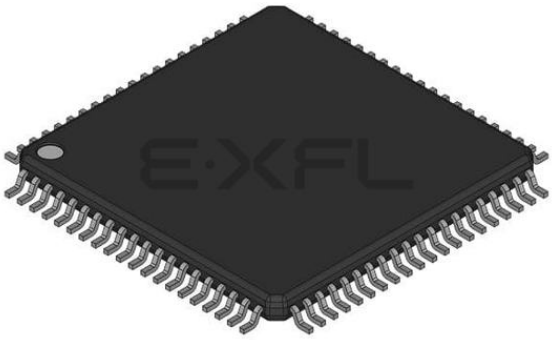

**Welcome to E-XFL.COM**

## 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 | Active |
| Core Processor | HCS12X |
| Core Size | 16-Bit |
| Speed | 50MHz |
| Connectivity | CANbus, EBI/EMI, I²C, IrDA, SCI, SPI |
| Peripherals | LVD, POR, PWM, WDT |
| Number of I/O | 59 |
| Program Memory Size | 128KB (128K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | 2K x 8 |
| RAM Size | 12K x 8 |
| Voltage - Supply (Vcc/Vdd) | 1.72V ~ 5.5V |
| Data Converters | A/D 8x12b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 125°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 80-QFP |
| Supplier Device Package | 80-QFP (14x14) |
| Purchase URL | https://www.e-xfl.com/pro/item?MUrl=&PartUrl=s912xeg128j2maa |

# Blank Page

**Table 2-3. Pin Configuration Summary**

| DDR | IO | RDR | PE | PS[1] | IE[2] | Function | Pull Device | Interrupt |
|---|---|---|---|---|---|---|---|---|
| 0 | x | x | 0 | x | 0 | Input | Disabled | Disabled |
| 0 | x | x | 1 | 0 | 0 | Input | Pull Up | Disabled |
| 0 | x | x | 1 | 1 | 0 | Input | Pull Down | Disabled |
| 0 | x | x | 0 | 0 | 1 | Input | Disabled | Falling edge |
| 0 | x | x | 0 | 1 | 1 | Input | Disabled | Rising edge |
| 0 | x | x | 1 | 0 | 1 | Input | Pull Up | Falling edge |
| 0 | x | x | 1 | 1 | 1 | Input | Pull Down | Rising edge |
| 1 | 0 | 0 | x | x | 0 | Output, full drive to 0 | Disabled | Disabled |
| 1 | 1 | 0 | x | x | 0 | Output, full drive to 1 | Disabled | Disabled |
| 1 | 0 | 1 | x | x | 0 | Output, reduced drive to 0 | Disabled | Disabled |
| 1 | 1 | 1 | x | x | 0 | Output, reduced drive to 1 | Disabled | Disabled |
| 1 | 0 | 0 | x | 0 | 1 | Output, full drive to 0 | Disabled | Falling edge |
| 1 | 1 | 0 | x | 1 | 1 | Output, full drive to 1 | Disabled | Rising edge |
| 1 | 0 | 1 | x | 0 | 1 | Output, reduced drive to 0 | Disabled | Falling edge |
| 1 | 1 | 1 | x | 1 | 1 | Output, reduced drive to 1 | Disabled | Rising edge |

1. Always “0” on Port A, B, C, D, E, K, AD0, and AD1.
2. Applicable only on Port P, H, and J.

**NOTE**

All register bits in this module are completely synchronous to internal clocks during a register read.

This register controls input pin threshold level and determines the external address and data bus sizes in normal expanded mode. If not in use with the external bus interface, the related pins can be used for alternative functions.

External bus is available as programmed in normal expanded mode and always full-sized in emulation modes and special test mode; function not available in single-chip modes.

**Table 5-3. EBICTL0 Field Descriptions**

| Field | Description |
|---|---|
| 7<br>ITHRS | **Reduced Input Threshold** — This bit selects reduced input threshold on external data bus pins and specific control input signals which are in use with the external bus interface in order to adapt to external devices with a 3.3 V, 5 V tolerant I/O.<br>The reduced input threshold level takes effect depending on ITHRS, the operating mode and the related enable signals of the EBI pin function as summarized in Table 5-4.<br>0 Input threshold is at standard level on all pins<br>1 Reduced input threshold level enabled on pins in use with the external bus interface |
| 5<br>HDBE | **High Data Byte Enable** — This bit enables the higher half of the 16-bit data bus. If disabled, only the lower 8-bit data bus can be used with the external bus interface. In this case the unused data pins and the data select signals ($\overline{UDS}$ and $\overline{LDS}$) are free to be used for alternative functions.<br>0 DATA[15:8], $\overline{UDS}$, and $\overline{LDS}$ disabled<br>1 DATA[15:8], $\overline{UDS}$, and $\overline{LDS}$ enabled |
| 4–0<br>ASIZ[4:0] | **External Address Bus Size** — These bits allow scalability of the external address bus. The programmed value corresponds to the number of available low-aligned address lines (refer to Table 5-5). All address lines ADDR[22:0] start up as outputs after reset in expanded modes. This needs to be taken into consideration when using alternative functions on relevant pins in applications which utilize a reduced external address bus. |

**Table 5-4. Input Threshold Levels on External Signals**

| ITHRS | External Signal | NS | SS | NX | ES | EX | ST |
|---|---|---|---|---|---|---|---|
| 0 | DATA[15:8]<br>$\overline{TAGHI}$, $\overline{TAGLO}$ | Standard | Standard | Standard | Reduced | Reduced | Standard |
| | DATA[7:0] | | | | | | |
| | $\overline{EWAIT}$ | | | | Standard | Standard | |
| 1 | DATA[15:8]<br>$\overline{TAGHI}$, $\overline{TAGLO}$ | Standard | Standard | Reduced<br>if HDBE = 1 | Reduced | Reduced | Reduced |
| | DATA[7:0] | | | Reduced | | | |
| | $\overline{EWAIT}$ | | | Reduced<br>if EWAIT<br>enabled(1) | Standard | Reduced<br>if EWAIT<br>enabled1 | Standard |

1. EWAIT function is enabled if at least one $\overline{CSx}$ line is configured respectively in MMCCTL0. Refer to S12X_MMC section and Table 5-6.

