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	e200z4d
Core Size	32-Bit Single-Core
Speed	80MHz
Connectivity	CANbus, LINbus, SPI, UART/USART
Peripherals	DMA, POR, PWM, WDT
Number of I/O	-
Program Memory Size	1MB (1M x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 32x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mpxs2010vlq80

NOTES:
1 DMA_MUX_1 is disabled in DP mode.
2 Test flash memory is mapped to this address if the SCTR[TFE] bit in the SSCM is set (see Section 48.3.1.9, SSCM Control Register (SCTR)).
3 This range cannot be accessed by eDMA_0.

Table 2-3. SRAM map for PXS20 in LSM

Start address	End address	Size (KB)	Region[1]
0x4000_0000	0x4001_FFFF	128	SRAM
0x4002_0000	0x5FFF_FFFF	524160	Reserved

NOTES:
1 Access to reserved memory space can cause unexpected behavior. The MPU must be configured to force a deterministic action for accesses to reserved memory space.

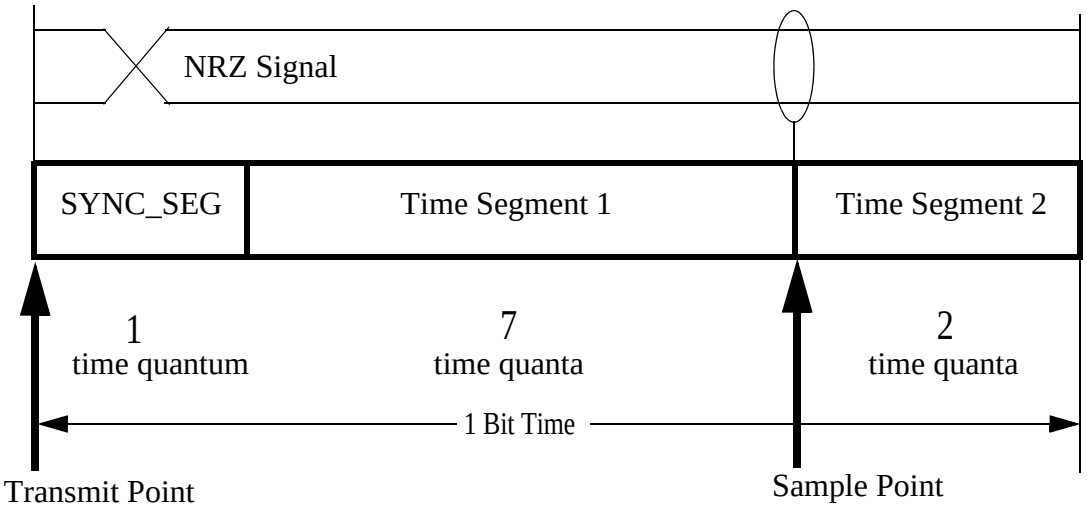
Table 2-4. SRAM map for PXS20 in DPM

Start address	End address	Size (KB)	Region[1]
0x4000_0000	0x4000_FFFF	64	SRAM
0x4001_0000	0x4FFF_FFFF	262080	Reserved
0x5000_0000	0x5000_FFFF	64	SRAM[2]
0x5001_0000	0x5FFF_FFFF	262080	Reserved

NOTES:
1 Access to reserved memory space can cause unexpected behavior. The MPU must be configured to force a deterministic action for accesses to reserved memory space.
2 This range cannot be accessed by eDMA_0.

It uses the standard 11 bit identifier format detailed in FlexCAN 2.0A specification.

FlexCAN controller bit timing is programmed with 10 time quanta, and the sample point is 2 time quanta before the end, as shown in Figure 8-5.



1 time Quantum = 4 system clock periods

Figure 8-5. FlexCAN bit timing

8.4.4.2 Protocol

Table 8-6 summarizes the protocol and BAM action during this boot mode. All data is transmitted byte wise.

