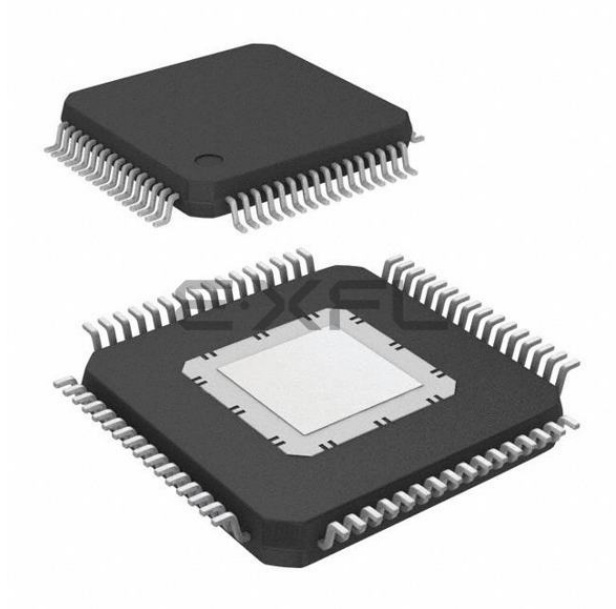


**Welcome to E-XFL.COM**

## What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

## Applications of "Embedded - Microcontrollers"

| Details | |
|---|---|
| Product Status | Obsolete |
| Core Processor | S12Z |
| Core Size | 16-Bit |
| Speed | 50MHz |
| Connectivity | LINbus, SCI, SPI |
| Peripherals | DMA, POR, PWM, WDT |
| Number of I/O | 31 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 512 x 8 |
| RAM Size | 8K x 8 |
| Voltage - Supply (Vcc/Vdd) | 3.5V ~ 40V |
| Data Converters | A/D 9x12b |
| Oscillator Type | Internal |
| Operating Temperature | -40°C ~ 105°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 64-LQFP Exposed Pad |
| Supplier Device Package | 64-HLQFP (10x10) |
| Purchase URL | https://www.e-xfl.com/product-detail/nxp-semiconductors/s912zvml12f1vkh |

**Table 1-5. Module Register Address Ranges**

| Address | Module | Size (Bytes) |
|---|---|---|
| 0x06E0–0x06EF | Reserved | 16 |
| 0x06F0–0x06F7 | BATS | 8 |
| 0x06F8–0x06FF | Reserved | 8 |
| 0x0700–0x0707 | SCI0 | 8 |
| 0x0708–0x070F | Reserved | 8 |
| 0x0710–0x0717 | SCI1 | 8 |
| 0x0718–0x077F | Reserved | 104 |
| 0x0780–0x0787 | SPI0 | 8 |
| 0x0788–0x07FF | Reserved | 120 |
| 0x0800–0x083F | CAN0 | 64 |
| 0x0840–0x097F | Reserved | 320 |
| 0x0980–0x0987 | LINPHY (S12ZVML derivatives) | 8 |
| 0x0980–0x0987 | HV Physical Interface (S12ZVM32, S12ZVM16 derivatives) | 8 |
| 0x0988–0x098F | Reserved | 8 |
| 0x0990–0x0997 | CANPHY (ZVMC256 only) | 8 |
| 0x0998–0x0FFF | Reserved | 1640 |

1. Reading from the first 16 locations in this reserved range returns undefined data

2. Address range = 0x0690-0x069F on Maskset N06E

**NOTE**

Reserved register space shown above is not allocated to any module. This register space is reserved for future use. Writing to these locations has no effect. Read access to these locations returns zero.

