	FFFF F44EH	R/W	00H
	R	R	R	R	R	R	R/W	R/W			

PMC7L	Port mode control
0	ANI7- ANI0 are analog input mode
1	ANI7- ANI0 can be used as input port

PMC7H	Port mode control
0	ANI13 - ANI8 are analog input mode
1	ANI13 - ANI8 can be used as input port

Port 7	Control mode	Remarks
P70	ANI0	Analog input to A/D converter
P71	ANI1	Analog input to A/D converter
P72	ANI2	Analog input to A/D converter
P73	ANI3	Analog input to A/D converter
P74	ANI4	Analog input to A/D converter
P75	ANI5	Analog input to A/D converter
P76	ANI6	Analog input to A/D converter
P77	ANI7	Analog input to A/D converter
P78	ANI8	Analog input to A/D converter
P79	ANI9	Analog input to A/D converter
P710	ANI10	Analog input to A/D converter
P711	ANI11	Analog input to A/D converter
P712	ANI12	Analog input to A/D converter
P713	ANI13	Analog input to A/D converter

13.8 Port DL

Port DL is a 16-bit input/output port that can be set in the input or output mode in 1-bit units.

13.8.1 Port DL register (PDL)

This register can be read or written in 1/8/16-bit units.
When using 16 bits access, use the name PDL.
When using 8 bit access, use PDLL for the low-order byte, and PDLH for the high-order byte.
When using 1 bit access, use the bit name.

Figure 13-19: Port DL Register (PDL) Format

	15	14	13	12	11	10	9	8	Address	R/W	After Reset
PDL	PDL15	PDL14	PDL13	PDL12	PDL12	PDL10	PDL9	PDL8	FFFF F004H	R/W	undefined.
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			
	7	6	5	4	3	2	1	0			
	PDL7	PDL6	PDL5	PDL4	PDL3	PDL2	PDL1	PDL0			
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

(a) Port DLL register low-order byte (PDLL) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PDLL	PDL15	PDL14	PDL13	PDL12	PDL12	PDL10	PDL9	PDL8	FFFF F004H	R/W	undefined.
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

(b) Port DL register high-order byte (PDLH) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PDLH	PDL7	PDL6	PDL5	PDL4	PDL3	PDL2	PDL1	PDL0	FFFF F005H	R/W	undefined.
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

13.8.2 Port DL mode register (PMDL)

This register can be read or written in 1/8/16-bit units.
When using 16-bit access, use the name PMDL.
When using 8-bit access, use PMDLL for the low-order byte, and PMDLH for the high-order byte.
When using 1-bit access, use the bit name.

Figure 13-20: Port DL Mode Register (PMDL) Format

	15	14	13	12	11	10	9	8	Address	R/W	After Reset
PMDL	PMDL15	PMDL14	PMDL13	PMDL12	PMDL12	PMDL10	PMDL9	PMDL8	FFFF F024H	R/W	FFFFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			
	7	6	5	4	3	2	1	0			
	PMDL7	PMDL6	PMDL5	PMDL4	PMDL3	PMDL2	PMDL1	PMDL0			
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

(a) Port DL mode register low-order byte (PMDLL) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMDLL	PMDL15	PMDL14	PMDL13	PMDL12	PMDL12	PMDL10	PMDL9	PMDL8	FFFF F024H	R/W	FFFFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

(b) Port DL mode register high-order byte (PMDLH) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMDLH	PMDL7	PMDL6	PMDL5	PMDL4	PMDL3	PMDL2	PMDL1	PMDL0	FFFF F025H	R/W	FFFFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

PMDLx	Port mode
0	Output port mode
1	Input port mode

13.9 Port DH

Port DH is a 9-bit input/output port that can be set in the input or output mode in 1-bit units.

13.9.1 Port DH register (PDH)

This register can be read or written in 1/8/16-bit units.
When using 16-bit access, use the name PDH.
When using 8-bit access, use PDHL for the low-order byte, and PDHH for the high-order byte.
When using 1-bit access, use the bit name.

Figure 13-21: Port DH Register (PDH) Format

	15	14	13	12	11	10	9	8	Address	R/W	After Reset
PDH	0	0	0	0	0	0	0	PDH8	FFFF F006H	R/W	undefined.
	R	R	R	R	R	R	R	R/W			
	7	6	5	4	3	2	1	0			
	PDH7	PDH6	PDH5	PDH4	PDH3	PDH2	PDH1	PDH0			
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

(a) Port DH register low-order byte (PDHL) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PDHL	0	0	0	0	0	0	0	PDH8	FFFF F006H	R/W	undefined.
	R	R	R	R	R	R	R	R/W			

(b) Port DH register high-order byte (PDHH) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PDHH	PDH7	PDH6	PDH5	PDH4	PDH3	PDH2	PDH1	PDH0	FFFF F007H	R/W	undefined.
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

13.9.2 Port DH mode register (PMDH)

This register can be read or written in 1/8/16-bit units.
When using 16-bit access, use the name PMDH.
When using 8-bit access, use PMDHL for the low-order byte, and PMDHH for the high-order byte.
When using 1-bit access, use the bit name.

Figure 13-22: Port DH Mode Register (PMDH) Format

	15	14	13	12	11	10	9	8	Address	R/W	After Reset
PMDH	0	0	0	0	0	0	0	PMDH8	FFFF F026H	R/W	01FFH
	R	R	R	R	R	R	R	R/W			

7	6	5	4	3	2	1	0
PMDH7	PMDH6	PMDH5	PMDH4	PMDH3	PMDH2	PMDH1	PMDH0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

(a) Port DH mode register low-order byte (PMDHL) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMDHL	0	0	0	0	0	0	0	PMDH8	FFFF F026H	R/W	01FFH
	R	R	R	R	R	R	R	R/W			

(b) Port DH mode register high-order byte (PMDHH) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMDHH	PMDH7	PMDH6	PMDH5	PMDH4	PMDH3	PMDH2	PMDH1	PMDH0	FFFF F027H	R/W	01FFH
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W			

PMDLx	Port mode
0	Output port mode
1	Input port mode

13.10 Port CT

Port CT is a 4-bit input/output port that can be set in the input or output mode in 1-bit units.

13.10.1 Port CT register (PCT)

This register can be read or written in 1/8-bit units.

Figure 13-23: Port CT Register (PCT) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PCT	0	PCT6	0	PCT4	0	0	PCT1	PCT0	FFFF F00AH	R/W	undefined.
	R	R/W	R	R/W	R	R	R/W	R/W			

13.10.2 Port CT mode register (PMCT)

This register can be read or written in 1/8-bit units.

Figure 13-24: Port CT Mode Register (PMCT) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMCT	0	PMCT6	0	PMCT4	0	0	PMCT1	PMCT0	FFFF F02AH	R/W	53H
	R	R/W	R	R/W	R	R	R/W	R/W			

PMCTx	Port mode
0	Output port mode
1	Input port mode

13.11 Port CM

Port CM is a 4-bit input/output port that can be set in the input or output mode in 1-bit units.

13.11.1 Port CM register (PCM)

This register can be read or written in 1/8-bit units.

Figure 13-25: Port CM Register (PCM) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PCM	0	0	0	0	PCM3	PCM2	PCM1	PCM0	FFFF F00CH	R/W	undefined
	R	R	R	R	R/W	R/W	R/W	R/W			

13.11.2 Port CM mode register (PMCM)

This register can be read or written in 1/8-bit units.

Figure 13-26: Port CM Mode Register (PMCM) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMCM	0	0	0	0	PMCM3	PMCM2	PMCM1	PMCM0	FFFF F02CH	R/W	0FH
	R	R/W	R	R/W	R/W	R/W	R/W	R/W			

PMCMx	Port mode
0	Output port mode
1	Input port mode

13.12 Port CD

Port CD is a 2-bit input/output port that can be set in the input or output mode in 1-bit units.

13.12.1 Port CD register (PCD)

This register can be read or written in 1/8-bit units.

Figure 13-27: Port CD Register (PCD) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PCD	0	0	0	0	PCD3	PCD2	0	0	FFFF F00EH	R/W	undefined
	R	R/W	R	R	R/W	R/W	R	R			

13.12.2 Port CD mode register (PMCD)

This register can be read or written in 1/8-bit units.

Figure 13-28: Port CD Mode Register (PMCD) Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
PMCD	0	0	0	0	PMCD3	PMCD2	0	0	FFFF F02EH	R/W	0CH
	R	R/W	R	R	R/W	R/W	R	R			

PMCDx	Port mode
0	Output port mode
1	Input port mode

13.13 Port Type

Table 13-1: Port Type (1/3)

Port name	Terminal name	Alternated function	Registers for mode setting	Port block type	R/W
Port 0	P00	INTP00	PM0 PMC0	A	R/W
	P01	INTP01/FCTXD1		A	
	P02	INTP02/PCL		A	
Port 1	P10	TIG00/INTP10	PM1 PMC1	A	R/W
	P11	TIG01/TOG01		A	
	P12	TIG02/TOG02		A	
	P13	TIG03/TOG03/SI10		A	
	P14	TIG04/TOG04/SO10		A	
	P15	TIG05/INTP15/SCK10		A	
Port 2	P20	TIG10/INTP20	PM2 PMC2	A	R/W
	P21	TIG11/TOG01		A	
	P22	TIG12/TOG02		A	
	P23	TIG13/TOG03		A	
	P24	TIG14/TOG04		A	
	P25	TIG15/INTP25		A	
Port 3	P30	RXD60/INTP30	PM3 PMC3	A	R/W
	P31	TXD60		A	
	P32	RXD61/INTP32		A	
	P33	TXD61		A	
	P34	FCRXD0/INTP34		A	
	P35	FCTXD0		A	
Port 4	P40	SI00	PM4 PMC4	A	R/W
	P41	SO00		A	
	P42	SCK00		A	
	P43	SI01		A	
	P44	SO01		A	
	P45	SCK01		A	
Note: Drawing corresponding to port structure does not exist					

Table 13-1: Port Type (2/3)

Port name	Terminal name	Alternated function	Registers for mode setting	Port block type	R/W
Port 7	P70	ANI0	PMC7	B	R
	P71	ANI1		B	
	P72	ANI2		B	
	P73	ANI3		B	
	P74	ANI4		B	
	P75	ANI5		B	
	P76	ANI6		B	
	P77	ANI7		B	
	P78	ANI8		B	
	P79	ANI9		B	
	P710	ANI10		B	
	P711	ANI11		B	
	P712	ANI12		B	
	P713	ANI13		B	
Port DL	PDL0		PMDL	C	R/W
	PDL1			C	
	PDL2			C	
	PDL3			C	
	PDL4			C	
	PDL5			C	
	PDL6			C	
	PDL7			C	
	PDL8			C	
	PDL9			C	
	PDL10			C	
	PDL11			C	
	PDL12			C	
	PDL13			C	
	PDL14			C	
	PDL15			C	
Port DH	PDH0		PMDH	D	R/W
	PDH1			D	
	PDH2			D	
	PDH3			D	
	PDH4			D	
	PDH5			D	
	PDH6			D	
	PDH7			D	
	PDH8			D	
Note: Drawing corresponding to port structure does not exist					

Table 13-1: Port Type (3/3)

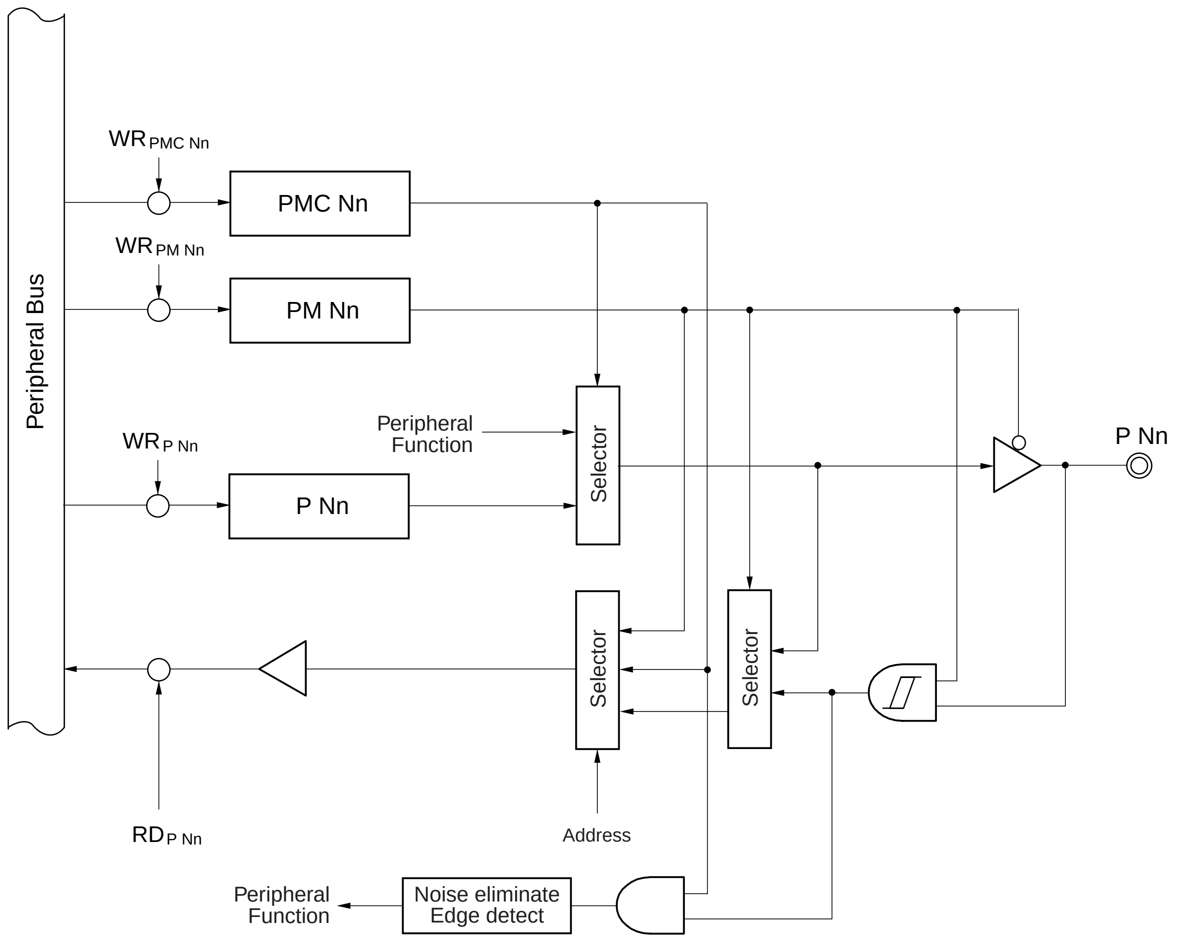
Port name	Terminal name	Alternated function	Registers for mode setting	Port block type	R/W
Port CT	PCT0		PMCT	E	R/W
	PCT1			E	
	PCT4			E	
	PCT6			E	
Port CD	PCD2		PMCD	F	R/W
	PCD3			F	
Port CM	PCM0		PMCM	H	R/W
	PCM1			G	
	PCM2			G	
	PCM3			H	
NMI	NMI	No alternate function	NMIM	I	R