**Table 5-5. External Address Bus Size**

| ASIZ[4:0] | Available External Address Lines |
|---|---|
| 00000 | None |
| 00001 | $\overline{UDS}$ |
| 00010 | ADDR1, $\overline{UDS}$ |

state operation (stretching) of the external bus access is done in emulation modes when accessing internal memory or emulation memory addresses.

In both modes observation of the internal operation is supported through the external bus (internal visibility).

### 5.5.2.1 Example 2a: Emulation Single-Chip Mode

This mode is used for emulation systems in which the target application is operating in normal single-chip mode.

Figure 5-5 shows the PRU connection with the available external bus signals in an emulator application.

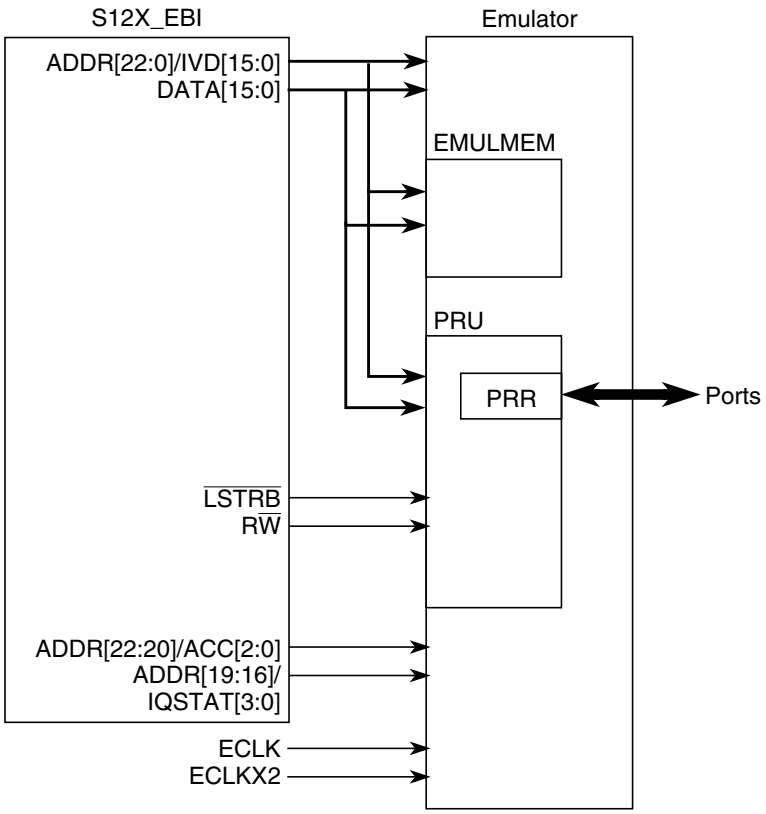


**Figure 5-5. Application in Emulation Single-Chip Mode**

The timing diagram for this operation is shown in:

- Figure ‘Example 2a: Emulation Single-Chip Mode — Read Followed by Write’

The associated timing numbers are given in:

- Table ‘Example 2a: Emulation Single-Chip Mode Timing (EWAIT disabled)’

Timing considerations:

- Signals muxed with address lines ADDRx, i.e., IVDx, IQSTATx and ACCx, have the same timing.
- $\overline{\text{LSTRB}}$ has the same timing as R$\overline{\text{W}}$.

#### 8.4.7.1 XGATE Software Breakpoints

The XGATE software breakpoint instruction BRK can request a CPU12X breakpoint, via the S12XDBG module. In this case, if the XGSBPE bit is set, the S12XDBG module immediately generates a forced breakpoint request to the CPU12X, the state sequencer is returned to state0 and tracing, if active, is terminated. If configured for BEGIN trigger and tracing has not yet been triggered from another source, the trace buffer contains no information. Breakpoint requests from the XGATE module do not depend upon the state of the DBGBRK or ARM bits in DBGC1. They depend solely on the state of the XGSBPE and BDM bits. Thus it is not necessary to ARM the DBG module to use XGATE software breakpoints to generate breakpoints in the CPU12X program flow, but it is necessary to set XGSBPE. Furthermore, if a breakpoint to BDM is required, the BDM bit must also be set. When the XGATE requests an CPU12X breakpoint, the XGATE program flow stops by default, independent of the S12XDBG module.

#### 8.4.7.2 Breakpoints From Internal Comparator Channel Final State Triggers

Breakpoints can be generated when internal comparator channels trigger the state sequencer to the Final State. If configured for tagging, then the breakpoint is generated when the tagged opcode reaches the execution stage of the instruction queue.

If a tracing session is selected by TSOURCE, breakpoints are requested when the tracing session has completed, thus if Begin or Mid aligned triggering is selected, the breakpoint is requested only on completion of the subsequent trace (see Table 8-48). If no tracing session is selected, breakpoints are requested immediately.

If the BRK bit is set on the triggering channel, then the breakpoint is generated immediately independent of tracing trigger alignment.

**Table 8-48. Breakpoint Setup For Both XGATE and CPU12X Breakpoints**

| BRK | TALIGN | DBGBRK[n] | Breakpoint Alignment |
|---|---|---|---|
| 0 | 00 | 0 | Fill Trace Buffer until trigger<br>(no breakpoints — keep running) |
| 0 | 00 | 1 | Fill Trace Buffer until trigger, then breakpoint request occurs |
| 0 | 01 | 0 | Start Trace Buffer at trigger<br>(no breakpoints — keep running) |
| 0 | 01 | 1 | Start Trace Buffer at trigger<br>A breakpoint request occurs when Trace Buffer is full |
| 0 | 10 | 0 | Store a further 32 Trace Buffer line entries after trigger<br>(no breakpoints — keep running) |
| 0 | 10 | 1 | Store a further 32 Trace Buffer line entries after trigger<br>Request breakpoint after the 32 further Trace Buffer entries |
| 1 | 00,01,10 | 1 | Terminate tracing and generate breakpoint immediately on trigger |
| 1 | 00,01,10 | 0 | Terminate tracing immediately on trigger |
| x | 11 | x | Reserved |