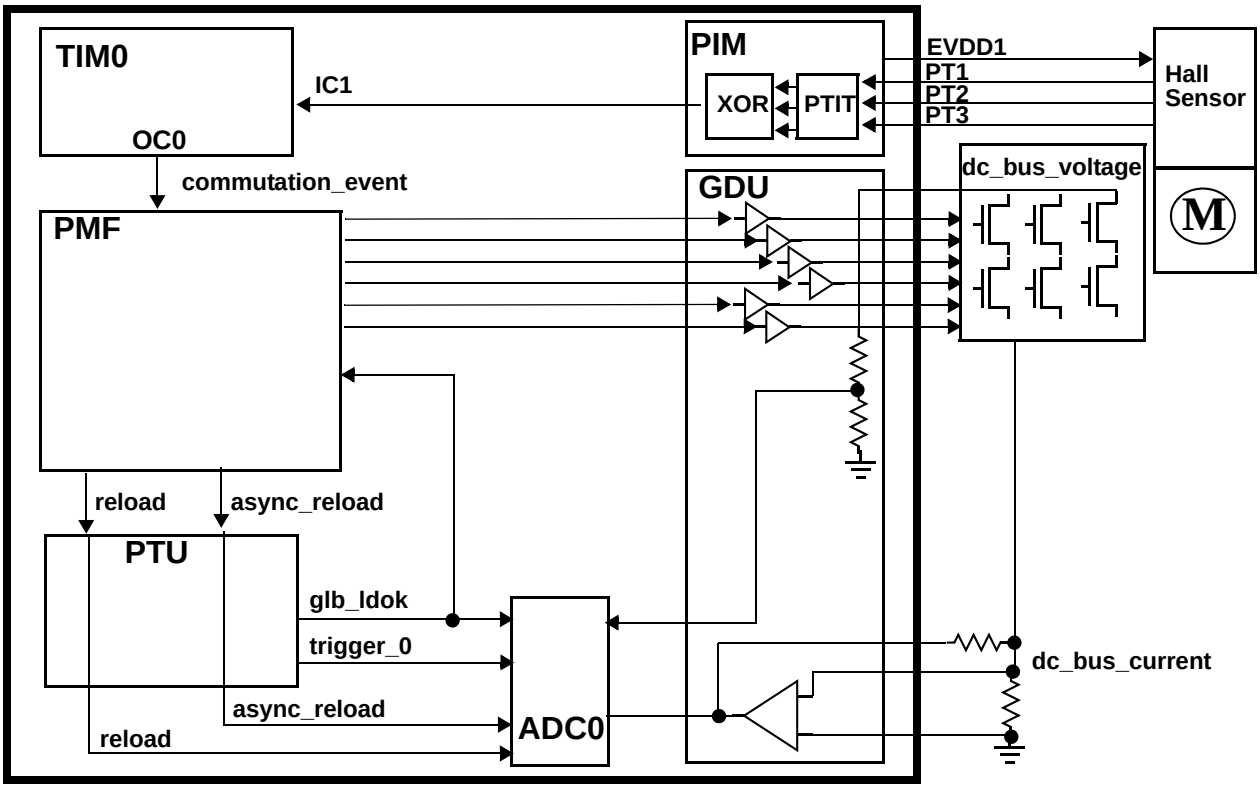
### 1.6.1 Flash Module

This device family instantiates different flash modules, depending on derivative. The flash documentation for the all devices is featured in the FTMRZ section.

## 1.13.5 BLDC Six-Step Commutation

### 1.13.5.1 Hall Sensor Triggered Commutation

**Figure 1-14. BLDC Configuration With Hall Sensors**

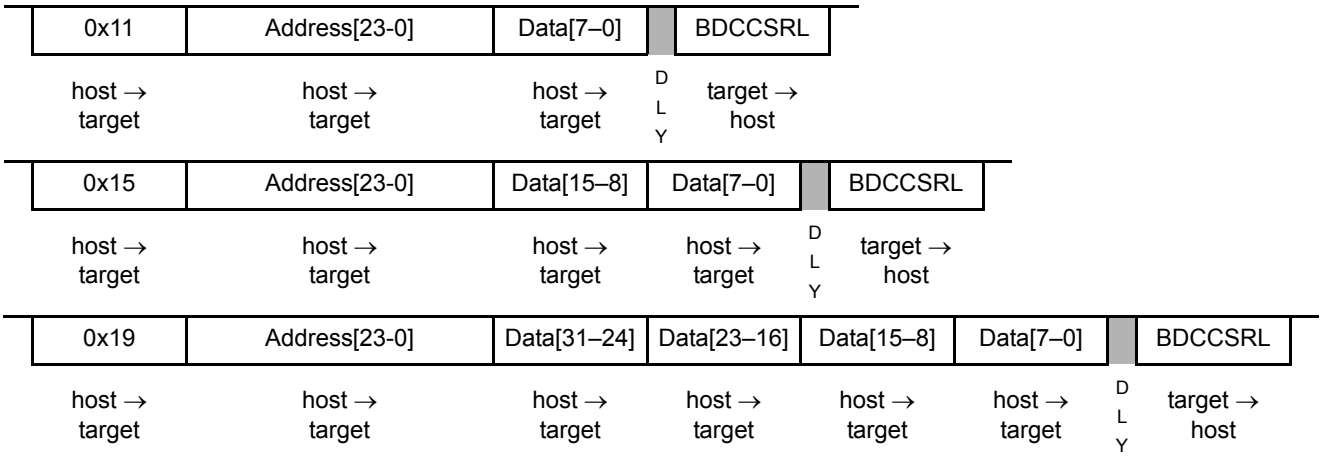



This BLDC application uses Hall sensor signals to create commutation triggers. The integrated sense amplifier and an ADC module are used to measure DC bus current, for torque calculation. The DC bus voltage measurement is used in the control algorithm to counter-modulate the PWM such that the variation of the DC-bus voltage does not affect the motor current closed loop. The configuration is as follows:

1. Connect the three Hall sensor signals from the motor to input pins PT3-1.
2. Set [T0IC1RR=1] in the register MODRR2 to establish the link from Hall sensor input pins to TIM input capture channel 1.
3. Setup TIM IC1 for speed measurement of XORed Hall sensor signals. Enable interrupt on both edges.
4. Enable TIM OC0 and select toggle action on output compare event: TCTL2[OM0:OL0]=01.
5. Configure PMF for edge-aligned PWM mode with or without restart at commutation: PMFENCx[RSTRT]. If using the restart option, then select generator A as reload signal source and keep the following configurations at their default setting: multi timebase generators (PMFCFG0[MTG]=b0), reload frequency (PMFFQCx[LDFQx]=b0), prescaler (PMFFQCx[PRSCx]=b00).
6. Enable PMF commutation event input: PMFCFG1[ENCE]=1.

**WRITE_MEM.sz_WS**

**Write memory at the specified address with status** **Non-intrusive**




Write data to the specified memory address. The address is transmitted as three 8-bit packets (msb to lsb) immediately after the command.

If the with-status option is specified, the status byte contained in BDCCSRL is returned after the write data. This status byte reflects the state after the memory write was performed. The examples show the WRITE_MEM.B{_WS}, WRITE_MEM.W{_WS}, and WRITE_MEM.L{_WS} commands. If enabled an ACK pulse is generated after the internal write access has been completed or aborted.