Table 8-6. FlexCAN boot mode download protocol (autobaud disabled).

Protocol step	Host sent message	BAM response message	Action
1	FlexCAN ID 0x011+ 64-bit password	FlexCAN ID 0x001+ 64-bit password	Password checked for validity and compared against stored password.
2	FlexCAN ID 0x012+ 32-bit store address+ VLE bit+ 31-bit number of bytes	FlexCAN ID 0x002+ 32-bit store address+ VLE bit+ 31-bit number of bytes	Load address is stored for future use. Size of download are stored for future use. Verify VLE bit.
3	FlexCAN ID 0x013+ 8 to 64 bits of raw binary data	FlexCAN ID 0x003+ 8 to 64 bits of raw binary data	4 x 8 bits of data are packed into 32-bit words. These words are saved into SRAM starting from the “Load address”. “Load address” increments until the number of data received and stored matches the size as specified in the previous step.
4	none	none	Branch to downloaded code

- At end of the normal conversion chain, Step1 of RC algorithm is executed.
- This process continues for all the Steps of all three algorithms.
- State Machine returns to IDLE state when MCR[NSTART] is cleared.

Instead of starting normal conversion by software (by setting MCR[NSTART]), if it is started by external trigger, the test channel behaviour will remain same.

In case of Injected conversions, test channel conversion is not performed. It is performed only during Normal conversions.

If during a test channel conversion, injection conversion arrives, then the test channel is aborted (just as a normal functional channel) and injected conversions are done. After injected conversions are completed, the test channel resumes from the Step at which it was aborted. In this case, the MSR.SELF_TEST_S remains high during the injected conversion.

For self testing, Mode bit should be programmed (atleast one cycle) BEFORE setting MCR.NSTART bit and should not be changed thereafter until conversion is ongoing.

9.4.11.3 CTU mode

The CTU mode is enabled by setting MCR[CTUEN]. With this bit set, the CTU operates in control mode.

If CTUEN is set, the test channel conversion can be started only by CTU interface and software cannot start it. The EN bit in STCR2 register does not have any effect in CTU mode. The CTUEN bit should not be changed while normal conversion is ongoing.

The interface between CTU and ADC (for Self Test) is shown in Figure 9-50.

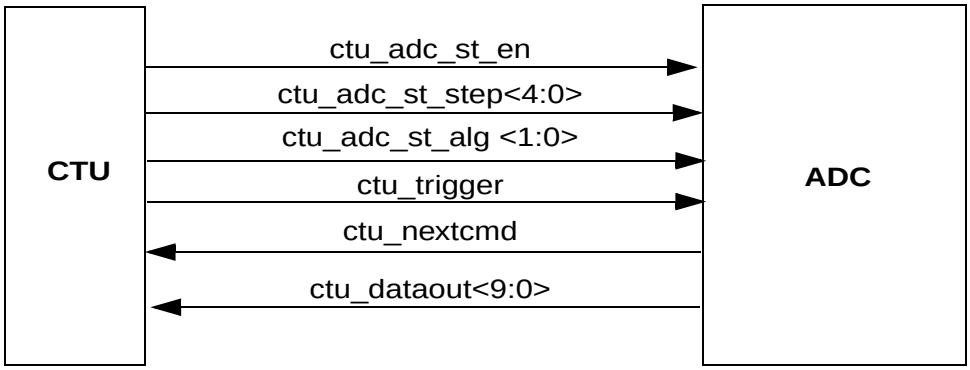


Figure 9-50. Interface between ADC and CTU to manage self test

For self testing conversions in CTU mode, CTU asserts ctu_adc_st_en signal along with ctu_trigger. The algorithm and the step number to be executed is put on ctu_adc_st_alg and ctu_adc_st_step respectively. The other signals ctu_nextcmd, ctu_trigger and ctu_dataout have same meaning as for normal CTU functional conversion.