Note: Drawing corresponding to port structure does not exist

13.14 Port Block Type

13.14.1 Port Block Type A

Figure 13-29: Type A Block Diagram

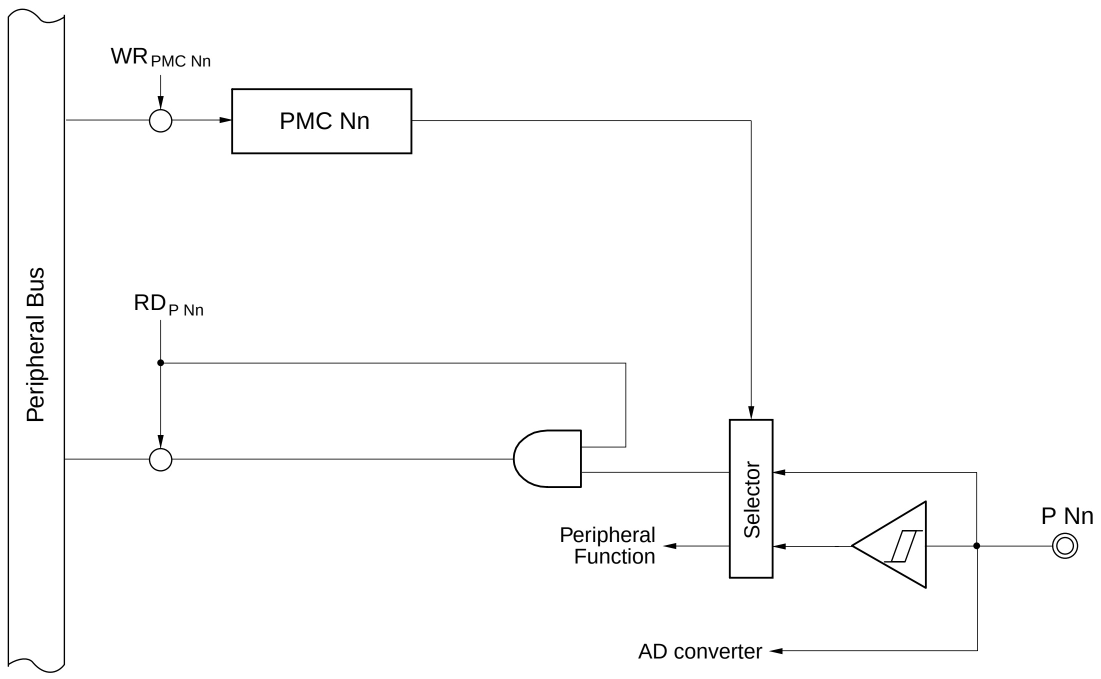


Remarks: 1. N:Port number

2. n:Bit number

13.14.2 Port Block Type B

Figure 13-30: Type B Block Diagram

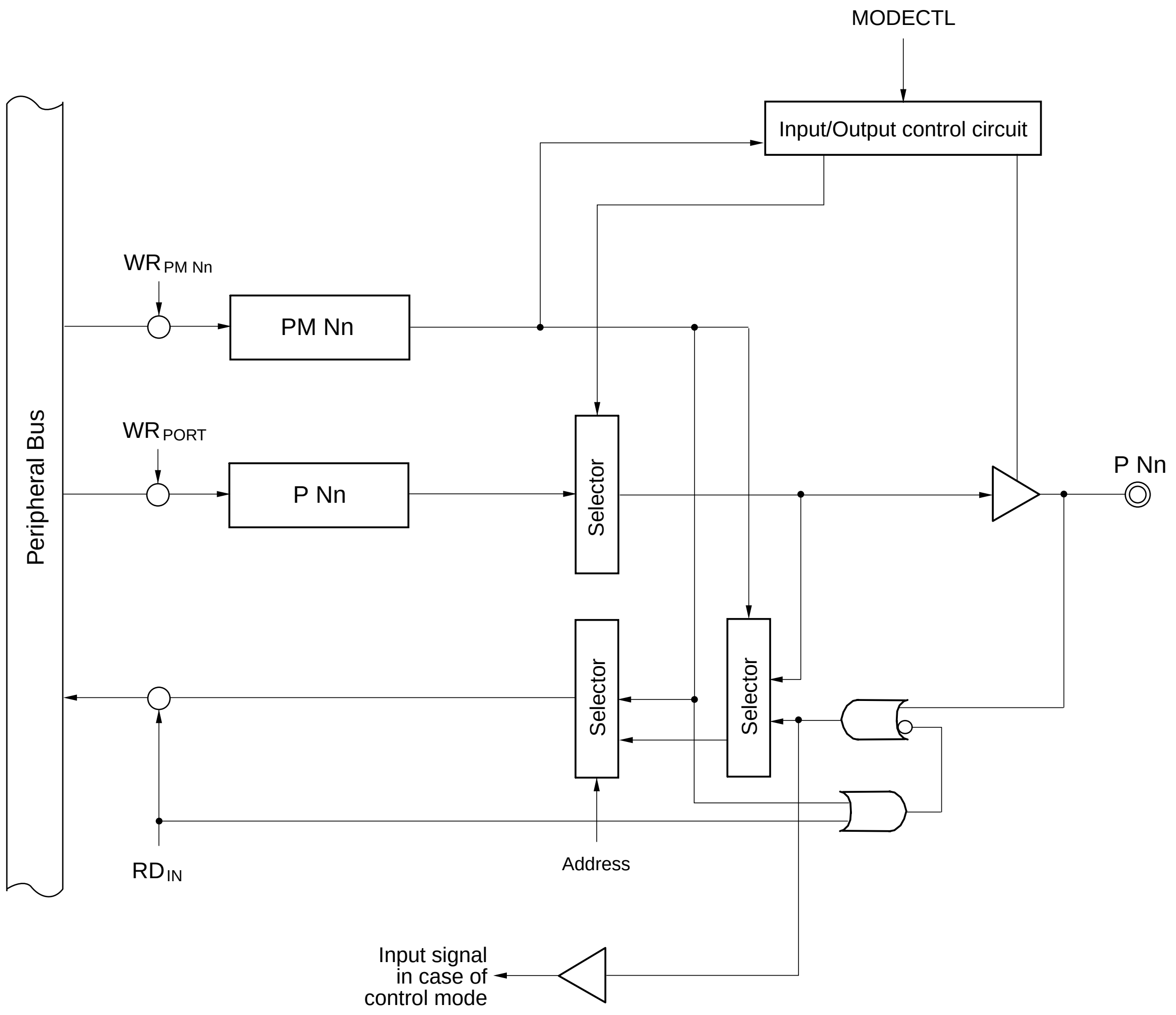


Remarks: **1.** N:Port number

2. n:Bit number

13.14.3 Port Block Type C

Figure 13-31: Type C Block Diagram

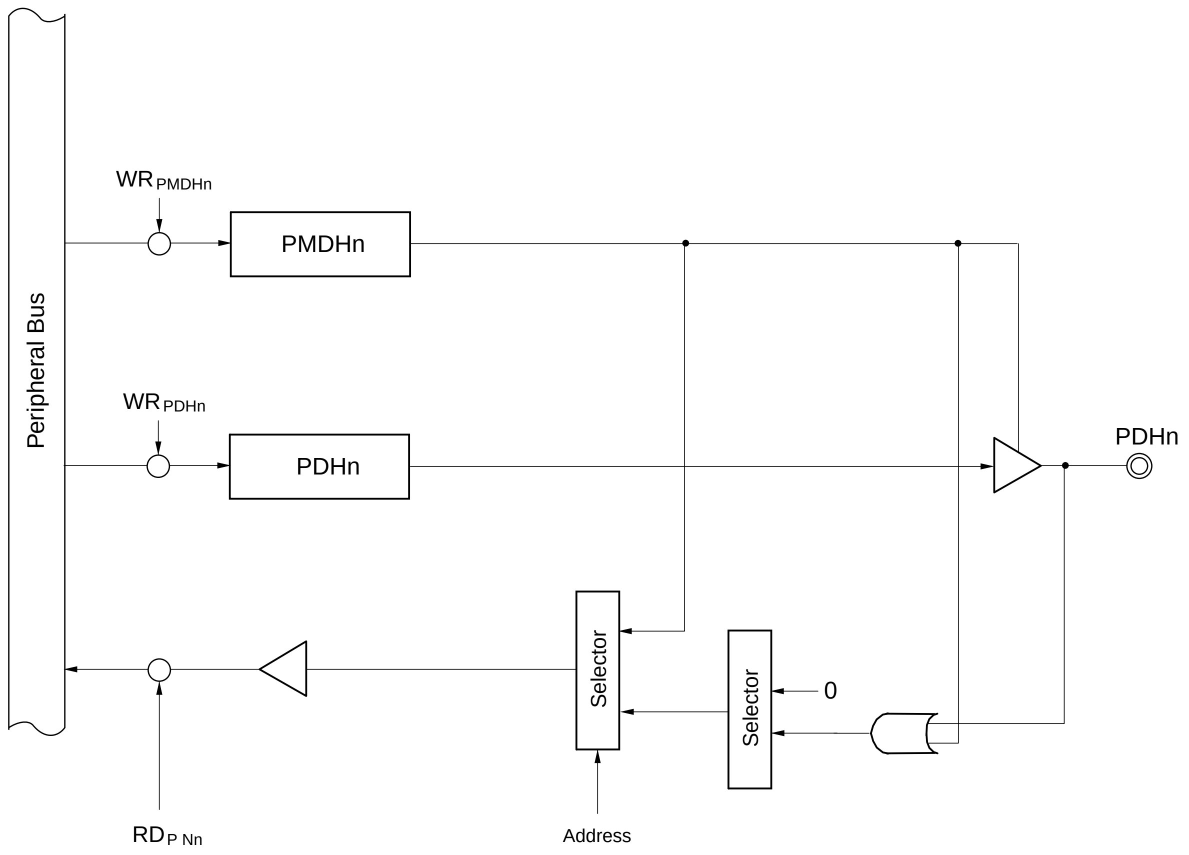


Remarks: 1. N:Port number

2. n :Bit number

13.14.4 Port Block Type D

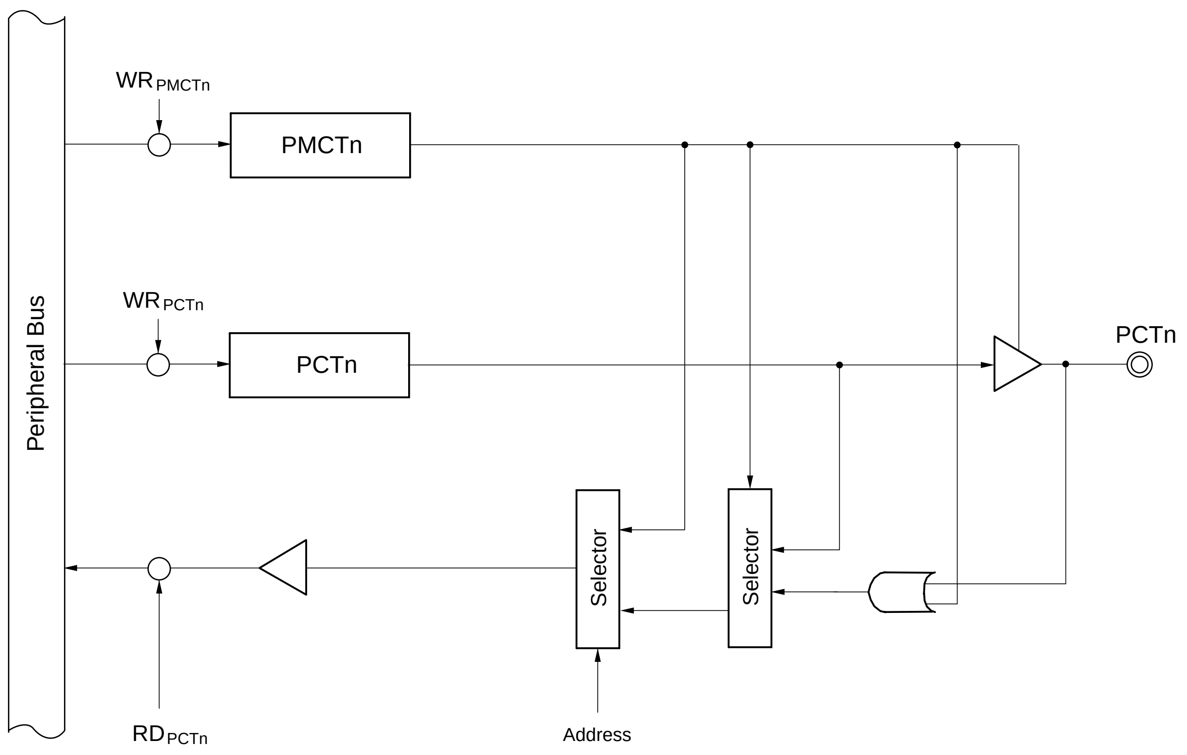
Figure 13-32: Type D Block Diagram



Remark: n : Bit number
n = 0 to 7.