The hardware forces low-order address bits to zero longword accesses to ensure these accesses are on 0-modulo-size alignments. Byte alignment details are described in Section 5.4.5.2”.

### 5.4.4.17 WRITE_Rn

**Write general-purpose CPU register** **Active Background**


If the device is in active BDM, this command writes the 32-bit operand to the selected CPU general-purpose register. See Section 5.4.5.1 for the CRN details. Accesses to CPU registers are always 32-bits wide, regardless of implemented register width. If enabled an ACK pulse is generated after the internal write access has been completed or aborted.

If the device is not in active BDM, this command is rejected as an illegal operation, the ILLCMD bit is set and no operation is performed.

### 6.3.2.18 Debug Comparator C Control Register (DBGCCTL)

Address: 0x0130

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 6-20. Debug Comparator C Control Register**

Read: Anytime.

Write: If DBG not armed and PTACT is clear.

**Table 6-34. DBGCCTL Field Descriptions**

| Field | Description |
|---|---|
| 6<br>NDB | **Not Data Bus** — The NDB bit controls whether the match occurs when the data bus matches the comparator register value or when the data bus differs from the register value. This bit is ignored if the INST bit in the same register is set.<br>0 Match on data bus equivalence to comparator register contents<br>1 Match on data bus difference to comparator register contents |
| 5<br>INST | **Instruction Select** — This bit configures the comparator to compare PC or data access addresses.<br>0 Comparator compares addresses of data accesses<br>1 Comparator compares PC address |
| 3<br>RW | **Read/Write Comparator Value Bit** — The RW bit controls whether read or write is used in compare for the associated comparator. The RW bit is ignored if RWE is clear or INST is set.<br>0 Write cycle is matched<br>1 Read cycle is matched |
| 2<br>RWE | **Read/Write Enable Bit** — The RWE bit controls whether read or write comparison is enabled for the associated comparator. This bit is not used if INST is set.<br>0 Read/Write is not used in comparison<br>1 Read/Write is used in comparison |
| 0<br>COMPE | **Enable Bit** — Determines if comparator is enabled<br>0 The comparator is not enabled<br>1 The comparator is enabled |

Table 6-35 shows the effect for RWE and RW on the comparison conditions. These bits are ignored if INST is set, because matches based on opcodes reaching the execution stage are data independent.

**Table 6-35. Read or Write Comparison Logic Table**

| RWE Bit | RW Bit | RW Signal | Comment |
|---|---|---|---|
| 0 | x | 0 | RW not used in comparison |
| 0 | x | 1 | RW not used in comparison |
| 1 | 0 | 0 | Write match |
| 1 | 0 | 1 | No match |
| 1 | 1 | 0 | No match |

### 8.3.2.23 S12CPMU_UHV_V10_V6 IRC1M Trim Registers (CPMUIRCTRIMH / CPMUIRCTRIML)

Module Base + 0x0018

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| R | TCTRIM[4:0] | | | | | 0 | IRCTRIM[9:8] | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | 0 | F | F |

After de-assert of System Reset a factory programmed trim value is automatically loaded from the Flash memory to provide trimmed Internal Reference Frequency $f_{IRC1M\_TRIM}$.

**Figure 8-31. S12CPMU_UHV_V10_V6 IRC1M Trim High Register (CPMUIRCTRIMH)**

Module Base + 0x0019

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | IRCTRIM[7:0] | | | | | | | |
| W | | | | | | | | |
| Reset | F | F | F | F | F | F | F | F |

After de-assert of System Reset a factory programmed trim value is automatically loaded from the Flash memory to provide trimmed Internal Reference Frequency $f_{IRC1M\_TRIM}$.

**Figure 8-32. S12CPMU_UHV_V10_V6 IRC1M Trim Low Register (CPMUIRCTRIML)**

Read: Anytime

Write: Anytime if PROT=0 (CPMUPROT register). Else write has no effect

**NOTE**

Writes to these registers while PLLSEL=1 clears the LOCK and UPOSC status bits.

**Table 8-27. CPMUIRCTRIMH/L Field Descriptions**

| Field | Description |
|---|---|
| 15-11<br>TCTRIM[4:0] | IRC1M temperature coefficient Trim Bits<br>Trim bits for the Temperature Coefficient (TC) of the IRC1M frequency.<br>Table 8-28 shows the influence of the bits TCTRIM[4:0] on the relationship between frequency and temperature.<br>Figure 8-34 shows an approximate TC variation, relative to the nominal TC of the IRC1M (i.e. for TCTRIM[4:0]=0x00000 or 0x10000). |
| 9-0<br>IRCTRIM[9:0] | **IRC1M Frequency Trim Bits** — Trim bits for Internal Reference Clock<br>After System Reset the factory programmed trim value is automatically loaded into these registers, resulting in a Internal Reference Frequency $f_{IRC1M\_TRIM}$.See device electrical characteristics for value of $f_{IRC1M\_TRIM}$.<br>The frequency trimming consists of two different trimming methods:<br>A rough trimming controlled by bits IRCTRIM[9:6] can be done with frequency leaps of about 6% in average.<br>A fine trimming controlled by bits IRCTRIM[5:0] can be done with frequency leaps of about 0.3% (this trimming determines the precision of the frequency setting of 0.15%, i.e. 0.3% is the distance between two trimming values).<br>Figure 8-33 shows the relationship between the trim bits and the resulting IRC1M frequency. |

### 8.3.2.26 Reserved Register CPMUTEST2

**NOTE**

This reserved register is designed for factory test purposes only, and is not intended for general user access. Writing to this register when in Special Mode can alter the S12CPMU_UHV_V10_V6's functionality.