11.3.1.1 Output Clock Enable Register (CGM_OC_EN)

Address 0xC3FE_0370 Access: User read, Supervisor read/write, Test read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	EN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 11-2. Output Clock Enable Register (CGM_OC_EN)

This register is used to enable and disable the output clock.

Table 11-3. Output Clock Enable Register (CGM_OC_EN) Field Descriptions

Field	Description
EN	**Output Clock Enable control** 0 Output Clock is disabled 1 Output Clock is enabled

11.3.1.2 Output Clock Division Select Register (CGM_OCDS_SC)

Address 0xC3FE_0374 Access: User read-only, Supervisor read/write, Test read/write

	0	1	2	3	4	5	6	7
R	0	0	SELDIV		SELCTL			
W								
Reset	0	0	0	0	0	0	0	0

Figure 11-3. Output Clock Division Select Register (CGM_OCDS_SC)

This register is used to select the current output clock source and by which factor it is divided before being delivered at the output clock.

17.3.1.2 Processor ID Register (PIR)

The processor ID for each of the two CPU cores is contained in its own Processor ID Register (PIR). The contents of the PIR are a reflection of hardware input signals to the core following reset. This register may be written by software to modify the default reset value. This register value can be used by the application software to determine the core actually running the software.

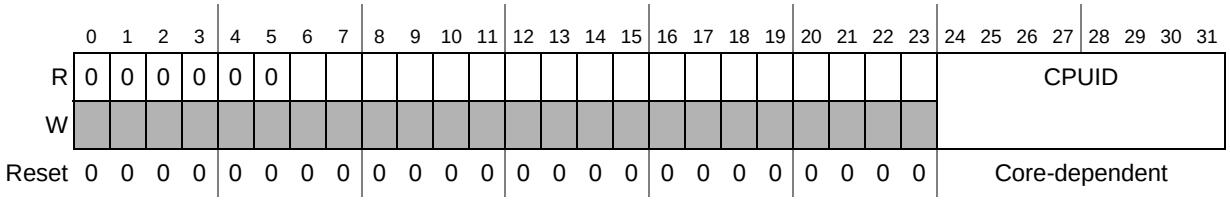


Figure 17-5. Processor ID Register (PIR)

Table 17-2. PIR field descriptions

Field	Description
CPUID	Processor ID

17.3.1.3 System Version Register (SVR)

The SVR contains system version information for this device.

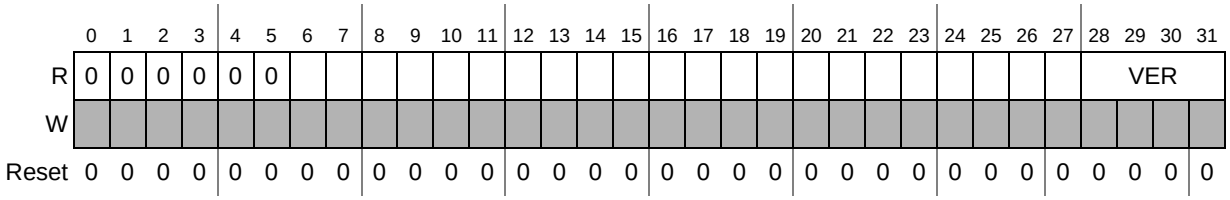


Figure 17-6. System Version Register (SVR)

Table 17-3. SVR field descriptions

Field	Description
VER	Device version

17.3.2 Instruction set

The e200z4d supports the Power ISA instruction set for 32-bit embedded implementations. This is composed primarily of the user-level instructions defined by the user instruction set architecture (UISA). The e200z4d does not include the Power ISA floating-point, load string, or store string instructions.