13.14.5 Port Block Type E

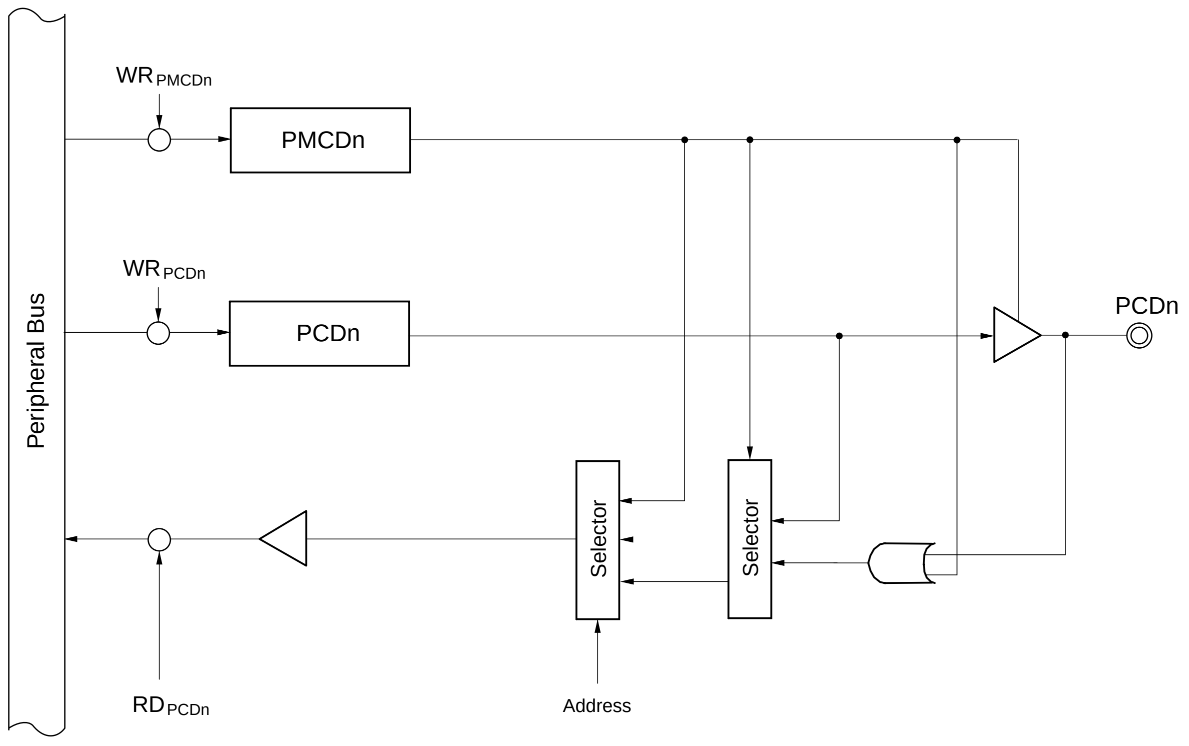
Figure 13-33: Type E Block Diagram



Remark: n: Bit number
n = 6, 4, 1, 0

13.14.6 Port Block Type F

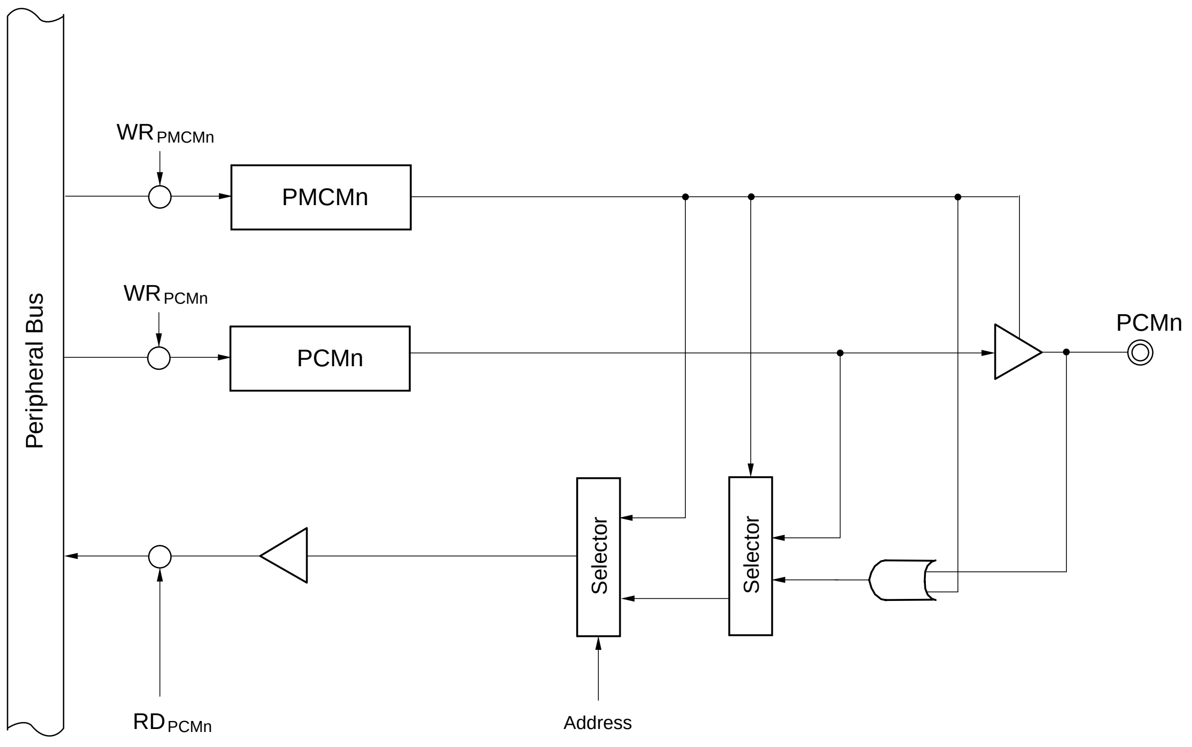
Figure 13-34: Type F Block Diagram



Remark: n: Bit number
n = 3, 2

13.14.7 Port Block Type G

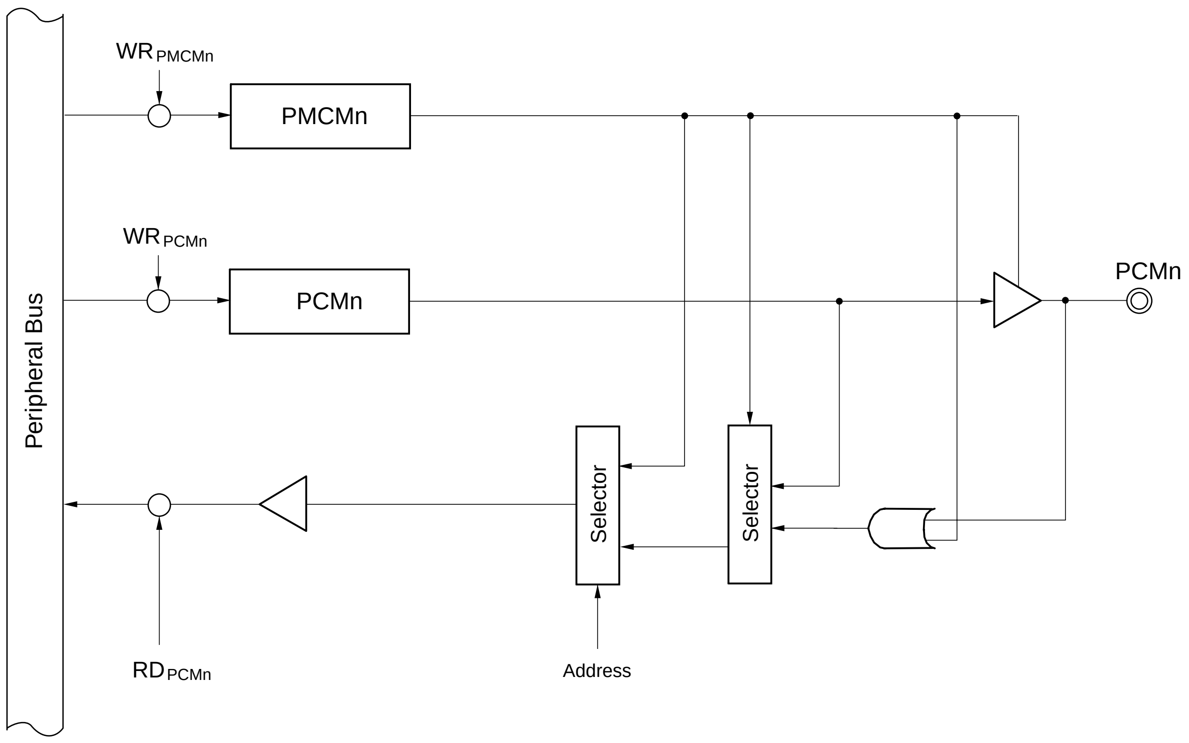
Figure 13-35: Type G Block Diagram



Remark: n: Bit number
n = 1, 2

13.14.8 Port Block Type H

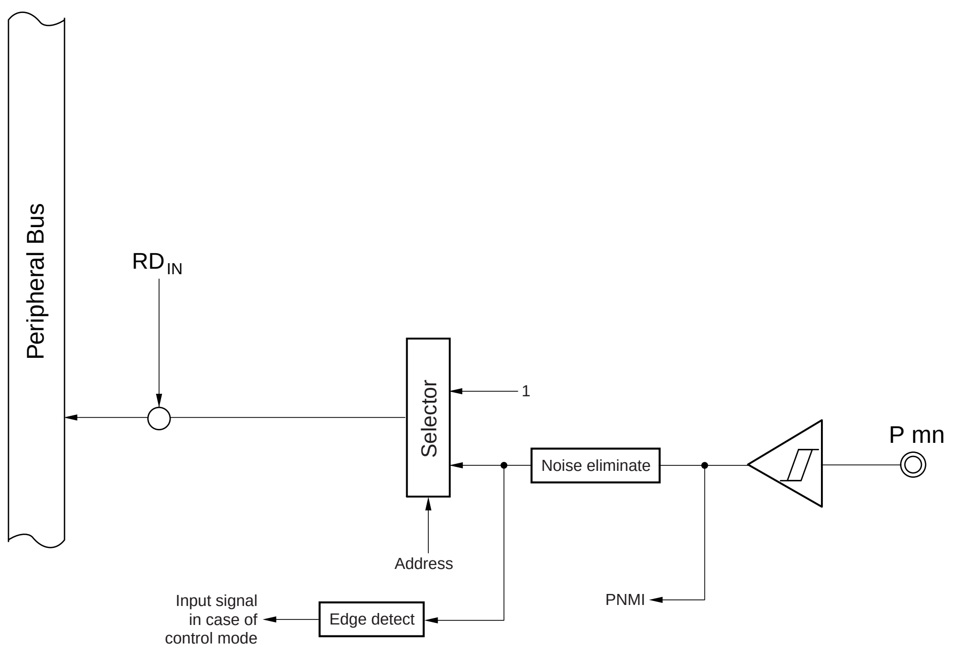
Figure 13-36: Type H Block Diagram



Remark: n: Bit number
n = 0, 3

13.14.9 Port Block Type I

Figure 13-37: Type I Block Diagram



Remarks: **1.** m:Port number

2. n:Bit number

13.15 Noise Elimination Filter and Edge detection

13.15.1 Filter and edge detection structure

Numbered ports P0, P1, P2, P3 have filter for noise suppression on all interrupts inputs.

Caution: To use peripheral interrupt or Timer G, input function of numbered ports, the user has to program the port register accordingly to peripheral input mode. Otherwise the peripheral function input level is always fixed to low level internally.

Filter have the following structure:

Figure 13-38: Type A: Input Filter and Edge Detection Overview for NMI

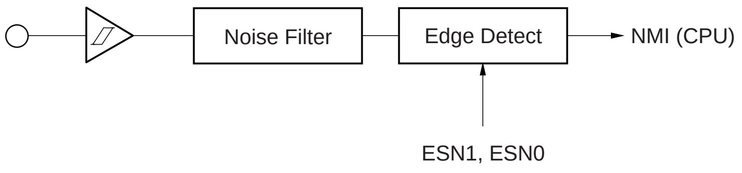
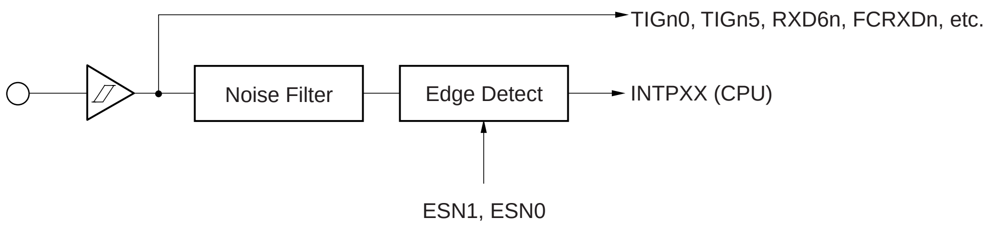


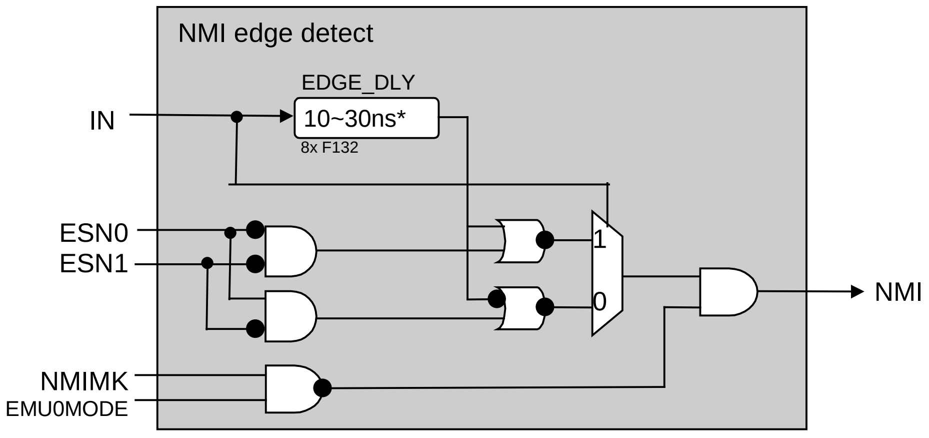
Figure 13-39: Type B: Input Filter and Edge Detection Overview for P0, P1, P2, P3



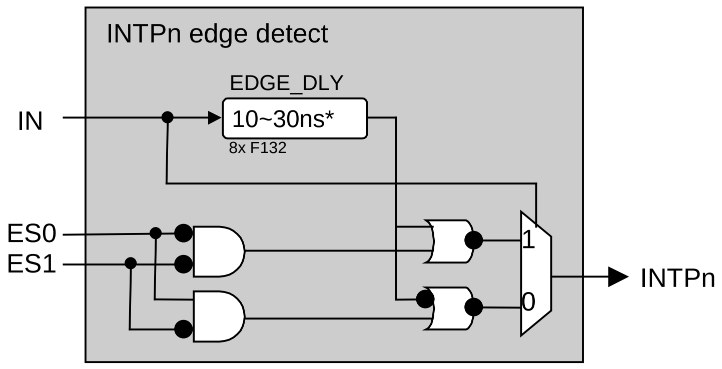
Edge select circuit is different for NMI and INTPn. First, default active edge is falling edge for INTPn and both edge for NMI. Both NMI and INTPn can be triggered by rising, falling or both edges. Level trigger is not available in HELIOS.
For NMI, active edge configuration is allowed only one time. Write access with default value is also count as write access and no further change is possible.

Caution: For NMI, masking in case of emulation is necessary. Please refer to emulation chapter for details. Glue logic for masking is implemented in Edge detection for NMI.

Figure 13-40: Edge Detect Circuit for INTPn / NMI



*: real time depends on wire delay



*: real time depends on wire delay

Remark: n=0 to 10

Caution: Delay time of delay lines for filter and edge detection has to be checked after layout. Delay of filter must be bigger than delay of edge detection circuit.

13.15.2 Filter configuration

The following table gives an overview about the assignment of filter types to the port input pins and defines the top-level names for the respective signals.

Table 13-2: Filter Assignment

Pin	Filter type	Port Pin	Remark
NMI	A	NMI	NMI
P00	B	INTP00	FCAN2 receive
P01	B	INTP01	FCAN2 transmit
P02	B	INTP02	PCL
P10	B	INTP10	TIG00
P15	B	INTP15	TIG05/SCK10
P20	B	INTP20	TIG10
P25	B	INTP25	TIG15
P30	B	INTP30	UART receive
P32	B	INTP32	UART receive
P34	B	INTP34	FCAN1 receive

13.15.3 Register for interrupt edge detection

External interrupt mode register INTM0,1,2,3, and INTNMI specify the valid edges of
external interrupt requests INTP00 to INTP03, INTP10, INTP15, INTP20, INTP25, INTP30, INTP32, INTP34 and NMI that are input from external pins. The correspondence of each register and the external interrupt request controlled by the register is shown below.

This register can be read or written in 1/8-bit units.