| | | 127 | 126 | 125 | 124 | 123 | 122 | 121 | 120 | 119 | 118 | 117 | 116 | 115 | 114 | 113 | 112 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0008 XGIF | R | 0 | 0 | 0 | 0 | 0 | 0 | 0 | XGIF_78 | XGF_77 | XGIF_76 | XGIF_75 | XGIF_74 | XGIF_73 | XGIF_72 | XGIF_71 | XGIF_70 |
| | W | | | | | | | | | | | | | | | | |

| | | 111 | 110 | 109 | 108 | 107 | 106 | 105 | 104 | 103 | 102 | 101 | 100 | 99 | 98 | 97 | 96 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000A XGIF | R | XGIF_6F | XGIF_6E | XGIF_6D | XGIF_6C | XGIF_6B | XGIF_6A | XGIF_69 | XGIF_68 | XGF_67 | XGIF_66 | XGIF_65 | XGIF_64 | XGIF_63 | XGIF_62 | XGIF_61 | XGIF_60 |
| | W | | | | | | | | | | | | | | | | |

| | | 95 | 94 | 93 | 92 | 91 | 90 | 89 | 88 | 87 | 86 | 85 | 84 | 83 | 82 | 81 | 80 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000C XGIF | R | XGIF_5F | XGIF_5E | XGIF_5D | XGIF_5C | XGIF_5B | XGIF_5A | XGIF_59 | XGIF_58 | XGF_57 | XGIF_56 | XGIF_55 | XGIF_54 | XGIF_53 | XGIF_52 | XGIF_51 | XGIF_50 |
| | W | | | | | | | | | | | | | | | | |

| | | 79 | 78 | 77 | 76 | 75 | 74 | 73 | 72 | 71 | 70 | 69 | 68 | 67 | 66 | 65 | 64 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x000E XGIF | R | XGIF_4F | XGIF_4E | XGIF_4D | XGIF_4C | XGIF_4B | XGIF_4A | XGIF_49 | XGIF_48 | XGF_47 | XGIF_46 | XGIF_45 | XGIF_44 | XGIF_43 | XGIF_42 | XGIF_41 | XGIF_40 |
| | W | | | | | | | | | | | | | | | | |

| | | 63 | 62 | 61 | 60 | 59 | 58 | 57 | 56 | 55 | 54 | 53 | 52 | 51 | 50 | 49 | 48 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0010 XGIF | R | XGIF_3F | XGIF_3E | XGIF_3D | XGIF_3C | XGIF_3B | XGIF_3A | XGIF_39 | XGIF_38 | XGF_37 | XGIF_36 | XGIF_35 | XGIF_34 | XGIF_33 | XGIF_32 | XGIF_31 | XGIF_30 |
| | W | | | | | | | | | | | | | | | | |

| | | 47 | 46 | 45 | 44 | 43 | 42 | 41 | 40 | 39 | 38 | 37 | 36 | 35 | 34 | 33 | 32 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0012 XGIF | R | XGIF_2F | XGIF_2E | XGIF_2D | XGIF_2C | XGIF_2B | XGIF_2A | XGIF_29 | XGIF_28 | XGF_27 | XGIF_26 | XGIF_25 | XGIF_24 | XGIF_23 | XGIF_22 | XGIF_21 | XGIF_20 |
| | W | | | | | | | | | | | | | | | | |

| | | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0014 XGIF | R | XGIF_1F | XGIF_1E | XGIF_1D | XGIF_1C | XGIF_1B | XGIF_1A | XGIF_19 | XGIF_18 | XGF_17 | XGIF_16 | XGIF_15 | XGIF_14 | XGIF_13 | XGIF_12 | XGIF_11 | XGIF_10 |
| | W | | | | | | | | | | | | | | | | |

| | | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0x0016 XGIF | R | XGIF_0F | XGIF_0E | XGIF_0D | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | W | | | | | | | | | | | | | | | | |

= Unimplemented or Reserved

**Figure 10-2. XGATE Register Summary (Sheet 2 of 3)**

### 10.3.1.14 XGATE Register 2 (XGR2)

The XGR2 register (Figure 10-16) provides access to the RISC core's register 2.

Module Base +0x00024

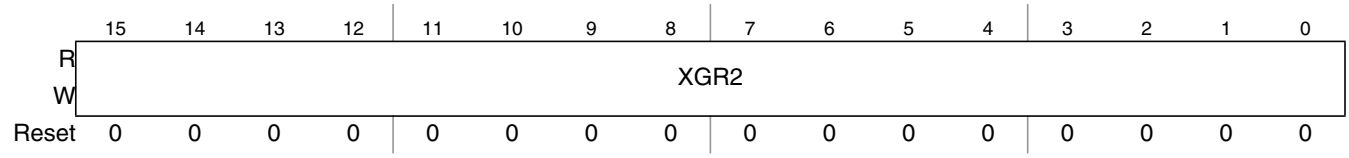


**Figure 10-16. XGATE Register 2 (XGR2)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-16. XGR2 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR2[15:0] | **XGATE Register 2** — The RISC core's register 2 |

### 10.3.1.15 XGATE Register 3 (XGR3)

The XGR3 register (Figure 10-17) provides access to the RISC core's register 3.

Module Base +0x00026

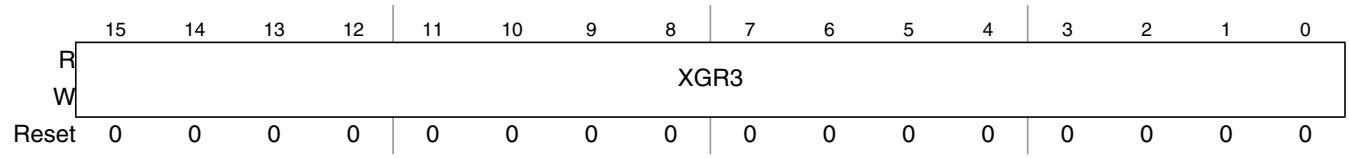


**Figure 10-17. XGATE Register 3 (XGR3)**

Read: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

Write: In debug mode if unsecured and not idle (XGCHID ≠ 0x00)

**Table 10-17. XGR3 Field Descriptions**

| Field | Description |
|---|---|
| 15–0<br>XGR3[15:0] | **XGATE Register 3** — The RISC core's register 3 |

### 13.3.2.12 ATD Conversion Result Registers (ATDDR*n*)

The A/D conversion results are stored in 16 result registers. Results are always in unsigned data representation. Left and right justification is selected using the DJM control bit in ATDCTL3.