Module Base + 0x001C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 8-37. Reserved Register CPMUTEST2**

Read: Anytime

Write: Only in Special Mode

### 9.5.2.5 ADC Format Register (ADCFMT)

Module Base + 0x0004

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | DJM | 0 | 0 | 0 | 0 | SRES[2:0] | | |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 9-8. ADC Format Register (ADCFMT)**

Read: Anytime

Write: Bits DJM and SRES[2:0] are writable if bit ADC_EN clear or bit SMOD_ACC set

**Table 9-8. ADCFMT Field Descriptions**

| Field | Description |
|---|---|
| 7<br>DJM | **Result Register Data Justification** — Conversion result data format is always unsigned. This bit controls justification of conversion result data in the conversion result list.<br>0 Left justified data in the conversion result list.<br>1 Right justified data in the conversion result list. |
| 2-0<br>SRES[2:0] | **ADC Resolution Select** — These bits select the resolution of conversion results. See Table 9-9 for coding. |

**Table 9-9. Selectable Conversion Resolution**

| SRES[2] | SRES[1] | SRES[0] | ADC Resolution |
|---|---|---|---|
| 0 | 0 | 0 | 8-bit data |
| 0 | 0 | 1 | Reserved[1.] |
| 0 | 1 | 0 | 10-bit data |
| 0 | 1 | 1 | Reserved[1.] |
| 1 | 0 | 0 | 12-bit data |
| 1 | x | x | Reserved[(1)] |

1. Reserved settings cause a severe error at ADC conversion start whereby the CMD_EIF flag is set and ADC ceases operation

**Table 13-23. CANIDMR0–CANIDMR3 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AM[7:0] | **Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.<br>0 Match corresponding acceptance code register and identifier bits<br>1 Ignore corresponding acceptance code register bit |

Module Base + 0x001C to Module Base + 0x001F — Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | AM7 | AM6 | AM5 | AM4 | AM3 | AM2 | AM1 | AM0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-23. MSCAN Identifier Mask Registers (Second Bank) — CANIDMR4–CANIDMR7**

1. Read: Anytime
   Write: Anytime in initialization mode (INITRQ = 1 and INITAK = 1)

**Table 13-24. CANIDMR4–CANIDMR7 Register Field Descriptions**

| Field | Description |
|---|---|
| 7-0<br>AM[7:0] | **Acceptance Mask Bits** — If a particular bit in this register is cleared, this indicates that the corresponding bit in the identifier acceptance register must be the same as its identifier bit before a match is detected. The message is accepted if all such bits match. If a bit is set, it indicates that the state of the corresponding bit in the identifier acceptance register does not affect whether or not the message is accepted.<br>0 Match corresponding acceptance code register and identifier bits<br>1 Ignore corresponding acceptance code register bit |

## 13.3.3 Programmer’s Model of Message Storage

The following section details the organization of the receive and transmit message buffers and the associated control registers.

To simplify the programmer interface, the receive and transmit message buffers have the same outline. Each message buffer allocates 16 bytes in the memory map containing a 13 byte data structure.

An additional transmit buffer priority register (TBPR) is defined for the transmit buffers. Within the last two bytes of this memory map, the MSCAN stores a special 16-bit time stamp, which is sampled from an internal timer after successful transmission or reception of a message. This feature is only available for transmit and receiver buffers, if the TIME bit is set (see Section 13.3.2.1, “MSCAN Control Register 0 (CANCTL0)”).

The time stamp register is written by the MSCAN. The CPU can only read these registers.

Figure 13-40 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces a filter 0 hit. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces a filter 1 hit.

- Four identifier acceptance filters, each to be applied to:
  - The 14 most significant bits of the extended identifier plus the SRR and IDE bits of CAN 2.0B messages.
  - The 11 bits of the standard identifier, the RTR and IDE bits of CAN 2.0A/B messages. Figure 13-41 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 and 1 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 2 and 3 hits.
- Eight identifier acceptance filters, each to be applied to the first 8 bits of the identifier. This mode implements eight independent filters for the first 8 bits of a CAN 2.0A/B compliant standard identifier or a CAN 2.0B compliant extended identifier.
  Figure 13-42 shows how the first 32-bit filter bank (CANIDAR0–CANIDAR3, CANIDMR0–CANIDMR3) produces filter 0 to 3 hits. Similarly, the second filter bank (CANIDAR4–CANIDAR7, CANIDMR4–CANIDMR7) produces filter 4 to 7 hits.
- Closed filter. No CAN message is copied into the foreground buffer RxFG, and the RXF flag is never set.