Figure 13-41: INTM Register for Interrupt Edge Detection Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
INTM0	0	0	ES021	ES020	ES011	ES010	ES001	ES000	FFFF FC00H	R/W	0000H
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
INTM1	0	0	0	0	ES151	ES150	ES101	ES100	FFFF FC02H	R/W	0000H
	R	R	R	R	R/W	R/W	R/W	R/W			

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
INTM2	0	0	0	0	ES251	ES250	ES201	ES200	FFFF FC04H	R/W	0000H
	R	R	R	R	R/W	R/W	R/W	R/W			

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
INTM3	0	0	ES341	ES340	ES321	ES320	ES301	ES300	FFFF FC06H	R/W	0000H
	R	R	R/W	R/W	R/W	R/W	R/W	R/W			

ESNx1	ESNx0	Edge select for INTPNx terminal
0	0	Falling edge
0	1	Rising edge
1	0	Both rising and falling edge
1	1	Both rising and falling edge

Caution: Programming edge detection or port mode register can trigger unintended interrupt requests. Therefore be sure to mask the respective interrupt requests and install appropriate interrupt handler for INT before reprogramming edge detection or port function.

Figure 13-42: NMIM Register for Interrupt Edge Detection Format

	7	6	5	4	3	2	1	0	Address	R/W	After Reset
NMIM	PNMI	0	0	0	0	0	ESN1	ESN0	FFFF FC08H	R/W	03H
	R	R	R	R	R	R	R/W	R/W			

ESN1	ESN0	Edge select for NMI terminal
0	0	Falling edge
0	1	Rising edge
1	0	Both rising and falling edges
1	1	Both rising and falling edges

PNMI	Edge select for NMI terminal

Caution: ESN1 and ESN0 are possible to change the setting only once. This register can be access by only 8bit unit.

Chapter 14 RESET Function

14.1 Overview

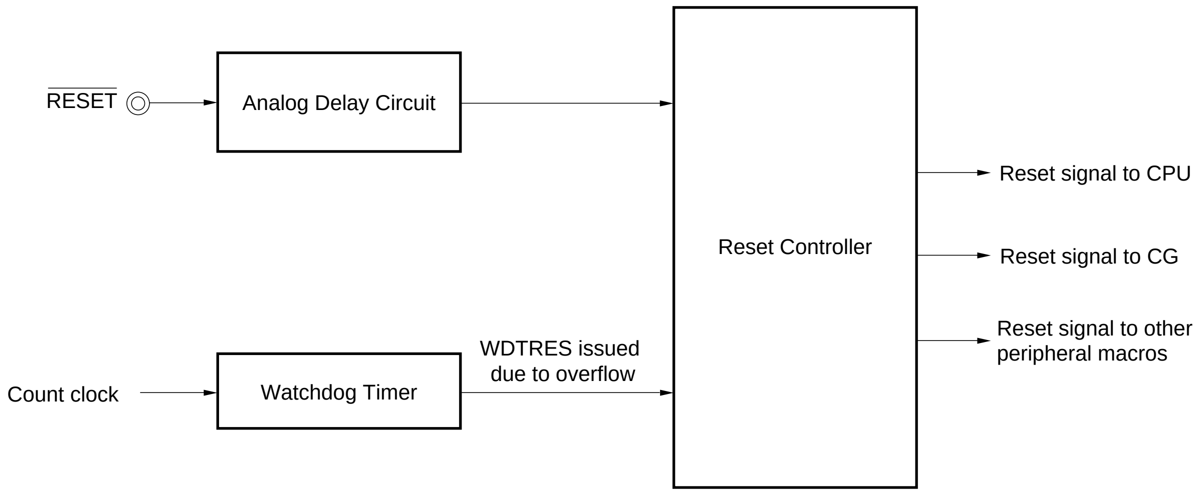
The following reset functions are available.

- Reset function by $\overline{\text{RESET}}$ pin input
- Reset function by overflow of the watchdog timer (WDTRES)

When the $\overline{\text{RESET}}$ pin goes high, the reset status is released, and the CPU starts executing the program. The contents of each register used in the program should be initialized as necessary.
The $\overline{\text{RESET}}$ pin has a noise eliminator that operates by analog delay.

14.2 Configuration

Figure 14-1: Reset Block Diagram



14.3 Operation

The system is reset, initializing each hardware unit, when a low level is input to the $\overline{\text{RESET}}$ pin or if watchdog timer overflows (WDTRES).
When the $\overline{\text{RESET}}$ pin goes high or after the WDTRES has been received, the reset status is released, and then the CPU starts program execution.

Table 14-1: Hardware Status on $\overline{\text{RESET}}$ Pin Input or Occurrence of WDTRES

Item	During Reset	After Reset
Main clock oscillator (f_{XX})	Oscillation starts (f_{XX} = 0 level).	Oscillation continues
Peripheral clock (PCLK), CPU clock (f_{CPU})	Operation stops	Operation starts. However, operation stops during oscillation stabilization time count.
Watchdog timer clock (f_{WDT})	Operation stops	Operation starts
Internal RAM	Undefined if power-on reset occurs or writing data to RAM and reset conflict (data loss); otherwise, retains values immediately before reset input.	
I/O lines (ports)	High impedance	
On-chip peripheral I/O registers	Initialized to specified status	
Other on-chip peripheral functions	Operation stops	Operation can be started

Figure 14-2: Hardware Status on $\overline{\text{RESET}}$ Input or Watchdog Request

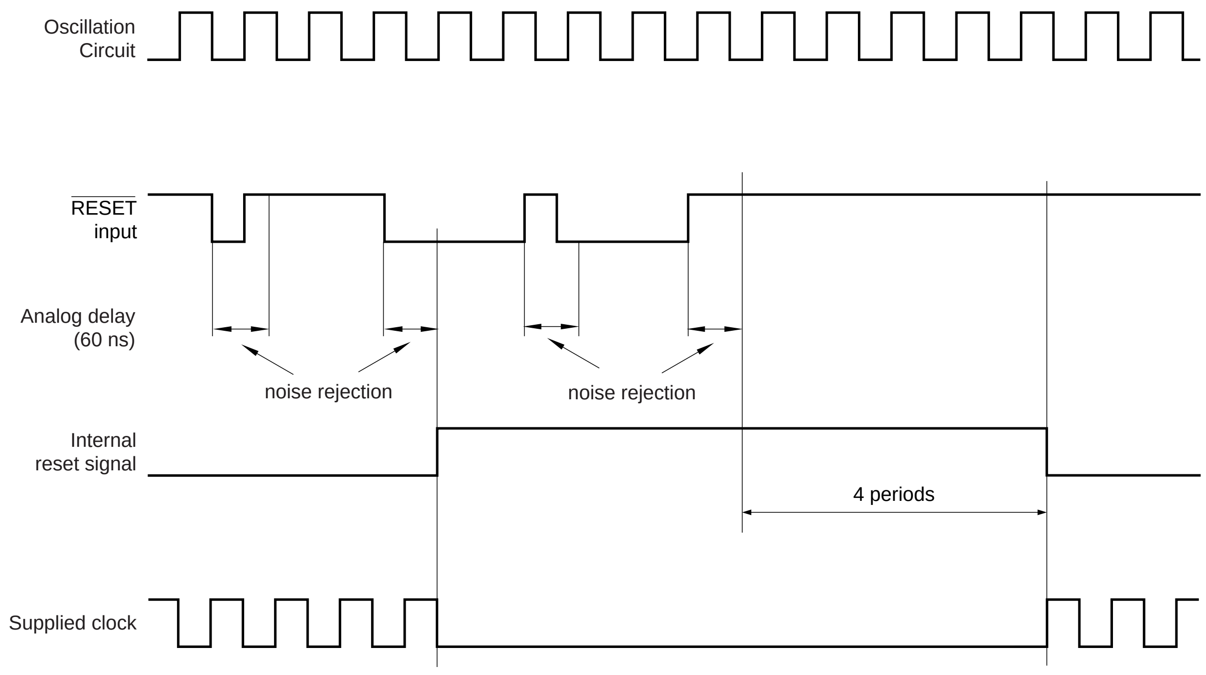
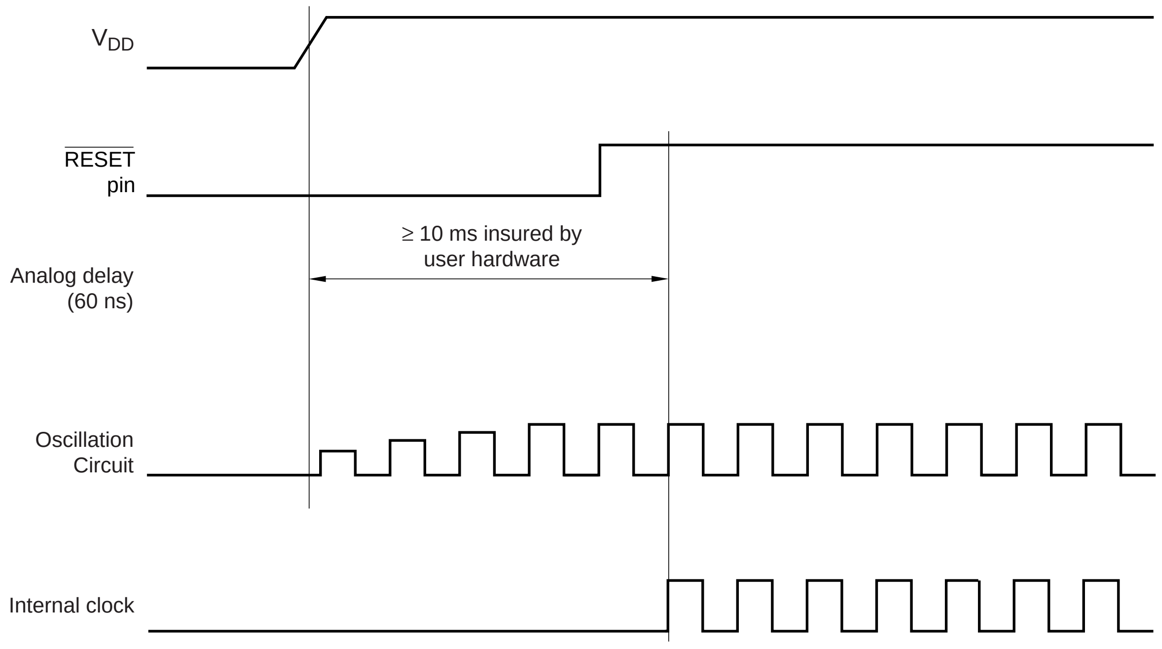


Figure 14-3: Operation on Power Application


V_{DD}
$\overline{\text{RESET}}$ pin
≥ 10 ms insured by user hardware
Analog delay (60 ns)
Oscillation Circuit
Internal clock

14.4 RESET cause identification

HELIOS provides the RESM flag useful in determination of the Reset cause:
RESM is only cleared by external hardware reset from reset pin, and only set by reset request from the watchdog timer.
A jump to the reset vector does not affect RESM.

The RESM flag is held in the Reset Source Monitor register RSM, which can be only read with 8 bits access.

Figure 14-4: Reset Cause Identification Register RSM

Symbol	7	6	5	4	3	2	1	0	Address	At Reset
RSM	0	0	0	0	0	0	0	RESM	FFFF F830H	00H
R	R	R	R	R	R	R	R	R		

RESM	Reset cause flag
0	Latest reset was caused by external $\overline{\text{RESET}}$ pin
1	Latest reset was caused by internal Watchdog reset

Chapter 15 Flash Memory (µPD70F3175 only)

The V850E/CA4 HELIOS provides a 256 KB flash memory. An instruction fetch from the flash memory takes one clock.
The flash memory can be programmed using a dedicated flash writer. Furthermore this product has a Selfprogramming mode, which allows to program the flash memory by control of the application without any dedicated writer.

The following can be considered as the development environment and the application using a flash memory:

- Software can be altered after the µPD70F3175 is solder mounted on the target system.
- Small scale production of various models is made easier by differentiating software.
- Data adjustment in starting mass production is made easier.
- Alter the software in the field using the Selfprogramming option.

15.1 Features

- 4-byte (1-word) access in 1 clock (instruction fetch access)
- Entire flash memory is divided into 2 areas, which can be erased separately
  - Area 0: 128 KB
  - Area 1: 128 KB
- Communication through serial interfaces (CSI0 and UART60) from the dedicated flash writer
- Erase/write voltage: V_{PP} = 7.8 V
- On-board programming using flash writer
- Selfprogramming mode
- After erase flash memory becomes FFFFFFFFH.

15.2 Writing by Flash writer

Writing can be performed either on-board or off-board by the dedicated flash writer.

(1) On-board programming

The contents of the flash memory is rewritten after the µPD70F3175 is mounted on the target system. It has to be ensured that the signals required for programming are made available to the flash writer.

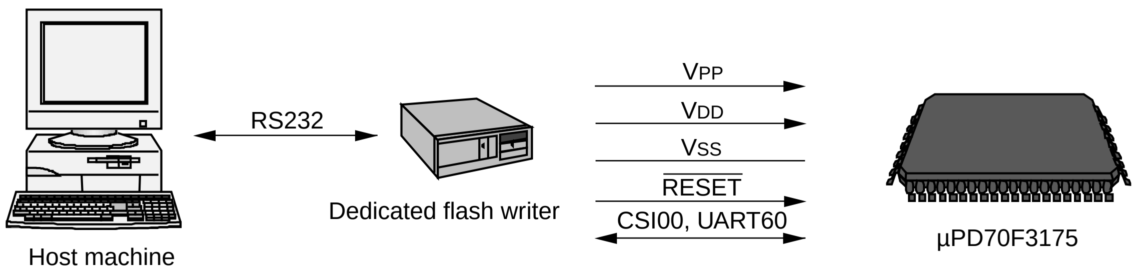
(2) Off-board programming

Writing to a flash memory is performed using a dedicated programming adapter (PA board), etc., before mounting the µPD70F3175 onto the target system.

15.3 Programming Environment

The following diagram shows the environment required for writing programs to the flash memory.

Figure 15-1: Programming Environment in Conjunction with External Flash Writer



A host machine can be used to control the flash writer.
CSI00 or UART60 is used as the interface between the flash writer and the µPD70F3175 to perform writing, erasing, etc. A programming adapter board is required for off-board writing.

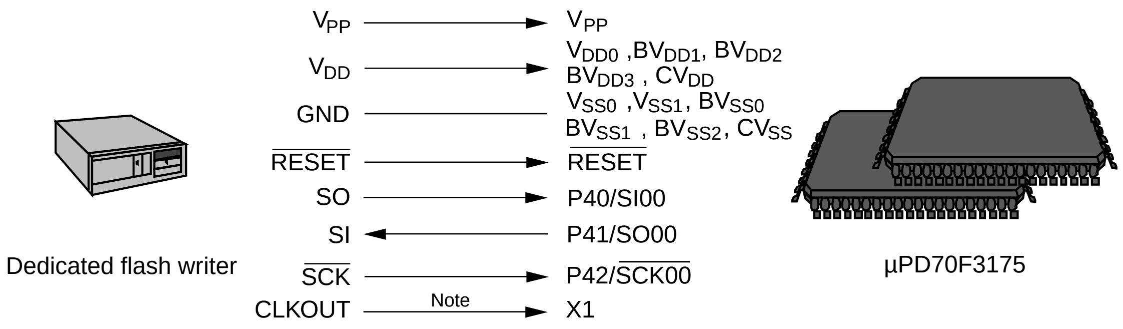
15.4 Communication System

The communication between the dedicated flash writer and the µPD70F3175 is performed by serial communication using CSI.