If automatic compare of conversions results is enabled (CMPE[*n*]=1 in ATDCMPE), these registers must be written with the compare values in left or right justified format depending on the actual value of the DJM bit. In this case, as the ATDDR*n* register is used to hold the compare value, the result will not be stored there at the end of the conversion but is lost.

Read: Anytime

Write: Anytime

**NOTE**

For conversions not using automatic compare, results are stored in the result registers after each conversion. In this case avoid writing to ATDDRn except for initial values, because an A/D result might be overwritten.

#### 13.3.2.12.1 Left Justified Result Data (DJM=0)

Module Base +
0x0010 = ATDDR0, 0x0012 = ATDDR1, 0x0014 = ATDDR2, 0x0016 = ATDDR3
0x0018 = ATDDR4, 0x001A = ATDDR5, 0x001C = ATDDR6, 0x001E = ATDDR7
0x0020 = ATDDR8, 0x0022 = ATDDR9, 0x0024 = ATDDR10, 0x0026 = ATDDR11
0x0028 = ATDDR12, 0x002A = ATDDR13, 0x002C = ATDDR14, 0x002E = ATDDR15

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | 0 | 0 | 0 | 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-14. Left justified ATD conversion result register (ATDDR*n*)**

#### 13.3.2.12.2 Right Justified Result Data (DJM=1)

Module Base +
0x0010 = ATDDR0, 0x0012 = ATDDR1, 0x0014 = ATDDR2, 0x0016 = ATDDR3
0x0018 = ATDDR4, 0x001A = ATDDR5, 0x001C = ATDDR6, 0x001E = ATDDR7
0x0020 = ATDDR8, 0x0022 = ATDDR9, 0x0024 = ATDDR10, 0x0026 = ATDDR11
0x0028 = ATDDR12, 0x002A = ATDDR13, 0x002C = ATDDR14, 0x002E = ATDDR15

| | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| R | 0 | 0 | 0 | 0 | Bit 11 | Bit 10 | Bit 9 | Bit 8 | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bi1 1 | Bit 0 |
| W | | | | | | | | | | | | | | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Figure 13-15. Right justified ATD conversion result register (ATDDR*n*)**

Table 13-16 shows how depending on the A/D resolution the conversion result is transferred to the ATD result registers. Compare is always done using all 12 bits of both the conversion result and the compare value in ATDDRn.

software simpler because only one address area is applicable for the transmit process, and the required address space is minimized.

The CPU then stores the identifier, the control bits, and the data content into one of the transmit buffers. Finally, the buffer is flagged as ready for transmission by clearing the associated TXE flag.

The MSCAN then schedules the message for transmission and signals the successful transmission of the buffer by setting the associated TXE flag. A transmit interrupt (see Section 16.4.7.2, "Transmit Interrupt") is generated[1] when TXEx is set and can be used to drive the application software to re-load the buffer.

If more than one buffer is scheduled for transmission when the CAN bus becomes available for arbitration, the MSCAN uses the local priority setting of the three buffers to determine the prioritization. For this purpose, every transmit buffer has an 8-bit local priority field (PRIO). The application software programs this field when the message is set up. The local priority reflects the priority of this particular message relative to the set of messages being transmitted from this node. The lowest binary value of the PRIO field is defined to be the highest priority. The internal scheduling process takes place whenever the MSCAN arbitrates for the CAN bus. This is also the case after the occurrence of a transmission error.

When a high priority message is scheduled by the application software, it may become necessary to abort a lower priority message in one of the three transmit buffers. Because messages that are already in transmission cannot be aborted, the user must request the abort by setting the corresponding abort request bit (ABTRQ) (see Section 16.3.2.9, "MSCAN Transmitter Message Abort Request Register (CANTARQ)".) The MSCAN then grants the request, if possible, by:

1. Setting the corresponding abort acknowledge flag (ABTAK) in the CANTAAK register.
2. Setting the associated TXE flag to release the buffer.
3. Generating a transmit interrupt. The transmit interrupt handler software can determine from the setting of the ABTAK flag whether the message was aborted (ABTAK = 1) or sent (ABTAK = 0).

### 16.4.2.3 Receive Structures

The received messages are stored in a five stage input FIFO. The five message buffers are alternately mapped into a single memory area (see Figure 16-39). The background receive buffer (RxBG) is exclusively associated with the MSCAN, but the foreground receive buffer (RxFG) is addressable by the CPU (see Figure 16-39). This scheme simplifies the handler software because only one address area is applicable for the receive process.

All receive buffers have a size of 15 bytes to store the CAN control bits, the identifier (standard or extended), the data contents, and a time stamp, if enabled (see Section 16.3.3, "Programmer's Model of Message Storage").

The receiver full flag (RXF) (see Section 16.3.2.5, "MSCAN Receiver Flag Register (CANRFLG)") signals the status of the foreground receive buffer. When the buffer contains a correctly received message with a matching identifier, this flag is set.