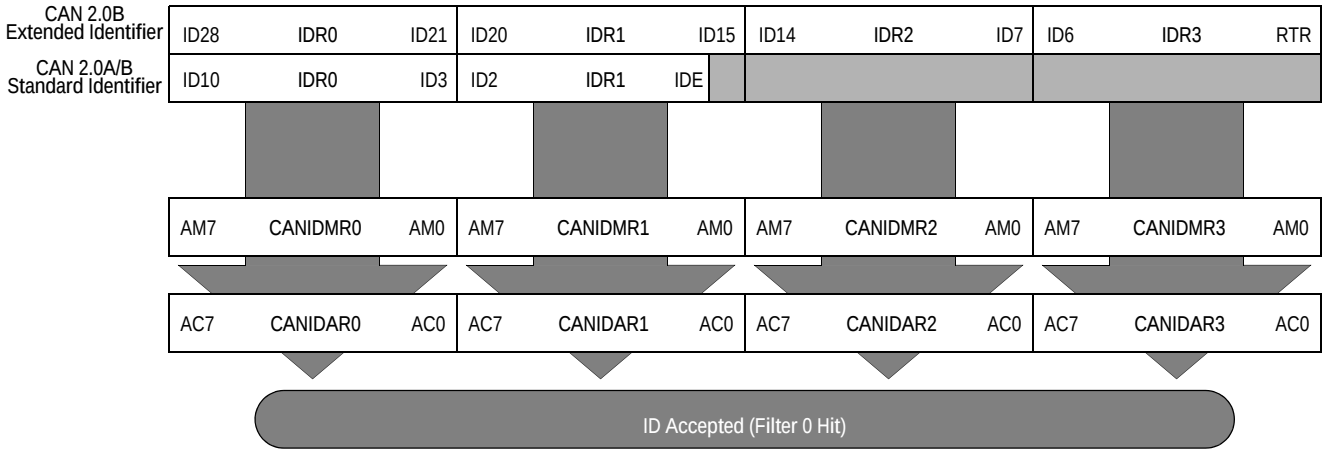



**Figure 13-40. 32-bit Maskable Identifier Acceptance Filter**

**Table 17-4. SPICR2 Field Descriptions**

| Field | Description |
|---|---|
| 6<br>XFRW | **Transfer Width —** This bit is used for selecting the data transfer width. If 8-bit transfer width is selected, SPIDRL becomes the dedicated data register and SPIDRH is unused. If 16-bit transfer width is selected, SPIDRH and SPIDRL form a 16-bit data register. Please refer to Section 17.3.2.4, “SPI Status Register (SPISR) for information about transmit/receive data handling and the interrupt flag clearing mechanism. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 8-bit Transfer Width (n = 8)(1)<br>1 16-bit Transfer Width (n = 16)[1] |
| 4<br>MODFEN | **Mode Fault Enable Bit** — This bit allows the MODF failure to be detected. If the SPI is in master mode and MODFEN is cleared, then the $\overline{SS}$ port pin is not used by the SPI. In slave mode, the $\overline{SS}$ is available only as an input regardless of the value of MODFEN. For an overview on the impact of the MODFEN bit on the $\overline{SS}$ port pin configuration, refer to Table 17-3. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state.<br>0 $\overline{SS}$ port pin is not used by the SPI.<br>1 $\overline{SS}$ port pin with MODF feature. |
| 3<br>BIDIROE | **Output Enable in the Bidirectional Mode of Operation** — This bit controls the MOSI and MISO output buffer of the SPI, when in bidirectional mode of operation (SPC0 is set). In master mode, this bit controls the output buffer of the MOSI port, in slave mode it controls the output buffer of the MISO port. In master mode, with SPC0 set, a change of this bit will abort a transmission in progress and force the SPI into idle state.<br>0 Output buffer disabled.<br>1 Output buffer enabled. |
| 1<br>SPISWAI | **SPI Stop in Wait Mode Bit** — This bit is used for power conservation while in wait mode.<br>0 SPI clock operates normally in wait mode.<br>1 Stop SPI clock generation when in wait mode. |
| 0<br>SPC0 | **Serial Pin Control Bit 0** — This bit enables bidirectional pin configurations as shown in Table 17-5. In master mode, a change of this bit will abort a transmission in progress and force the SPI system into idle state. |

1. n is used later in this document as a placeholder for the selected transfer width.

**Table 17-5. Bidirectional Pin Configurations**

| Pin Mode | SPC0 | BIDIROE | MISO | MOSI |
|---|---|---|---|---|
| **Master Mode of Operation** | | | | |
| Normal | 0 | X | Master In | Master Out |
| Bidirectional | 1 | 0 | MISO not used by SPI | Master In |
| | | 1 | | Master I/O |
| **Slave Mode of Operation** | | | | |
| Normal | 0 | X | Slave Out | Slave In |
| Bidirectional | 1 | 0 | Slave In | MOSI not used by SPI |
| | | 1 | Slave I/O | |

**Table 18-6. GDUDSE Register Field Descriptions**

| Field | Description |
|---|---|
| 6-4<br>GDHSIF[2:0] | GDU High-Side Driver Desaturation Interrupt Flags — The flag is set by hardware to “1” when a desaturation error on associated high-side driver pin HS[2:0] occurs. If the GDSEIE bit is set an interrupt is requested. Writing a logic “1” to the bit field clears the flag.<br>0 No desaturation error on high-side driver<br>1 Desaturation error on high-side driver |
| 2-0<br>GDLSIF[2:0] | GDU Low-Side Driver Desaturation Interrupt Flag — The flag is set to “1” when a desaturation error on associated low-side driver pin LS[2:0] occurs. If the GDSEIE bit is set an interrupt is requested. Writing a logic “1” to the bit field clears the flag.<br>0 No desaturation error on low-side driver<br>1 Desaturation error on low-side driver |

**Table 20-17. FSTAT Field Descriptions (continued)**

| Field | Description |
|---|---|
| 2<br>RSVD | **Reserved Bit** — This bit is reserved and always reads 0. |
| 1–0<br>MGSTAT[1:0] | **Memory Controller Command Completion Status Flag** — One or more MGSTAT flag bits are set if an error is detected during execution of a Flash command or during the Flash reset sequence. The MGSTAT bits are cleared automatically at the start of the execution of a Flash command. See Section 20.4.7, “Flash Command Description,” and Section 20.6, “Initialization” for details. |

## 20.3.2.8 Flash Error Status Register (FERSTAT)

The FERSTAT register reflects the error status of internal Flash operations.