(1) CSI00

Transfer rate: up to 1.0 Mbps (MSB first)

Figure 15-2: Flash Writer Communication via CSI00



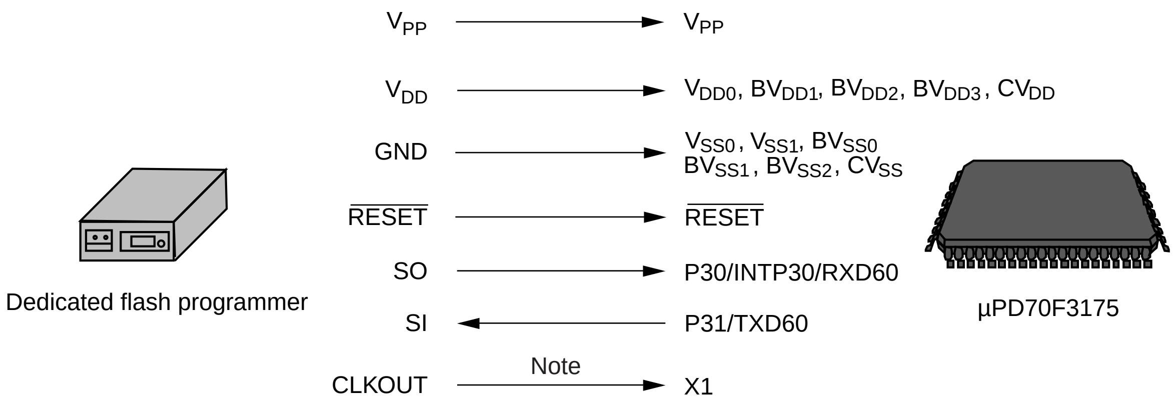
V_{PP}
V_{DD}
GND
$\overline{\text{RESET}}$
SO
SI
$\overline{\text{SCK}}$
CLKOUT
Note
V_{PP}
V_{DD0} ,BV_{DD1}, BV_{DD2}
BV_{DD3} , CV_{DD}
V_{SS0} ,V_{SS1}, BV_{SS0}
BV_{SS1} , BV_{SS2}, CV_{SS}
$\overline{\text{RESET}}$
P40/SI00
P41/SO00
P42/$\overline{\text{SCK00}}$
X1
Dedicated flash writer
µPD70F3175

Note: The supply of operating clock from the flash writer to the µPD70F3175 is not mandatory. Like in normal operating mode, the µPD70F3175 may also operate in flash memory programming mode with a target system clock, e.g. a crystal or resonator connected to X1, X2 pins. In such case do not connect the CLKOUT signal from the flash writer.

(2) UART

Transfer rate: 4,800 bps to 76,800 bps (LSB first)

Figure 15-3: Flash Writer Communication via UART60

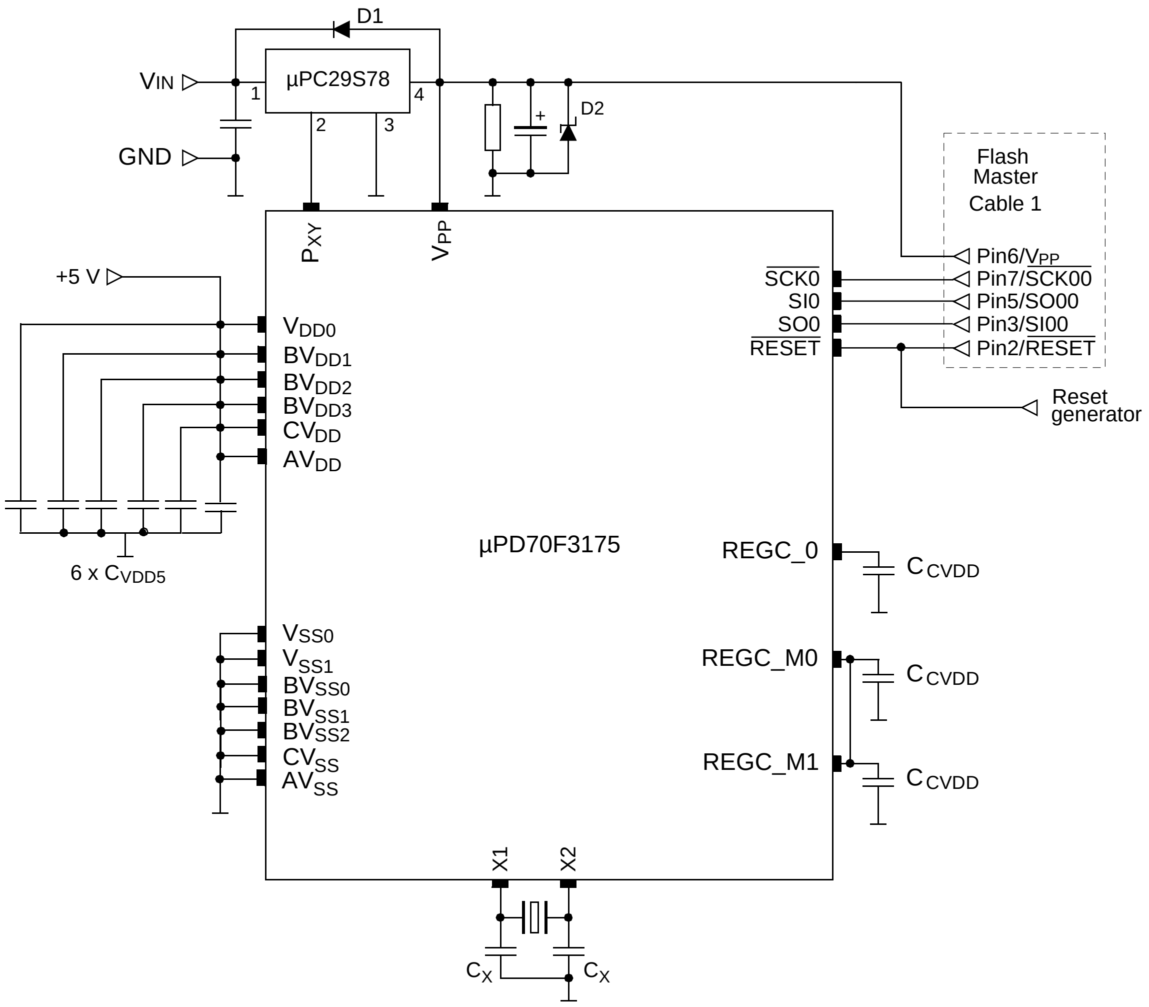

V_{PP}
V_{PP}
V_{DD}
V_{DD0}, BV_{DD1}, BV_{DD2}, BV_{DD3}, CV_{DD}
GND
V_{SS0}, V_{SS1}, BV_{SS0}
BV_{SS1}, BV_{SS2}, CV_{SS}
$\overline{RESET}$
$\overline{RESET}$
SO
P30/INTP30/RXD60
SI
P31/TXD60
CLKOUT
Note
X1
Dedicated flash programmer
µPD70F3175

Note: The supply of operating clock from the flash writer to the µPD70F3175 is not mandatory. Like in normal operating mode, the µPD70F3175 may also operate in flash memory programming mode with a target system clock, e.g. a crystal or resonator connected to X1, X2 pins. In such case do not connect the CLKOUT signal from the flash writer.

15.5 Flash Programming Circuitry

The following schematic shows the minimal circuitry. The circuitry incorporates a low-dropout voltage regulator (µPC29S78) as well as flash writer support. If the device is not used for Selfprogramming the V_{PP} pin have to be connected via a pull down resistor of 10 K to ground and the voltage regulator (µPC29S78) can be removed.

Figure 15-4: Minimal Circuitry for Flash Selfprogramming



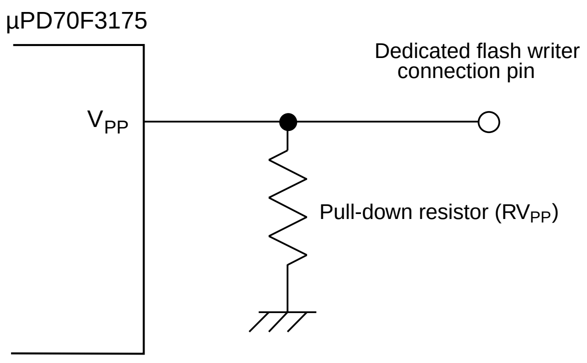
15.6 Pin Handling

When performing on-board writing, all required signals on the target system have to be made accessible to the dedicated flash writer. Also, it has to be ensured that the modes are set correctly and the V_{PP} signal, which is required to enter the programming mode can be controlled by the flash writer. In flash memory programming mode, all pins not required for the flash memory programming, remain in the same status as immediately after reset.

15.6.1 V_{PP} pin

In the normal operation mode, 0 V is input to V_{PP} pin. In the flash memory programming mode, 7.8 V writing voltage is supplied to V_{PP} pin. The following figure shows an example of the connection of the V_{PP} pin.

Figure 15-5: Pin Handling of V_{PP} pin


µPD70F3175
Dedicated flash writer connection pin
V_{PP}
Pull-down resistor (RV_{PP})

Remark: As pull-down resistor for using flash master 10 K are recommended.

15.6.2 Serial interface pins

The following shows the pins used by the serial interface.

Table 15-1: Serial Interface Pins

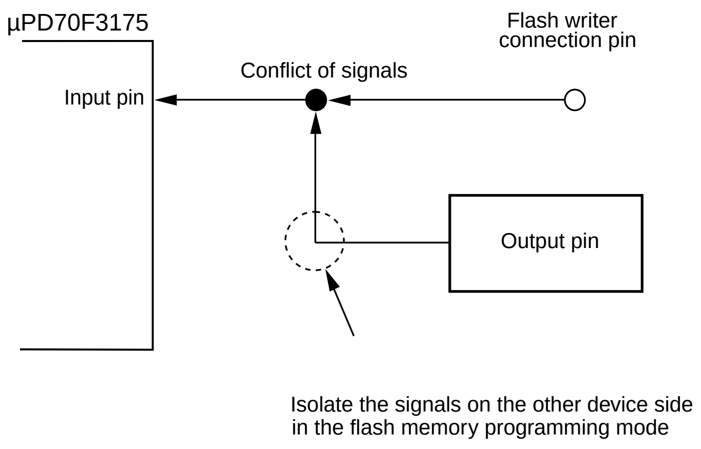
Serial Interface	Pins Used
CSI0	SO00, SI00, $\overline{\text{SCK00}}$
UART60	TXD60, RXD60

When connecting a dedicated flash writer to a serial interface pin, which is connected to other devices on-board, care should be taken to avoid the conflict of signals and the malfunction of other devices, etc.

(1) Conflict of signals

When connecting a flash writer (output) to a serial interface pin (input) which is connected to another device (output), conflict of signals may happen. To avoid the conflict of signals, isolate the connection to the other device or set the other device to the high-impedance status.

Figure 15-6: Conflict between Flash Writer and Other Output Pin

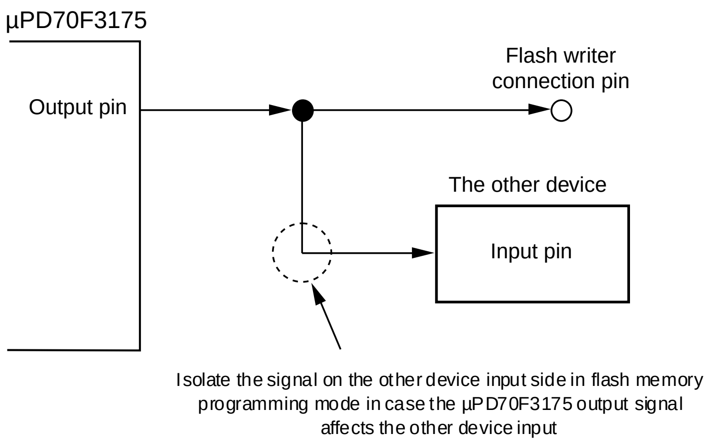

µPD70F3175
Flash writer connection pin
Conflict of signals
Input pin
Output pin
Isolate the signals on the other device side in the flash memory programming mode

(2) Malfunction of the other device

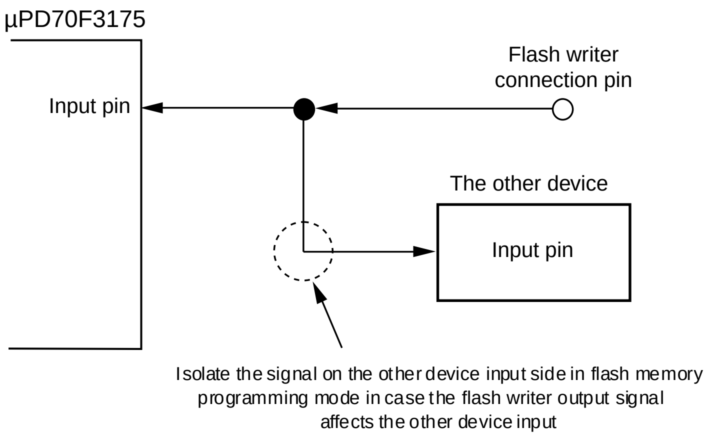
When connecting a flash writer (output or input) to a serial interface pin (input or output) connected to another device (input), the signal output to the other device may cause the device to malfunction. To avoid this, isolate the connection to the other device or make the setting so that the input signal to the other device is ignored.

Figure 15-7: Malfunction of Other Input Pins

(a) µPD70F3175 Serial Interface Output pin connection


µPD70F3175
Output pin
Flash writer connection pin
The other device
Input pin
Isolate the signal on the other device input side in flash memory programming mode in case the µPD70F3175 output signal affects the other device input

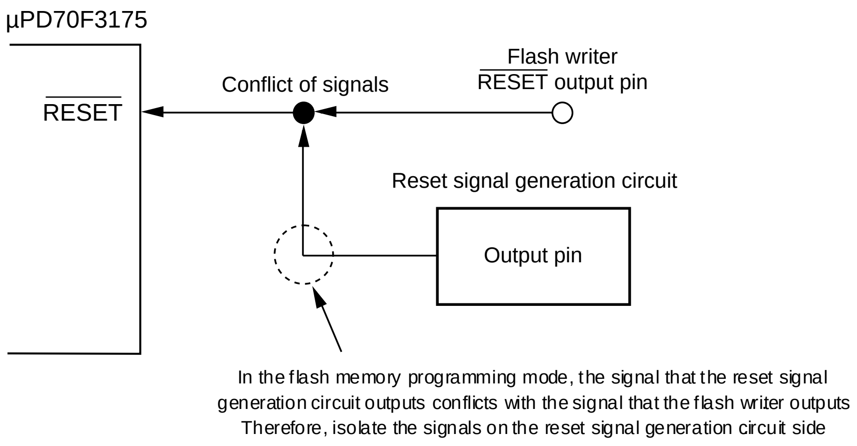
(b) µPD70F3175 Serial Interface Input pin connection


µPD70F3175
Input pin
Flash writer connection pin
The other device
Input pin
Isolate the signal on the other device input side in flash memory programming mode in case the flash writer output signal affects the other device input

15.6.3 $\overline{\text{RESET}}$ pin

When connecting the reset signals of the dedicated flash writer to the $\overline{\text{RESET}}$ pin which is connected to the reset signal generation circuit on-board, conflict of signals may happen. To avoid the conflict of signals, isolate the connection to the reset signal generation circuit.
When reset signal is input from the user system during the flash memory programming mode, programming operation will not be performed correctly. Therefore, do not input signals other than the reset signals from the dedicated flash writer.