On reception, each message is checked to see whether it passes the filter (see Section 16.4.3, "Identifier Acceptance Filter") and simultaneously is written into the active RxBG. After successful reception of a valid message, the MSCAN shifts the content of RxBG into the receiver FIFO, sets the RXF flag, and

1. The transmit interrupt occurs only if not masked. A polling scheme can be applied on TXEx also.

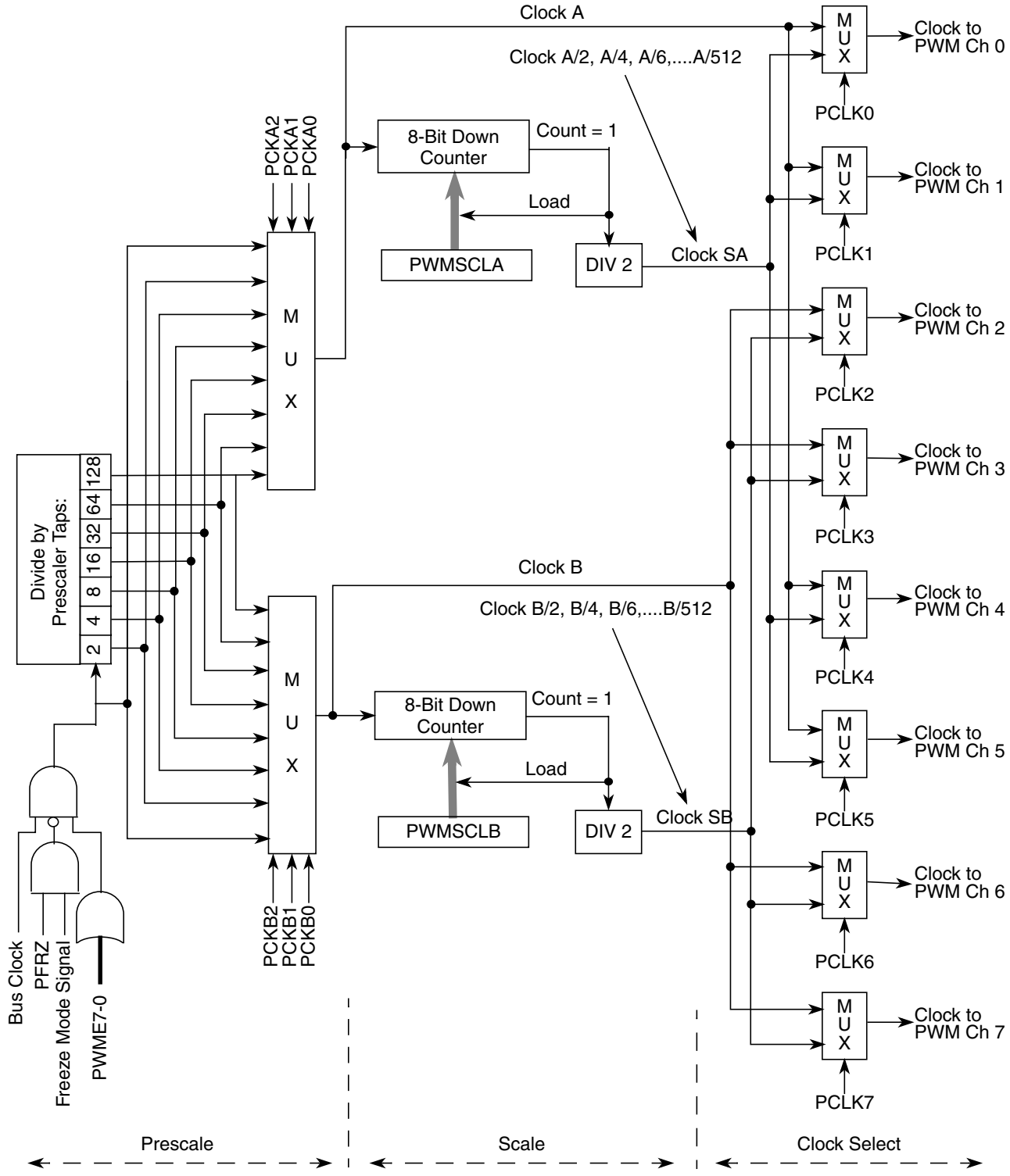


**Figure 19-18. PWM Clock Select Block Diagram**

On the front end of the PWM timer, the clock is enabled to the PWM circuit by the PWMEx bit being high. There is an edge-synchronizing circuit to guarantee that the clock will only be enabled or disabled at an edge. When the channel is disabled (PWMEx = 0), the counter for the channel does not count.

### 19.4.2.2 PWM Polarity

Each channel has a polarity bit to allow starting a waveform cycle with a high or low signal. This is shown on the block diagram as a mux select of either the Q output or the $\overline{Q}$ output of the PWM output flip flop. When one of the bits in the PWMPOL register is set, the associated PWM channel output is high at the beginning of the waveform, then goes low when the duty count is reached. Conversely, if the polarity bit is zero, the output starts low and then goes high when the duty count is reached.

### 19.4.2.3 PWM Period and Duty

Dedicated period and duty registers exist for each channel and are double buffered so that if they change while the channel is enabled, the change will NOT take effect until one of the following occurs:

- The effective period ends
- The counter is written (counter resets to $00)
- The channel is disabled

In this way, the output of the PWM will always be either the old waveform or the new waveform, not some variation in between. If the channel is not enabled, then writes to the period and duty registers will go directly to the latches as well as the buffer.

A change in duty or period can be forced into effect “immediately” by writing the new value to the duty and/or period registers and then writing to the counter. This forces the counter to reset and the new duty and/or period values to be latched. In addition, since the counter is readable, it is possible to know where the count is with respect to the duty value and software can be used to make adjustments

**NOTE**

When forcing a new period or duty into effect immediately, an irregular PWM cycle can occur.

Depending on the polarity bit, the duty registers will contain the count of either the high time or the low time.

### 19.4.2.4 PWM Timer Counters

Each channel has a dedicated 8-bit up/down counter which runs at the rate of the selected clock source (see Section 19.4.1, “PWM Clock Select” for the available clock sources and rates). The counter compares to two registers, a duty register and a period register as shown in Figure 19-19. When the PWM counter matches the duty register, the output flip-flop changes state, causing the PWM waveform to also change state. A match between the PWM counter and the period register behaves differently depending on what output mode is selected as shown in Figure 19-19 and described in Section 19.4.2.5, “Left Aligned Outputs” and Section 19.4.2.6, “Center Aligned Outputs”.