Offset Module Base + 0x0007

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | 0 | 0 | DFDF | SFDIF |
| W | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

= Unimplemented or Reserved

**Figure 20-12. Flash Error Status Register (FERSTAT)**

All flags in the FERSTAT register are readable and only writable to clear the flag.

**Table 20-18. FERSTAT Field Descriptions**

| Field | Description |
|---|---|
| 1<br>DFDF | **Double Bit Fault Detect Flag** — The setting of the DFDF flag indicates that a double bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation returning invalid data was attempted on a Flash block that was under a Flash command operation.(1) The DFDF flag is cleared by writing a 1 to DFDF. Writing a 0 to DFDF has no effect on DFDF.(2)<br>0 No double bit fault detected<br>1 Double bit fault detected or a Flash array read operation returning invalid data was attempted while command running. See Section 20.4.3, “Flash Block Read Access” for details |
| 0<br>SFDIF | **Single Bit Fault Detect Interrupt Flag** — With the IGNSF bit in the FCNFG register clear, the SFDIF flag indicates that a single bit fault was detected in the stored parity and data bits during a Flash array read operation or that a Flash array read operation returning invalid data was attempted on a Flash block that was under a Flash command operation. The SFDIF flag is cleared by writing a 1 to SFDIF. Writing a 0 to SFDIF has no effect on SFDIF.<br>0 No single bit fault detected<br>1 Single bit fault detected and corrected or a Flash array read operation returning invalid data was attempted while command running |

1. In case of ECC errors the corresponding flag must be cleared for the proper setting of any further error, i.e. any new error will only be indicated properly when DFDF and/or SFDIF are clear at the time the error condition is detected.

2. There is a one cycle delay in storing the ECC DFDF and SFDIF fault flags in this register. At least one NOP is required after a flash memory read before checking FERSTAT for the occurrence of ECC errors.

### 20.3.2.13 Flash Common Command Object Registers (FCCOB)

The FCCOB is an array of six words. Byte wide reads and writes are allowed to the FCCOB registers.

Offset Module Base + 0x000C

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-18. Flash Common Command Object 0 High Register (FCCOB0HI)**

Offset Module Base + 0x000D

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-19. Flash Common Command Object 0 Low Register (FCCOB0LO)**

Offset Module Base + 0x000E

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-20. Flash Common Command Object 1 High Register (FCCOB1HI)**

Offset Module Base + 0x000F

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[7:0] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-21. Flash Common Command Object 1 Low Register (FCCOB1LO)**

Offset Module Base + 0x0010

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | CCOB[15:8] | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 20-22. Flash Common Command Object 2 High Register (FCCOB2HI)**

### 20.4.7.17 Protection Override Command

The Protection Override command allows the user to temporarily override the protection limits, either decreasing, increasing or disabling protection limits, on P-Flash and/or EEPROM, if the comparison key provided as a parameter loaded on FCCOB matches the value of the key previously programmed on the Flash Configuration Field (see Table 20-4). The value of the Protection Override Comparison Key must not be 16'hFFFF, that is considered invalid and if used as argument will cause the Protection Override feature to be disabled. Any valid key value that does not match the value programmed in the Flash Configuration Field will cause the Protection Override feature to be disabled. Current status of the Protection Override feature can be observed on FPSTAT FPOVRD bit (see Section 20.3.2.4, "Flash Protection Status Register (FPSTAT)).

**Table 20-68. Protection Override Command FCCOB Requirements**

| Register | FCCOB Parameters | |
|---|---|---|
| FCCOB0 | 0x13 | Protection Update Selection [1:0] See Table 20-69 |
| FCCOB1 | Comparison Key | |
| FCCOB2 | reserved | New FPROT value |
| FCCOB3 | reserved | New DFPROT value |

**Table 20-69. Protection Override selection description**

| Protection Update Selection code [1:0] | Protection register selection |
|---|---|
| bit 0 | Update P-Flash protection<br>0 - keep unchanged (do not update)<br>1 - update P-Flash protection with new FPROT value loaded on FCCOB |
| bit 1 | Update EEPROM protection<br>0 - keep unchanged (do not update)<br>1 - update EEPROM protection with new DFPROT value loaded on FCCOB |

If the comparison key successfully matches the key programmed in the Flash Configuration Field the Protection Override command will preserve the current values of registers FPROT and DFPROT stored in an internal area and will override these registers as selected by the Protection Update Selection field with the value(s) loaded on FCCOB parameters. The new values loaded into FPROT and/or DFPROT can reconfigure protection without any restriction (by increasing, decreasing or disabling protection limits). If the command executes successfully the FPSTAT FPOVRD bit will set.