Figure 15-8: Conflict between Flash Writer Reset Line and Reset Signal Generation Circuit


µPD70F3175
$\overline{\text{RESET}}$
Conflict of signals
Flash writer
$\overline{\text{RESET}}$ output pin
Reset signal generation circuit
Output pin
In the flash memory programming mode, the signal that the reset signal generation circuit outputs conflicts with the signal that the flash writer outputs Therefore, isolate the signals on the reset signal generation circuit side

15.6.4 NMI pin

Do not change the input signal to the NMI pin during the flash memory programming mode. If the NMI pin is changed during the flash memory programming mode, the programming may not be performed correctly.

15.6.5 Flash memory programming mode

To switch to the flash memory programming mode, apply writing voltage to the V_{PP} pin, and release the reset.

15.6.6 Port pins

When the flash memory programming mode is set, all the port pins except the pins which communicate with the dedicated flash writer become high-impedance state. The treatment of these port pins is not necessary.

15.6.7 Other signal pins

Connect X1, X2 and AV_{REF} to the same state as that in the normal operation mode.

15.6.8 Power supply

Provide the same power supply (V_{DD0} and BV_{DD0} to BV_{DD2}, C_{VDD}, C_{VSS}, V_{SS0} and V_{SS1} and BV_{SS0} to BV_{SS2}, AV_{DD}, AV_{SS}) as that in normal operation mode.

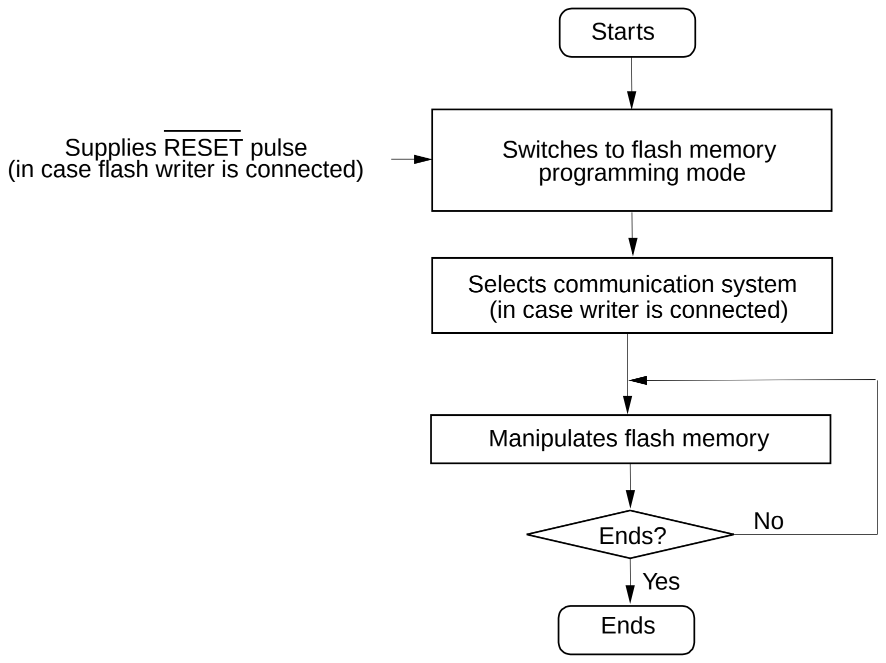
15.7 Programming Method

15.7.1 Flash memory control

To manipulate the flash memory the µPD70F3175 has to operate in a special flash memory programming mode. This mode can be entered either by applying the programming voltage of 7.8 V to the V_{PP} before the reset is release or by entering the Selfprogramming mode.

The following figure shows the procedure for manipulating the flash memory.

Figure 15-9: Flow Chart of Flash Memory Manipulation


Starts
Supplies $\overline{\text{RESET}}$ pulse (in case flash writer is connected)
Switches to flash memory programming mode
Selects communication system (in case writer is connected)
Manipulates flash memory
Ends?
No
Yes
Ends

15.7.2 Selection of communication mode

In the µPD70F3175 as well as for other V850 family devices, a communication system is selected by inputting pulses (16 pulses max.) to V_{PP} pin after switching to the flash memory programming mode. The V_{PP} pulses are generated by the dedicated flash writer.
The following table shows the relation between the number of pulses and the communication systems.

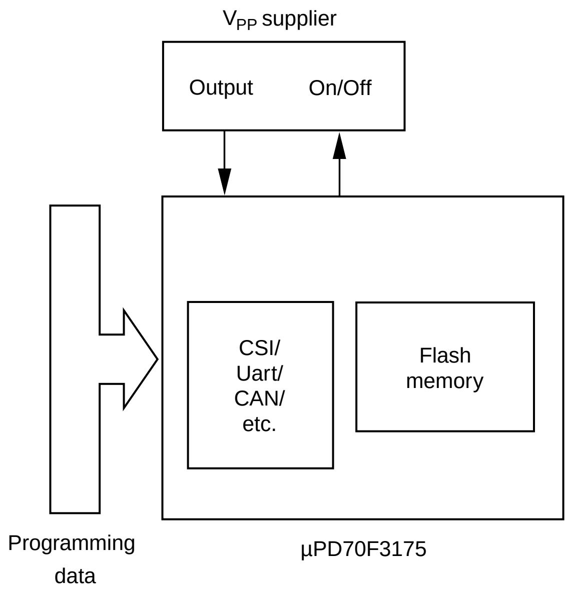
Table 15-2: List of Communication Systems

V_{PP} pulse	Communication System	Remarks
0	CSI00	µPD70F3175 performs slave operation, MSB first
8	UART60 Communication	rate: 9600 bps (after reset), LSB first
Others	(reserved)	Setting prohibited

15.8 Selfprogramming Mode

The flash Selfprogramming feature allows user to reprogram the flash contents by a user application program, without the necessity of an external flash writer.
This feature allows an update of the application with only on-board resources and a user defined communication interface.

Figure 15-10: Configuration in Selfprogramming Mode



In order to operate flash Selfprogramming, flash Selfprogramming libraries are prepared for user.

Following operations to the flash memory are supported by libraries.

- Initialize
- Blank Check
- Erase
- Write
- Verify
- Blank check
- V_{PP} Voltage Check
- Create Signature
- Check Signature
- Swap Area
- Check Area

For further details please refer to the following document:

- Application Note - Self-Programming
  32-/16-bit Single-Chip Microcontroller -- Self-Programming Library
  (Doc.No.: U15352EE2V0AN00)

15.9 Secure Selfprogramming

15.9.1 General description

A flash memory area can only be erased as a whole. If parts of the lower flash area have to be updated, the complete flash has to be erased. This bears the risk that a problem during Selfprogramming, particularly a power failure, leaves the device without any valid program for start-up.
To overcome those limitations µPD70F3175 features a method which is called “secure Selfprogramming”. By using “secure Selfprogramming”, it is always ensured that a valid boot program is available in the flash memory. This is achieved by enabling the user to select which of the two flash areas is mapped at address 0 and therefore accessed after reset, thus ensuring that the boot program located in this area is executed.

This selection is done by creating a signature at address 08H or 20008H, depending on which area should become the one located at address 0. Directly after reset, the device determines which area contains a valid signature and maps this area to address 0.

15.9.2 Signature structure

The library provides a function to create a signature in either one of the two areas. It is located within the user address space of the flash memory. The signature structure was chosen in a way to ensure that no user data can be mistakenly interpreted as a signature. This is achieved by a different usage of internal structures of the flash memory.

15.9.3 Secure Selfprogramming flow

A reprogramming of the flash memory starts with an erase of the upper area. After a successful erase, the boot program that is located in the lower area has to be copied (modification possible) to the upper area. Afterwards a signature is created in the upper area, indicating that this area contains a valid boot program. The signature which is found in the lower area is destroyed by writing a 00000000H to this address, and the areas are swapped, ensuring that the copied boot program is now located at address 0. This is followed by an erase of the area, which became the upper one still containing the old boot program and an invalid signature. After completion of the erase operation, the flash memory contains now only the boot program, so that the new application program can be written.

The flow looks as follows:

- Erase upper area
- Copy boot program from lower area to upper area
- Create signature in upper area
- Kill signature in lower area
- Swap area so that the lower area becomes the upper and vice versa
- Erase the upper area (which was formerly the lower)
- Write new application program to the flash memory

Figure 15-11: Secure Selfprogramming Flow (1/2)

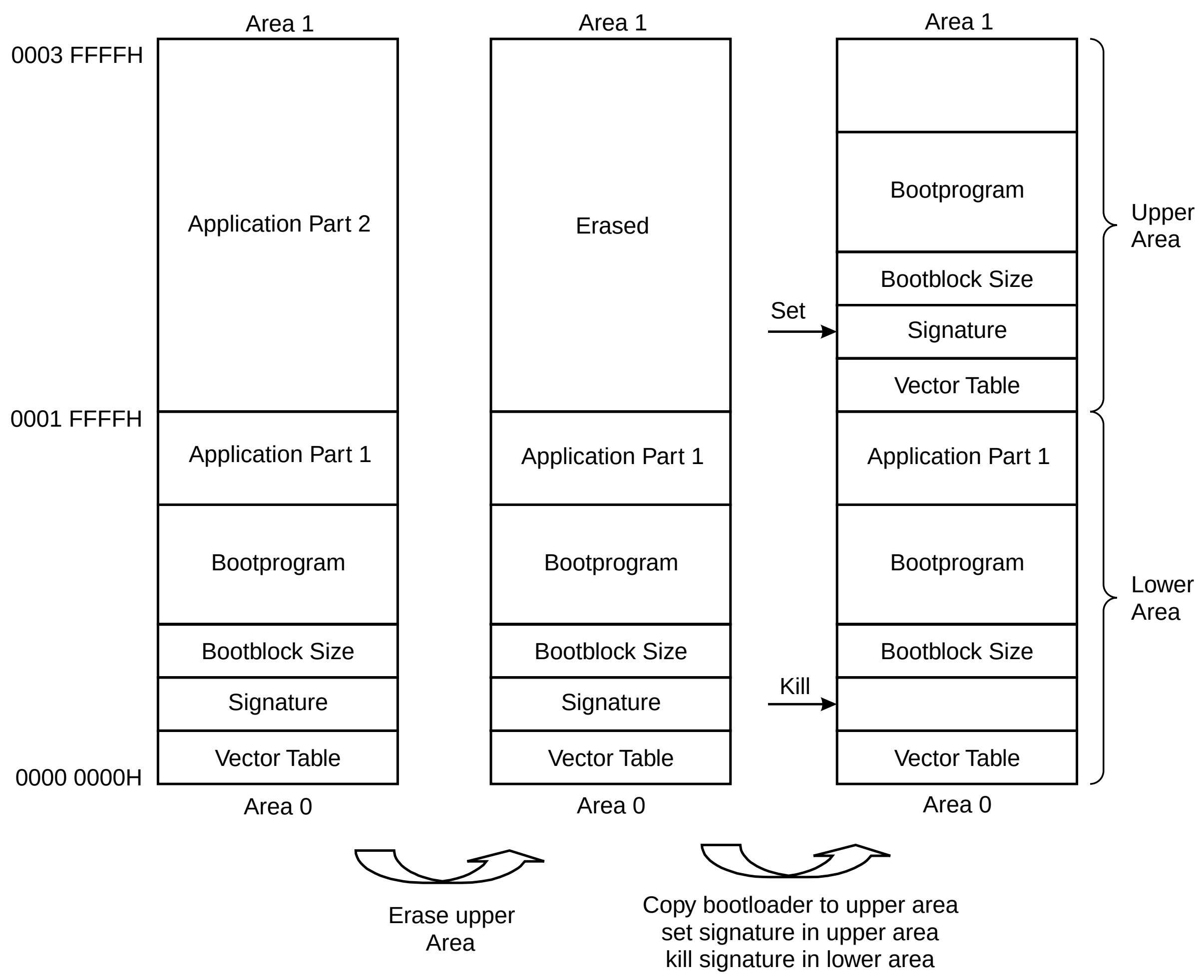
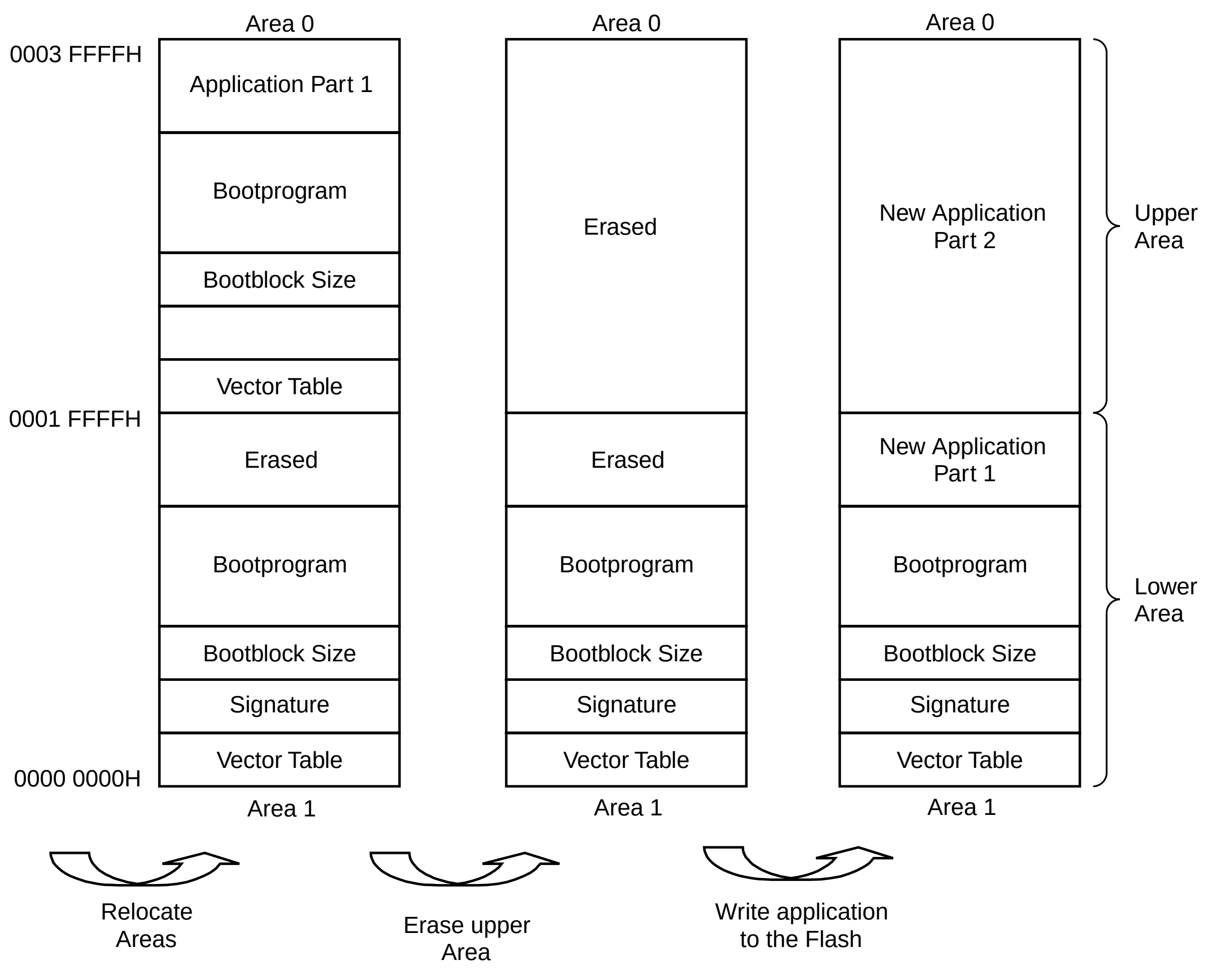


Figure 15-11: Secure Selfprogramming Flow (2/2)



15.9.4 Advantages of Secure Selfprogramming

- A boot program is always available, thus ensuring that the device can always be reprogrammed.
- The size of the boot block is not fixed. All sizes up to 128KB are supported.
- It is possible to update the boot program using the secure mechanisms.