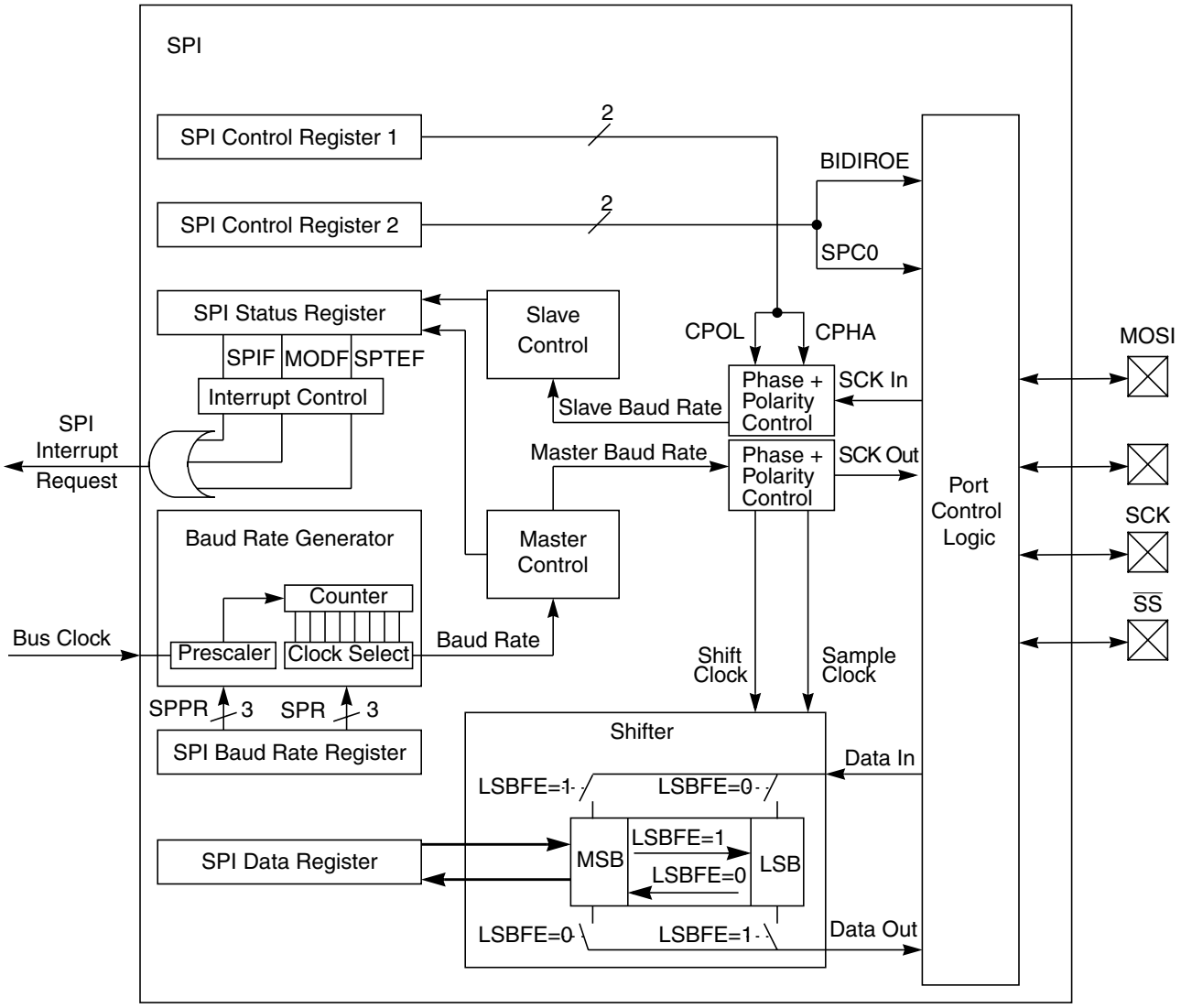


**Figure 21-1. SPI Block Diagram**

## 21.2 External Signal Description

This section lists the name and description of all ports including inputs and outputs that do, or may, connect off chip. The SPI module has a total of four external pins.

### 21.2.1 MOSI — Master Out/Slave In Pin

This pin is used to transmit data out of the SPI module when it is configured as a master and receive data when it is configured as slave.

### 21.2.2 MISO — Master In/Slave Out Pin

This pin is used to transmit data out of the SPI module when it is configured as a slave and receive data when it is configured as master.

### 22.2.7 IOC1 — Input Capture and Output Compare Channel 1 Pin

This pin serves as input capture or output compare for channel 1.

### 22.2.8 IOC0 — Input Capture and Output Compare Channel 0 Pin

This pin serves as input capture or output compare for channel 0.

**NOTE**

For the description of interrupts see Section 22.6, “Interrupts”.

## 22.3 Memory Map and Register Definition

This section provides a detailed description of all memory and registers.

### 22.3.1 Module Memory Map

The memory map for the TIM16B8CV2 module is given below in Figure 22-5. The address listed for each register is the address offset. The total address for each register is the sum of the base address for the TIM16B8CV2 module and the address offset for each register.

### 22.3.2 Register Descriptions

This section consists of register descriptions in address order. Each description includes a standard register diagram with an associated figure number. Details of register bit and field function follow the register diagrams, in bit order.

| Register Name | | Bit 7 | 6 | 5 | 4 | 3 | 2 | 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|---|
| 0x0000 TIOS | R<br>W | IOS7 | IOS6 | IOS5 | IOS4 | IOS3 | IOS2 | IOS1 | IOS0 |
| 0x0001 CFORC | R<br>W | 0<br>FOC7 | 0<br>FOC6 | 0<br>FOC5 | 0<br>FOC4 | 0<br>FOC3 | 0<br>FOC2 | 0<br>FOC1 | 0<br>FOC0 |
| 0x0002 OC7M | R<br>W | OC7M7 | OC7M6 | OC7M5 | OC7M4 | OC7M3 | OC7M2 | OC7M1 | OC7M0 |
| 0x0003 OC7D | R<br>W | OC7D7 | OC7D6 | OC7D5 | OC7D4 | OC7D3 | OC7D2 | OC7D1 | OC7D0 |
| 0x0004 TCNTH | R<br>W | TCNT15 | TCNT14 | TCNT13 | TCNT12 | TCNT11 | TCNT10 | TCNT9 | TCNT8 |
| 0x0005 TCNTL | R<br>W | TCNT7 | TCNT6 | TCNT5 | TCNT4 | TCNT3 | TCNT2 | TCNT1 | TCNT0 |