If the comparison key does not match the key programmed in the Flash Configuration Field, or if the key loaded on FCCOB is 16'hFFFF, the value of registers FPROT and DFPROT will be restored to their original contents before executing the Protection Override command and the FPSTAT FPOVRD bit will be cleared. If the contents of the Protection Override Comparison Key in the Flash Configuration Field is left in the erased state (i.e. 16'hFFFF) the Protection Override feature is permanently disabled. If the command execution is flagged as an error (ACCERR being set for incorrect command launch) the values of FPROT and DFPROT will not be modified.

### 20.4.8.1 Description of Flash Interrupt Operation

The Flash module uses the CCIF flag in combination with the CCIE interrupt enable bit to generate the Flash command interrupt request. The Flash module uses the SFDIF flag in combination with the SFDIE interrupt enable bits to generate the Flash error interrupt request. For a detailed description of the register bits involved, refer to Section 20.3.2.5, “Flash Configuration Register (FCNFG)”, Section 20.3.2.6, “Flash Error Configuration Register (FERCNFG)”, Section 20.3.2.7, “Flash Status Register (FSTAT)”, and Section 20.3.2.8, “Flash Error Status Register (FERSTAT)”.

The logic used for generating the Flash module interrupts is shown in Figure 20-31.

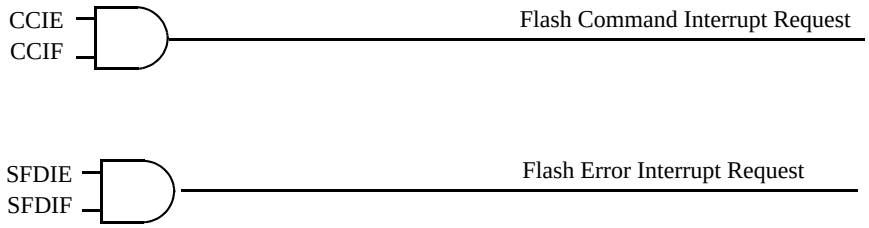



**Figure 20-31. Flash Module Interrupts Implementation**

## 20.4.9 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 20.4.8, “Interrupts”).

## 20.4.10 Stop Mode

If a Flash command is active (CCIF = 0) when the MCU requests stop mode, the current Flash operation will be completed before the MCU is allowed to enter stop mode.

# 20.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 20-11). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0xFF_FE0F. The security state out of reset can be permanently changed by programming the security byte assuming that the MCU is starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

- Unsecuring the MCU using Backdoor Key Access
- Unsecuring the MCU in Special Single Chip Mode using BDM
- Mode and Security Effects on Flash Command Availability

## 21.4.2 Register Descriptions

This section describes all CAN Physical Layer registers and their individual bits.

### 21.4.2.1 Port CP Data Register (CPDR)

Module Base + 0x0000 Access: User read/write(1)

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R | CPDR7 | 0 | 0 | 0 | 0 | 0 | CPDR1 | CPDR0 |
| W | | | | | | | | |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |

**Figure 21-2. Port CP Data Register (CPDR)**

1. Read: Anytime
   Write: Anytime

**Table 21-4. CPDR Register Field Descriptions**

| Field | Description |
|---|---|
| 7<br>CPDR7 | Port CP Data Bit 7<br>Read-only bit. The synchronized CAN Physical Layer wake-up receiver output can be read at any time. |
| 1<br>CPDR1 | Port CP Data Bit 1<br>The CAN Physical Layer CPTXD input can be directly controlled through this register bit if routed here (see device-level specification). In this case the register bit value is driven to the pin.<br><br>0 CPTXD is driven low (dominant)<br>1 CPTXD is driven high (recessive) |
| 0<br>CPDR0 | Port CP Data Bit 0<br>Read-only bit. The synchronized CAN Physical Layer CPRXD output state can be read at any time. |

The clock source of each PWM channel is determined by PCLKx bits in PWMCLK (see Section 22.3.2.3, “PWM Clock Select Register (PWMCLK)) and PCLKABx bits in PWMCLKAB as shown in Table 22-5 and Table 22-6.

### 22.3.2.8 PWM Scale A Register (PWMSCLA)

PWMSCLA is the programmable scale value used in scaling clock A to generate clock SA. Clock SA is generated by taking clock A, dividing it by the value in the PWMSCLA register and dividing that by two.

Clock SA = Clock A / (2 * PWMSCLA)

**NOTE**

When PWMSCLA = $00, PWMSCLA value is considered a full scale value of 256. Clock A is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLA).

Module Base + 0x0008

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-10. PWM Scale A Register (PWMSCLA)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLA value)

### 22.3.2.9 PWM Scale B Register (PWMSCLB)

PWMSCLB is the programmable scale value used in scaling clock B to generate clock SB. Clock SB is generated by taking clock B, dividing it by the value in the PWMSCLB register and dividing that by two.

Clock SB = Clock B / (2 * PWMSCLB)

**NOTE**

When PWMSCLB = $00, PWMSCLB value is considered a full scale value of 256. Clock B is thus divided by 512.

Any value written to this register will cause the scale counter to load the new scale value (PWMSCLB).

Module Base + 0x0009

| | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| R<br>W | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 22-11. PWM Scale B Register (PWMSCLB)**

Read: Anytime

Write: Anytime (causes the scale counter to load the PWMSCLB value).