Appendix A Instruction Set List

A.1 Convention

(a) Register symbols used to describe operands

Register Symbol	Explanation
reg1	General registers: Used as source registers
reg2	General registers: Used mainly as destination registers. Also used as source register in some instructions.
reg3	General registers: Used mainly to store the remainders of division results and the higher order 3 bits of multiplication results.
bit#3	33-bit data for specifying the bit number
immX	X bit immediate data
dispX	X bit displacement data
regID	System register number
vector	5-bit data that specifies the trap vector (00H to 1FH)
cccc	4-bit data that shows the conditions code
sp	Stack pointer (SP)
ep	Element pointer (r30)
listX	X item register list

(b) Register symbols used to describe opcodes

Register Symbol	Explanation
R	1-bit data of a code that specifies reg1 or regID
r	1-bit data of the code that specifies reg2
w	1-bit data of the code that specifies reg3
d	1-bit displacement data
I	1-bit immediate data (indicates the higher bits of immediate data)
i	1-bit immediate data
cccc	4-bit data that shows the condition codes
CCCC	4-bit data that shows the condition codes of Bcond instruction
bbb	3-bit data for specifying the bit number
L	1-bit data that specifies a program register in the register list
S	1-bit data that specifies a system register in the register list

(c) Register symbols used in operation

Register Symbol	Explanation
←	Input for
GR [ ]	General register
SR [ ]	System register
zero-extend (n)	Expand n with zeros until word length.
sign-extend (n)	Expand n with signs until word length.
load-memory (a, b)	Read size b data from address a.
store-memory (a, b, c)	Write data b into address a in size c.
load-memory-bit (a, b)	Read bit b of address a.
store-memory-bit (a, b, c)	Write c to bit b of address a.
saturated (n)	Execute saturated processing of n (n is a 2's complement). If, as a result of calculations, n ≥ 7FFFFFFFH, let it be 7FFFFFFFH. n ≤ 80000000H, let it be 80000000H.
result	Reflects the results in a flag.
Byte	Byte (8 bits)
Half-word	Half word (16 bits)
Word	Word (32 bits)
+	Addition
–	Subtraction
||	Bit concatenation
×	Multiplication
÷	Division
%	Remainder from division results
AND	Logical product
OR	Logical sum
XOR	Exclusive OR
NOT	Logical negation
logically shift left by	Logical shift left
logically shift right by	Logical shift right
arithmetically shift right by	Arithmetic shift right

(d) Register symbols used in an execution clock

Register Symbol	Explanation
I	If executing another instruction immediately after executing the first instruction (issue).
r	If repeating execution of the same instruction immediately after executing the first instruction (repeat).
l	If using the results of instruction execution in the instruction immediately after the execution (latency).

(e) Register symbols used in flag operations

Identifier	Explanation
(Blank)	No change
0	Clear to 0
X	Set or cleared in accordance with the results.
R	Previously saved values are restored.

(f) Condition codes

Condition Name (cond)	Condition Code (cccc)	Condition Formula	Explanation
V	0 0 0 0	OV = 1	Overflow
NV	1 0 0 0	OV = 0	No overflow
C/L	0 0 0 1	CY = 1	Carry Lower (Less than)
NC/NL	1 0 0 1	CY = 0	No carry Not lower (Greater than or equal)
Z/E	0 0 1 0	Z = 1	Zero Equal
NZ/NE	1 0 1 0	Z = 0	Not zero Not equal
NH	0 0 1 1	(CY or Z) = 1	Not higher (Less than or equal)
H	1 0 1 1	(CY or Z) = 0	Higher (Greater than)
N	0 1 0 0	S = 1	Negative
P	1 1 0 0	S = 0	Positive
T	0 1 0 1	—	Always (Unconditional)
SA	1 1 0 1	SAT = 1	Saturated
LT	0 1 1 0	(S xor OV) = 1	Less than signed
GE	1 1 1 0	(S xor OV) = 0	Greater than or equal signed
LE	0 1 1 1	((S xor OV) or Z) = 1	Less than or equal signed
GT	1 1 1 1	((S xor OV) or Z) = 0	Greater than signed

A.2 Instruction Set (In Alphabetical Order)

(1/4)

Mnemonic	Operand	Opcode	Operation	Execution Clock			Flags				
				i	r	l	CY	OV	S	Z	SAT
ADD	reg1,reg2	`rrrrr001110RRRRR`	GR[reg2]←GR[reg2] + GR[reg1]	1	1	1	×	×	×	×	
	imm5,reg2	`rrrrr010010iiiii`	GR[reg2]←GR[reg2] + sign-extend(imm5)	1	1	1	×	×	×	×	
ADDI	imm16,reg1,reg2	`rrrrr110000RRRRR` `iiiiiiiiiiiiiiii`	GR[reg2]←GR[reg1] + sign-extend(imm16)	1	1	1	×	×	×	×	
AND	reg1,reg2	`rrrrr001010RRRRR`	GR[reg2]←GR[reg2] AND GR[reg1]	1	1	1		0	×	×	
ANDI	imm16,reg1,reg2	`rrrrr110110RRRRR` `iiiiiiiiiiiiiiii`	GR[reg2]←GR[reg1] AND zero-extend(imm16)	1	1	1		0	0	×	×
Bcond	disp9	`ddddd1011dddcccc` Note 1	if conditions are satisfied then PC←PC+sign-extend(disp9) — When conditions are satisfied	2 Note 2	2 Note 2	2 Note 2					
			When conditions are not satisfied	1	1	1					
BSH	reg2,reg3	`rrrrr11111100000` `wwwww01101000010`	GR[reg3]←GR[reg2] (23 : 16) ll GR[reg2] (31 : 24) ll GR[reg2] (7 : 0) ll GR[reg2] (15 : 8)	1	1	1	×	0	×	×	
BSW	reg2,reg3	`rrrrr11111100000` `wwwww01101000000`	GR[reg3]←GR[reg2] (7 : 0) ll GR[reg2] (15 : 8) ll GR[reg2] (23 : 16) ll GR[reg2] (31 : 24)	1	1	1	×	0	×	×	
CALLT	imm6	`0000001000iiiiii`	CTPC←PC + 2(return PC) CTPSW←PSW adr←CTBP+zero-extend(imm6 logically shift left by 1) PC←CTBP+zero-extend(Load-memory(adr,Half-word))	4	4	4					
CLR1	bit#3, disp16[reg1]	`10bbb111110RRRRR` `dddddddddddddddd`	adr←GR[reg1] + sign-extend(disp16) Z flag←Not(Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,0)	3 Note 3	3 Note 3	3 Note 3				×	
	reg2,[reg1]	`rrrrr111111RRRRR` `0000000011100100`	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,0)	3 Note 3	3 Note 3	3 Note 3				×	
CMOV	cccc,imm5,reg2, reg3	`rrrrr111111iiiii` `wwwww011000cccc0`	if conditions are satisfied then GR[reg3]←sign-extended(imm5) else GR[reg3]←GR[reg2]	1	1	1					
	cccc,reg1,reg2, reg3	`rrrrr111111RRRRR` `wwwww011001cccc0`	if conditions are satisfied then GR[reg3]←GR[reg1] else GR[reg3]←GR[reg2]	1	1	1					
CMP	reg1,reg2	`rrrrr001111RRRRR`	result←GR[reg2] – GR[reg1]	1	1	1	×	×	×	×	
	imm5,reg2	`rrrrr010011iiiii`	result←GR[reg2] – sign-extend(imm5)	1	1	1	×	×	×	×	
CTRET		`0000011111100000` `0000000101000100`	PC←CTPC PSW←CTPSW	3	3	3	R	R	R	R	R
DI		`0000011111100000` `0000000101100000`	PSW.ID←1	1	1	1					
DISPOSE	imm5,list12	`0000011001iiiiiL` `LLLLLLLLLLL00000`	sp←sp + zero-extend(imm5 logically shift left by 2) GR[reg in list12]←Load-memory(sp,Word) sp←sp + 4 repeat 2 steps above until all regs in list12 are loaded	N+1 Note 4	N+1 Note 4	N+1 Note 4					
	imm5,list12,[reg1]	`0000011001iiiiiL` `LLLLLLLLLLLRRRRR` Note 5	sp←sp + zero-extend(imm5 logically shift left by 2) R[reg in list12]←Load-memory(sp,Word) sp←sp + 4 repeat 2 steps above until all regs in list12 are loaded PC←GR[reg1]	N+3 Note 4	N+3 Note 4	N+3 Note 4					
DIV	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01011000000`	GR[reg2]←GR[reg2] ÷ GR[reg1] GR[reg3]←GR[reg2] % GR[reg1]	35	35	35					
DIVH	reg1,reg2	`rrrrr000010RRRRR`	GR[reg2]←GR[reg2] ÷ GR[reg1]Note 6	35	35	35		×	×	×	
	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01010000000`	GR[reg2]←GR[reg2] ÷ GR[reg1]Note 6 GR[reg3]←GR[reg2] % GR[reg1]	35	35	35		×	×	×	
DIVHU	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01010000010`	GR[reg2]←GR[reg2] ÷ GR[reg1]Note 6 GR[reg3]←GR[reg2] % GR[reg1]	34	34	34		×	×	×	
DIVU	reg1,reg2,reg3	`rrrrr111111RRRRR` `wwwww01011000010`	GR[reg2]←GR[reg2] ÷ GR[reg1] GR[reg3]←GR[reg2] % GR[reg1]	34	34	34		×	×	×	
EI		`1000011111100000` `0000000101100000`	PSW.ID←0	1	1	1					
HALT		`0000011111100000` `0000000100100000`	Stop	1	1	1					
HSW	reg2,reg3	`rrrrr11111100000` `wwwww01101000100`	GR[reg3]←GR[reg2](15 : 0) ll GR[reg2] (31 : 16)	1	1	1	×	0	×	×	
JARL	disp22,reg2	`rrrrr11110dddddd` `ddddddddddddddd0` Note 7	GR[reg2]←PC + 4 PC←PC + sign-extend(disp22)	2	2	2					
JMP	[reg1]	`00000000011RRRRR`	PC←GR[reg1]	3	3	3					

(2/4)

Mnemonic	Operand	Opcode	Operation		Execution Clock			Flags				
					i	r	l	CY	OV	S	Z	SAT
JR	disp22	0000011110dddddd ddddddddddddddd0 Note 7	PC←PC + sign-extend(disp22)		2	2	2					
LD.B	disp16[reg1],reg2	rrrrr111000RRRRR dddddddddddddddd	adr←GR[reg1] + sign-extend(disp16) GR[reg2]←sign-extend(Load-memory(adr,Byte))		1	1	Note 11					
LD.BU	disp16[reg1],reg2	rrrrr11110bRRRRR ddddddddddddddd1 Notes 8, 10	adr←GR[reg1] + sign-extend(disp16) GR[reg2]←zero-extend(Load-memory(adr,Byte))		1	1	Note 11					
LD.H	disp16[reg1],reg2	rrrrr111001RRRRR ddddddddddddddd0 Note 8	adr←GR[reg1] + sign-extend(disp16) GR[reg2]←sign-extend(Load-memory(adr,Half-word))		1	1	Note 11					
LDSR	reg2,regID	rrrrr111111RRRRR 0000000000100000 Note 12	SR[regID]←GR[reg2]	Other than regID = PSW	1	1	1					
				regID = PSW	1	1	1	×	×	×	×	×
LD.HU	disp16[reg1],reg2	rrrrr111111RRRRR ddddddddddddddd1 Note 8	adr←GR[reg1]+sign-exend(disp16) GR[reg2]←zero-extend(Load-memory(adr,half-word)		1	1	Note 11					
LD.W	disp16[reg1],reg2	rrrrr111001RRRRR ddddddddddddddd1 Note 8	adr←GR[reg1] + sign-exend(disp16) GR[reg2]←Load-memory(adr,Word)		1	1	Note 9					
MOV	reg1,reg2	rrrrr000000RRRRR	GR[reg2]←GR[reg1]		1	1	1					
	imm5,reg2	rrrrr010000iiiii	GR[reg2]←sign-extend(imm5)		1	1	1					
	imm32,reg1	00000110001RRRRR iiiiiiiiiiiiiiii iiiiiiiiiiiiiiii	GR[reg1]←imm32		2	2	2					
MOVEA	imm16,reg1,reg2	rrrrr110001RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1] + sign-extend(imm16)		1	1	1					
MOVHI	imm16,reg1,reg2	rrrrr110010RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1] + (imm16 ll 0^{16})		1	1	1					
MUL	reg1,reg2,reg3	rrrrr111111RRRRR wwwww01000100000	GR[reg3] ll GR[reg2]←GR[reg2] × GR[reg1]		1	2 Note 14	2					
	imm9,reg2,reg3	rrrrr111111iiiii wwwww01001IIII00	GR[reg3] ll GR[reg2]←GR[reg2] × sign-extend(imm9) **Note 13**		1	2 Note 14	2					
MULH	reg1,reg2	rrrrr000111RRRRR	GR[reg2]←GR[reg2][Note 6] × GR[reg1][Note 6]		1	1	2					
	imm5,reg2	rrrrr010111iiiii	GR[reg2]←GR[reg2][Note 6] × sign-extend(imm5)		1	1	2					
MULHI	imm16,reg1,reg2	rrrrr110111RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1][Note 6] × imm16		1	1	2					
MULU	reg1,reg2,reg3	rrrrr111111RRRRR wwwww01000100010	GR[reg3] ll GR[reg2]←GR[reg2]xGR[reg1]		1	2 Note 14	2					
	imm9,reg2,reg3	rrrrr111111iiiii wwwww01001IIII10	GR[reg3] ll GR[reg2]←GR[reg2]xzero-extend(imm9) Note 13		1	2 Note 14	2					
NOP		0000000000000000	Pass at least one clock cycle doing nothing.		1	1	1					
NOT	reg1,reg2	rrrrr000001RRRRR	GR[reg2]←NOT(GR[reg1])		1	1	1		0	×	×	
NOT1	bit#3,disp16[reg1]	01bbb111110RRRRR dddddddddddddddd	adr←GR[reg1]+sign-extend(disp16) Z flag←Not(Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,Z flag)		3 Note 3	3 Note 3	3 Note 3				×	
	reg2,[reg1]	rrrrr111111RRRRR 0000000011100010	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,Z flag)		3 Note 3	3 Note 3	3 Note 3				×	
OR	reg1,reg2	rrrrr001000RRRRR	GR[reg2]←GR[reg2]OR GR[reg1]		1	1	1		0	×	×	
ORI	imm16,reg1,reg2	rrrrr110100RRRRR iiiiiiiiiiiiiiii	GR[reg2]←GR[reg1]OR zero-extend(imm16)		1	1	1		0	×	×	
PREPARE	list12,imm5	0000011110iiiiiL LLLLLLLLLLL00001	Store-memory(sp – 4,GR[reg in list12],Word) sp←sp – 4 repeat 1 step above until all regs in list12 are stored sp←sp – zero-extend(imm5)		n+1 Note 4	n+1 Note 4	n+1 Note 4					
	list12,imm5, sp/imm[Note 15]	0000011110iiiiiL LLLLLLLLLLLff011 imm16/imm32 Note 16	Store-memory(sp – 4,GR[reg in list12],Word) sp←sp – 4 repeat 1 step above until all regs in list12 are stored sp←sp – zero-extend(imm5) ep←sp/imm		n+2 Note 4 Note 17	n+2 Note 4 Note 17	n+2 Note 4 Note 17					