= Unimplemented or Reserved

**Figure 22-5. TIM16B8CV2 Register Summary (Sheet 1 of 3)**

**Table 23-13. Interrupt Vectors**

| Interrupt Source | Local Enable |
|---|---|
| Low-voltage interrupt (LVI) | LVIE = 1; available only in Full Performance Mode |
| High Temperature Interrupt (HTI) | HTIE=1; available only in Full Performance Mode |
| Autonomous periodical interrupt (API) | APIE = 1 |

### 23.4.11.1 Low-Voltage Interrupt (LVI)

In FPM, VREG_3V3 monitors the input voltage $V_{DDA}$. Whenever $V_{DDA}$ drops below level $V_{LVIA,}$ the status bit LVDS is set to 1. On the other hand, LVDS is reset to 0 when $V_{DDA}$ rises above level $V_{LVID}$. An interrupt, indicated by flag LVIF = 1, is triggered by any change of the status bit LVDS if interrupt enable bit LVIE = 1.

**NOTE**

On entering the Reduced Power Mode, the LVIF is not cleared by the VREG_3V3.

### 23.4.11.2 HTI - High Temperature Interrupt

In FPM VREG monitors the die temperature $T_{DIE}$. Whenever $T_{DIE}$ exceeds level $T_{HTIA}$ the status bit HTDS is set to 1. Vice versa, HTDS is reset to 0 when $T_{DIE}$ get below level $T_{HTID}$. An interrupt, indicated by flag HTIF=1, is triggered by any change of the status bit HTDS if interrupt enable bit HTIE=1.

**NOTE**

On entering the Reduced Power Mode the HTIF is not cleared by the VREG.

### 23.4.11.3 Autonomous Periodical Interrupt (API)

As soon as the configured timeout period of the API has elapsed, the APIF bit is set. An interrupt, indicated by flag APIF = 1, is triggered if interrupt enable bit APIE = 1.

**Table 24-3. Flash Configuration Field[1]**

| Global Address | Size (Bytes) | Description |
|---|---|---|
| 0x7F_FF0F[2] | 1 | Flash Security byte<br>Refer to Section 24.3.2.2, “Flash Security Register (FSEC)” |

1. Older versions may have swapped protection byte addresses
2. 0x7FF08 - 0x7F_FF0F form a Flash phrase and must be programmed in a single command write sequence. Each byte in the 0x7F_FF08 - 0x7F_FF0B reserved field should be programmed to 0xFF.

**Table 25-47. Erase All Blocks Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x08 | Not required |

Upon clearing CCIF to launch the Erase All Blocks command, the Memory Controller will erase the entire Flash memory space and verify that it is erased. If the Memory Controller verifies that the entire Flash memory space was properly erased, security will be released. During the execution of this command (CCIF=0) the user must not write to any Flash module register. The CCIF flag will set after the Erase All Blocks operation has completed.

**Table 25-48. Erase All Blocks Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 25-30) |
| | FPVIOL | Set if any area of the P-Flash memory is protected |
| | MGSTAT1 | Set if any errors have been encountered during the verify operation |
| | MGSTAT0 | Set if any non-correctable errors have been encountered during the verify operation |
| FERSTAT | EPVIOLIF | Set if any area of the buffer RAM EEE partition is protected |

### 25.4.2.9 Erase P-Flash Block Command

The Erase P-Flash Block operation will erase all addresses in a P-Flash block.

**Table 25-49. Erase P-Flash Block Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x09 | Global address [22:16] to identify P-Flash block |
| 001 | Global address [15:0] in P-Flash block to be erased | |

Upon clearing CCIF to launch the Erase P-Flash Block command, the Memory Controller will erase the selected P-Flash block and verify that it is erased. The CCIF flag will set after the Erase P-Flash Block operation has completed.

**Table 28-76. EEPROM Emulation Query Command Error Handling**

| Register | Error Bit | Error Condition |
|---|---|---|
| FSTAT | ACCERR | Set if CCOBIX[2:0] != 000 at command launch |
| | | Set if a Load Data Field command sequence is currently active |
| | | Set if command not available in current mode (see Table 28-30) |
| | FPVIOL | None |
| | MGSTAT1 | None |
| | MGSTAT0 | None |
| FERSTAT | EPVIOLIF | None |

### 28.4.2.22 Partition D-Flash Command

The Partition D-Flash command allows the user to allocate sectors within the D-Flash block for applications and a partition within the buffer RAM for EEPROM access. The D-Flash block consists of 128 sectors with 256 bytes per sector. The Erase All Blocks command must be run prior to launching the Partition D-Flash command.

**Table 28-77. Partition D-Flash Command FCCOB Requirements**

| CCOBIX[2:0] | FCCOB Parameters | |
|---|---|---|
| 000 | 0x20 | Not required |
| 001 | Number of 256 byte sectors for the D-Flash user partition (DFPART) | |
| 010 | Number of 256 byte sectors for buffer RAM EEE partition (ERPART) | |

Upon clearing CCIF to launch the Partition D-Flash command, the following actions are taken to define a partition within the D-Flash block for direct access (DFPART) and a partition within the buffer RAM for EEE use (ERPART):

- Validate the DFPART and ERPART values provided:
  - DFPART <= 128 (maximum number of 256 byte sectors in D-Flash block)
  - ERPART <= 16 (maximum number of 256 byte sectors in buffer RAM)
  - If ERPART > 0, 128 - DFPART >= 12 (minimum number of 256 byte sectors in the D-Flash block required to support EEE)
  - If ERPART > 0, ((128-DFPART)/ERPART) >= 8 (minimum ratio of D-Flash EEE space to buffer RAM EEE space to support EEE)
- Erase verify the D-Flash block and the EEE nonvolatile information register
- Program DFPART to the EEE nonvolatile information register at global address 0x12_0000 (see Table 28-7)

2. FDIV shown generates an FCLK frequency of 1.05 MHz

### 29.3.2.2 Flash Security Register (FSEC)

The FSEC register holds all bits associated with the security of the MCU and Flash module.