# M.4 0x0100-0x017F S12ZDBG

| Address | Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0103 | DBGTCRL[2] | R | 0 | 0 | 0 | PREND[(1)] | DSTAMP | PDOE | PROFILE | STAMP |
| | | W | | | | | | | | |
| 0x0104 | DBGTBH[2] | R | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x0105 | DBGTBL[2] | R | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
| | | W | | | | | | | | |
| 0x0106 | DBGCNT[2] | R | 0 | CNT | | | | | | |
| | | W | | | | | | | | |
| 0x0107 | DBGSCR1 | R | C3SC1 | C3SC0 | C2SC1[2] | C2SC0[2] | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x0108 | DBGSCR2 | R | C3SC1 | C3SC0 | C2SC1[2] | C2SC0[2] | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x0109 | DBGSCR3 | R | C3SC1 | C3SC0 | C2SC1[2] | C2SC0[2] | C1SC1 | C1SC0 | C0SC1 | C0SC0 |
| | | W | | | | | | | | |
| 0x010A | DBGEFR | R | PTBOVF[2] | TRIGF | 0 | EEVF | ME3 | ME2[2] | ME1 | ME0 |
| | | W | | | | | | | | |
| 0x010B | DBGSR | R | TBF[2] | 0 | 0 | PTACT[2] | 0 | SSF2 | SSF1 | SSF0 |
| | | W | | | | | | | | |
| 0x010C-0x010F | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0110 | DBGACTL | R | 0 | NDB | INST | 0 | RW | RWE | reserved | COMPE |
| | | W | | | | | | | | |
| 0x0111-0x0114 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0115 | DBGAAH | R | DBGAA[23:16] | | | | | | | |
| | | W | | | | | | | | |
| 0x0116 | DBGAAM | R | DBGAA[15:8] | | | | | | | |
| | | W | | | | | | | | |
| 0x0117 | DBGAAL | R | DBGAA[7:0] | | | | | | | |
| | | W | | | | | | | | |
| 0x0118 | DBGAD0 | R | Bit 31 | 30 | 29 | 28 | 27 | 26 | 25 | Bit 24 |
| | | W | | | | | | | | |
| 0x0119 | DBGAD1 | R | Bit 23 | 22 | 21 | 20 | 19 | 18 | 17 | Bit 16 |
| | | W | | | | | | | | |
| 0x011A | DBGAD2 | R | Bit 15 | 14 | 13 | 12 | 11 | 10 | 9 | Bit 8 |
| | | W | | | | | | | | |
| 0x011B | DBGAD3 | R | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
| | | W | | | | | | | | |

| Global Address | Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0284 | DDRADH | R | DDRADH7[2] | DDRADH6[2] | DDRADH5[2] | DDRADH4[2] | DDRADH3[2] | DDRADH2[2] | DDRADL1[2] | DDRADH0 |
| | | W | | | | | | | | |
| 0x0285 | DDRADL | R | DDRADL7 | DDRADL6 | DDRADL5 | DDRADL4 | DDRADL3 | DDRADL2 | DDRADL1 | DDRADL0 |
| | | W | | | | | | | | |
| 0x0286 | PERADH | R | PERADH7[2] | PERADH6[2] | PERADH5[2] | PERADH4[2] | PERADH3[2] | PERADH2[2] | PERADH1[2] | PERADH0 |
| | | W | | | | | | | | |
| 0x0287 | PERADL | R | PERADL7 | PERADL6 | PERADL5 | PERADL4 | PERADL3 | PERADL2 | PERADL1 | PERADL0 |
| | | W | | | | | | | | |
| 0x0288 | PPSADH | R | PPSADH7[2] | PPSADH6[2] | PPSADH5[2] | PPSADH4[2] | PPSADH3[2] | PPSADH2[2] | PPSADH1[2] | PPSADH0 |
| | | W | | | | | | | | |
| 0x0289 | PPSADL | R | PPSADL7 | PPSADL6 | PPSADL5 | PPSADL4 | PPSADL3 | PPSADL2 | PPSADL1 | PPSADL0 |
| | | W | | | | | | | | |
| 0x028A–0x028B | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x028C | PIEADH | R | PIEADH7[2] | PIEADH6[2] | PIEADH5[2] | PIEADH4[2] | PIEADH3[2] | PIEADH2[2] | PIEADH1[2] | PIEADH0 |
| | | W | | | | | | | | |
| 0x028D | PIEADL | R | PIEADL7 | PIEADL6 | PIEADL5 | PIEADL4 | PIEADL3 | PIEADL2 | PIEADL1 | PIEADL0 |
| | | W | | | | | | | | |
| 0x028E | PIFADH | R | PIFADH7[2] | PIFADH6[2] | PIFADH5[2] | PIFADH4[2] | PIFADH3[2] | PIFADH2[2] | PIFADH1[2] | PIFADH0 |
| | | W | | | | | | | | |
| 0x028F | PIFADL | R | PIFADL7 | PIFADL6 | PIFADL5 | PIFADL4 | PIFADL3 | PIFADL2 | PIFADL1 | PIFADL0 |
| | | W | | | | | | | | |
| 0x0290–0x0297 | Reserved | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | | W | | | | | | | | |
| 0x0298 | DIENADH | R | DIENADH7[2] | DIENADH6[2] | DIENADH5[2] | DIENADH4[2] | DIENADH3[2] | DIENADH2[2] | DIENADH1[2] | DIENADH0 |
| | | W | | | | | | | | |
| 0x0299 | DIENADL | R | DIENADL7 | DIENADL6 | DIENADL5 | DIENADL4 | DIENADL3 | DIENADL2 | DIENADL1 | DIENADL0 |
| | | W | | | | | | | | |