(3/4)

Mnemonic	Operand	Opcode	Operation	Execution Clock			Flags				
				i	r	l	CY	OV	S	Z	SAT
RETI		0000011111100000 0000000101000000	if PSW.EP=1 then PC ←EIPC PSW ←EIPSW else if PSW.NP=1 then PC ←FEPC PSW ←FEPSW else PC←EIPC PSW ←EIPSW	3	3	3	R	R	R	R	R
SAR	reg1,reg2	rrrrr111111RRRRR 0000000010100000	GR[reg2]←GR[reg2] arithmetically shift right by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010101iiiii	GR[reg2]←GR[reg2] arithmetically shift right by zero-extend (imm5)	1	1	1	×	0	×	×	
SASF	cccc,reg2	rrrrr1111110cccc 0000001000000000	if conditions are satisfied then GR[reg2]←(GR[reg2]Logically shift left by 1) OR 00000001H else GR[reg2]←(GR[reg2]Logically shift left by 1) OR 00000000H	1	1	1					
SATADD	reg1,reg2	rrrrr000110RRRRR	GR[reg2]←saturated(GR[reg2]+GR[reg1])	1	1	1	×	×	×	×	×
	imm5,reg2	rrrrr010001iiiii	GR[reg2]←saturated(GR[reg2]+sign-extend(imm5)	1	1	1	×	×	×	×	×
SATSUB	reg1,reg2	rrrrr000101RRRRR	GR[reg2]←saturated(GR[reg2]–GR[reg1])	1	1	1	×	×	×	×	×
SATSUBI	imm16,reg1,reg2	rrrrr110011RRRRR iiiiiiiiiiiiiiii	GR[reg2]←saturated(GR[reg1]–sign-extend(imm16)	1	1	1	×	×	×	×	×
SATSUBR	reg1,reg2	rrrrr000100RRRRR	GR[reg2]←saturated(GR[reg1]–GR[reg2])	1	1	1	×	×	×	×	×
SETF	cccc,reg2	rrrrr1111110cccc 0000000000000000	If conditions are satisfied then GR[reg2]←00000001H else GR[reg2]←00000000H	1	1	1					
SET1	bit#3,disp16[reg1]	00bbb111110RRRRR dddddddddddddddd	adr←GR[reg1] + sign-extend(disp16) Z flag←Not (Load-memory-bit(adr,bit#3)) Store-memory-bit(adr,bit#3,1)	3 Note 3	3 Note 3	3 Note 3				×	
	reg2,[reg1]	rrrrr111111RRRRR 0000000011100000	adr←GR[reg1] Z flag←Not(Load-memory-bit(adr,reg2)) Store-memory-bit(adr,reg2,1)	3 Note 3	3 Note 3	3 Note 3				×	
SHL	reg1,reg2	rrrrr111111RRRRR 0000000011000000	GR[reg2]←GR[reg2] logically shift left by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010110iiiii	GR[reg2]←GR[reg2] logically shift left by zero-extend(imm5)	1	1	1	×	0	×	×	
SHR	reg1,reg2	rrrrr111111RRRRR 0000000010000000	GR[reg2]←GR[reg2] logically shift right by GR[reg1]	1	1	1	×	0	×	×	
	imm5,reg2	rrrrr010100iiiii	GR[reg2]←GR[reg2] logically shift right by zero-extend(imm5)	1	1	1	×	0	×	×	
SLD.B	disp7[ep],reg2	rrrrr0110ddddddd	adr←ep + zero-extend(disp7) GR[reg2]←sign-extend(Load-memory(adr,Byte))	1	1	Note 9					
SLD.BU	disp4[ep],reg2	rrrrr0000110dddd Note 18	adr←ep + zero-extend(disp4) GR[reg2]←zero-extend(Load-memory(adr,Byte))	1	1	Note 9					
SLD.H	disp8[ep],reg2	rrrrr1000ddddddd Note 19	adr←ep + zero-extend(disp8) GR[reg2]←sign-extend(Load-memory(adr,Half-word))	1	1	Note 9					
SLD.HU	disp5[ep],reg2	rrrrr0000111dddd Notes 18, 20	adr←ep+zero-extend(disp5) GR[reg2]←zero-extend(Load-memory(adr,Half-word))	1	1	Note 9					
SLD.W	disp8[ep],reg2	rrrrr1010dddddd0 Note 21	adr←ep + zero-extend(disp8) GR[reg2]←Load-memory(adr,Word)	1	1	Note 9					
SST.B	reg2,disp7[ep]	rrrrr0111ddddddd	adr←ep + zero-extend(disp7) Store-memory(adr,GR[reg2],Byte)	1	1	1					
SST.H	reg2,disp8[ep]	rrrrr1001ddddddd Note 19	adr←ep + zero-extend(disp8) Store-memory(adr,GR[reg2],Half-word)	1	1	1					
SST.W	reg2,disp8[ep]	rrrrr1010dddddd1 Note 21	adr←ep + zero-extend(disp8) Store-memory(adr,GR[reg2],Word)	1	1	1					
ST.B	reg2,disp16[reg1]	rrrrr111010RRRRR dddddddddddddddd	adr←GR[reg1] + sign-extend(disp16) Store-memory(adr,GR[reg2],Byte)	1	1	1					
ST.H	reg2,disp16[reg1]	rrrrr111011RRRRR ddddddddddddddd0 Note 8	adr←GR[reg1] + sign-extend(disp16) Store-memory (adr,GR[reg2], Half-word)	1	1	1					
ST.W	reg2,disp16[reg1]	rrrrr111011RRRRR ddddddddddddddd1 Note 8	adr←GR[reg1] + sign-extend(disp16) Store-memory (adr,GR[reg2], Word)	1	1	1					
STSR	regID,reg2	rrrrr111111RRRRR 0000000001000000	GR[reg2]←SR[regID]	1	1	1					
SUB	reg1,reg2	rrrrr001101RRRRR	GR[reg2]←GR[reg2]–GR[reg1]	1	1	1	×	×	×	×	
SUBR	reg1,reg2	rrrrr001100RRRRR	GR[reg2]←GR[reg1]–GR[reg2]	1	1	1	×	×	×	×	

(4/4)

Mnemonic	Operand	Opcode	Operation	Execution Clock			Flags				
				i	r	l	CY	OV	S	Z	SAT
SWITCH	reg1	`00000000010RRRRR`	adr←(PC+2) + (GR [reg1] logically shift left by 1) PC←(PC+2) + (sign-extend (Load-memory (adr,Half-word))) logically shift left by 1	5	5	5					
SXB	reg1	`00000000101RRRRR`	GR[reg1]←sign-extend (GR[reg1] (7 : 0))	1	1	1					
SXH	reg1	`00000000111RRRRR`	GR[reg1]←sign-extend (GR[reg1] (15 : 0))	1	1	1					

TRAP	vector	`00000111111iiiii`
		`0000000100000000`
			EIPC←PC+4 (Return PC)
			EIPSW←PSW
			ECR.EICC←Interrupt Code
			PSW.EP←1
			PSW.ID←1
			PC←00000040H (when vector is 00H to 0FH)
			00000050H (when vector is 10H to 1FH)
				3	3	3					

TST	reg1,reg2	`rrrrr001011RRRRR`	result←GR[reg2] AND GR[reg1]	1	1	1		0	×	×	

TST1	bit#3,disp16[reg1]	`11bbb111110RRRRR`
		`dddddddddddddddd`
			adr←GR[reg1] + sign-extend(disp16)
			Z flag←Not (Load-memory-bit (adr,bit#3))
				3 Note 3	3 Note 3	3 Note 3				×	

	reg2, [reg1]	`rrrrr111111RRRRR`
		`0000000011100110`
			adr←GR[reg1]
			Z flag←Not (Load-memory-bit (adr,reg2))
				3 Note 3	3 Note 3	3 Note 3				×	

XOR	reg1,reg2	`rrrrr001001RRRRR`	GR[reg2]←GR[reg2] XOR GR[reg1]	1	1	1		0	×	×	
XORI	imm16,reg1,reg2	`rrrrr110101RRRRR` `iiiiiiiiiiiiiiii`	GR[reg2]←GR[reg1] XOR zero-extend (imm16)	1	1	1		0	×	×	
ZXB	reg1	`00000000100RRRRR`	GR[reg1]←zero-extend (GR[reg1] (7 : 0))	1	1	1					
ZXH	reg1	`00000000110RRRRR`	GR[reg1]←zero-extend (GR[reg1] (15 : 0))	1	1	1					

Notes: 1. `dddddddd`: Higher 8 bits of disp9.

2. 3 clocks if the final instruction includes PSW write access.

3. If there is no wait state (3 + the number of read access wait states).

4. n is the total number of list X load registers. (According to the number of wait states. Also, if there are no wait states, n is the number of list X registers.)

5. RRRRR: other than 00000.

6. The lower half word data only are valid.

7. `ddddddddddddddddddddd`: The higher 21 bits of disp22.

8. `ddddddddddddddd`: The higher 15 bits of disp16.

9. According to the number of wait states (1 if there are no wait states).

10. b: bit 0 of disp16.

11. According to the number of wait states (2 if there are no wait states).

12. In this instruction, for convenience of mnemonic description, the source register is made reg2, but the reg1 field is used in the opcode. Therefore, the meaning of register specification in the mnemonic description and in the opcode differs from other instructions.

`rrrrr`= regID specification

RRRRR= reg2 specification

13. `iiiii`: Lower 5 bits of imm9.

`IIII`: Lower 4 bits of imm9.

14. In the case of reg2 = reg3 (the lower 32 bits of the results are not written in the register) or reg3 = r0 (the higher 32 bits of the results are not written in the register), shortened by 1 clock.

15. sp/imm: specified by bits 19 and 20 of the sub-opcode.

16. `ff = 00`: Load sp in ep.
`10`: Load sign expanded 16-bit immediate data (bits 47 to 32) in ep.
`11`: Load 32-bit immediate data (bits 63 to 32) in ep.

17. If imm = imm32, n + 3 clocks.

18. `rrrrr` : Other than 00000.

19. `ddddddd`: Higher 7 bits of disp8.

20. `dddd`: Higher 4 bits of disp5.

21. `dddddd`: Higher 6 bits of disp8.

Appendix B Index

A

A/D conversion result registers 0 to 11 423
A/D converter operation
  A/D trigger mode 431
  A/D trigger polling mode 433
  Basic operation 427
  Operation modes 428
A/D scan mode register 0 417
ADCR0 to ADCR11 423
ADETM 422
ADSCM0 417
ADSCM1 419
ANI00 to ANI11 43
AVDD 49, 50

C

Clocked serial interface initial transmission buffer registers 0, 1 284
Clocked serial interface initial transmission buffer registers L0, L1 285
Clocked serial interface mode registers 0, 1 . . . . . 277, 305
Clocked serial interface operation (CSI0, CSI1)
  Repeat transfer mode 295
  Single transfer mode 288, 310
Clocked serial interface reception buffer registers 0, 1 280
Clocked serial interface reception buffer registers L0, L1 281
Clocked serial interface transmission buffer registers 0, 1 . . . . . 282, 308
Clocked serial interface transmission buffer registers L0, L1 283
CSIC0, CSIC1 . . . . . 278, 306
CSIM0, CSIM1 277

D

DADC0 to DADC3 95
DCHC0 to DCHC3 96
DDIS 97
DMA addressing control registers 0 to 3 95
DMA channel control registers 0 to 3 96
DMA controller (DMAC) 89
  Bus cycle state transition 103
  Bus states 102
  Channel priorities 105
  Forcible interruption 106
  Forcible termination 107
  Next address setting function 105
  Precautions 107
  Single transfer mode 104
  Transfer end 107
  Transfer mode 104
  Transfer object 104
  Transfer start factors 106
  Transfer types 104
  Two-cycle transfer 104
DMA disable status register 97
DMA restart register 97
DMA trigger factor registers 0 to 3 . . . . . 98, 99, 100, 101
DN 336, 337, 341, 354, 365, 367, 371, 372, 378, 403
DRST 97

DTFR0 to DTFR3 . . . . 98, 99, 100, 101

F

FCAN

Baud rate setting 400

Data new flag 336, 337, 341, 354, 365, 367, 371, 372, 378, 403

Priority storage 336

Time stamp. 331, 332

M

MODE0 to MODE3 49

P

P20 to P27 39

Pin I/O Circuits 51

Precautions

A/D converter 435

DMA controller (DMAC) 107

S

Serial I/O shift registers 0, 1 . . . . 286, 309

Serial I/O shift registes L0, L1. 287

SIO0, SIO1 286

SIOL0, SIOL1 287

SOTB0, SOTB1 . . . . 282, 308

SOTBF0, SOTBF1 284

SOTBL0, SOTBL1 283

W

Watch timer

Operation 224

Watch timer mode control register 222

WTM 222

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 America Inc.
Corporate Communications Dept.
Fax: 1-800-729-9288
1-408-588-6130

Europe
NEC Electronics (Europe) GmbH
Market Communication Dept.
Fax: +49(0)-211-6503-1344

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	☐	☐	☐	☐

CS 99.1