Offset Module Base + 0x0001

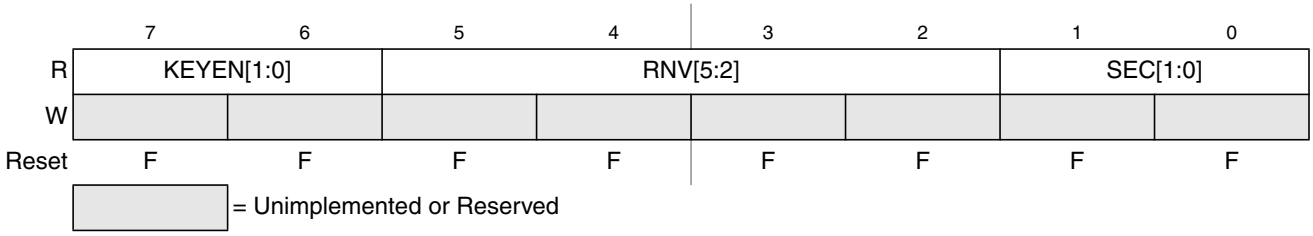


**Figure 29-6. Flash Security Register (FSEC)**

All bits in the FSEC register are readable but not writable.

During the reset sequence, the FSEC register is loaded with the contents of the Flash security byte in the Flash configuration field at global address 0x7F_FF0F located in P-Flash memory (see Table 29-3) as indicated by reset condition F in Figure 29-6. If a double bit fault is detected while reading the P-Flash phrase containing the Flash security byte during the reset sequence, all bits in the FSEC register will be set to leave the Flash module in a secured state with backdoor key access disabled.

**Table 29-10. FSEC Field Descriptions**

| Field | Description |
|---|---|
| 7–6<br>KEYEN[1:0] | **Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of backdoor key access to the Flash module as shown in Table 29-11. |
| 5–2<br>RNV[5:2} | **Reserved Nonvolatile Bits** — The RNV bits should remain in the erased state for future enhancements. |
| 1–0<br>SEC[1:0] | **Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 29-12. If the Flash module is unsecured using backdoor key access, the SEC bits are forced to 10. |

**Table 29-11. Flash KEYEN States**

| KEYEN[1:0] | Status of Backdoor Key Access |
|---|---|
| 00 | DISABLED |
| 01 | DISABLED(1) |
| 10 | ENABLED |
| 11 | DISABLED |

1. Preferred KEYEN state to disable backdoor key access.

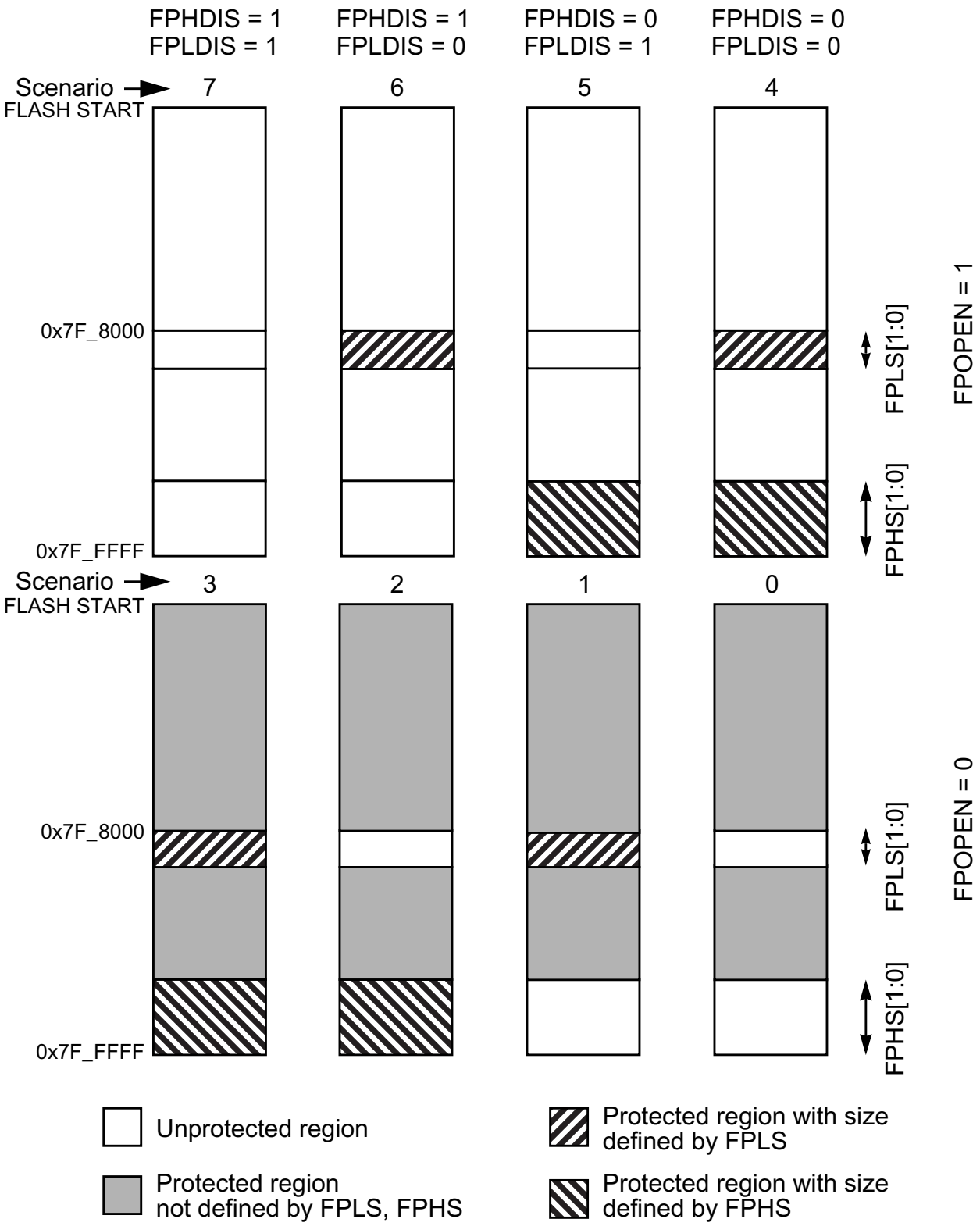


**Figure 29-14. P-Flash Protection Scenarios**