The e200z4d core implements the following architectural extensions:

- The VLE category
- The integer select category (ISEL)
- Enhanced debug and the debug notify halt instruction categories
- The machine check category

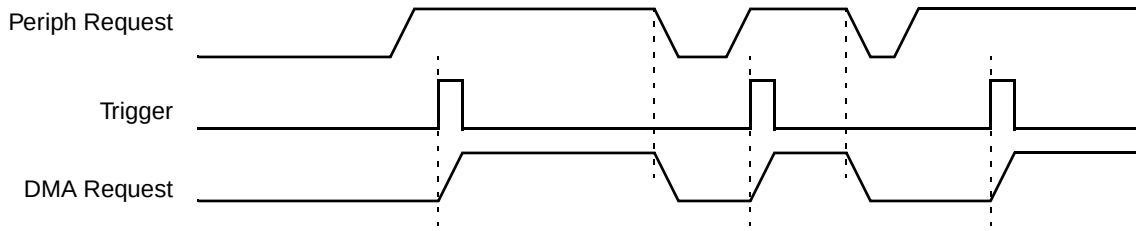


Figure 18-4. DMA_MUX Channel Triggering: Normal Operation

Once the DMA request has been serviced, the peripheral will negate its request, effectively resetting the gating mechanism until the peripheral re-asserts its request AND the next trigger event is seen. This means that if a trigger is seen, but the peripheral is not requesting a transfer, that trigger will be ignored. This situation is illustrated in Figure 18-5.

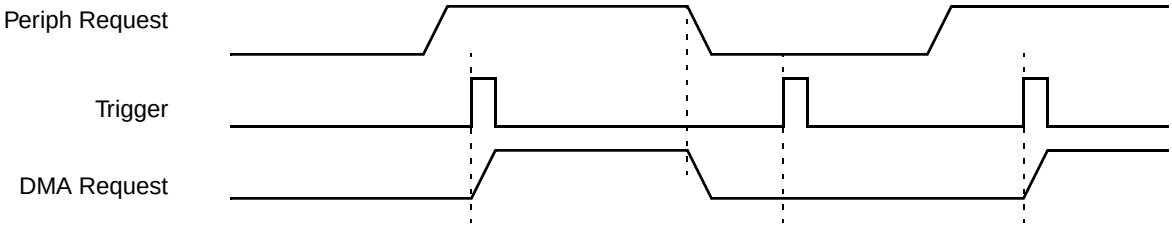


Figure 18-5. DMA_MUX Channel Triggering: Ignored Trigger

This triggering capability may be used with any peripheral that supports DMA transfers, and is most useful for two types of situations:

- Periodically polling external devices on a particular bus. As an example, the transmit side of an SPI is assigned to a DMA channel with a trigger, as described above. Once setup, the SPI will request DMA transfers (presumably from memory) as long as its transmit buffer is empty. By using a trigger on this channel, the SPI transfers can be automatically performed every 5µs (as an example). On the receive side of the SPI, the SPI and DMA can be configured to transfer receive data into memory, effectively implementing a method to periodically read data from external devices and transfer the results into memory without processor intervention.
- Using the GPIO Ports to drive or sample waveforms. By configuring the DMA to transfer data to one or more GPIO ports, it is possible to create complex waveforms using tabular data stored in on-chip memory. Conversely, using the DMA to periodically transfer data from one or more GPIO ports, it is possible to sample complex waveforms and store the results in tabular form in on-chip memory.

A more detailed description of the capability of each trigger (i.e.-resolution, range of values, etc.) may be found in the Periodic Interrupt Timer (PIT) Block Guide.

18.6.2 DMA channels with no triggering capability

The other channels of the DMA Mux provide the normal routing functionality as described in Section 18.1.3, Modes of operation.

To address this, the compare registers are updated in hardware in the same way the counter register is re-initialized to the value stored in the LOAD register. The compare load feature allows the user to calculate new compare values and store them in to the comparator load registers. When a compare event occurs, the new compare values in the comparator load registers are written to the compare registers eliminating the use of software to do this.

The compare load feature is intended to be used in variable frequency PWM mode. The COMP1 register determines the pulse width for the logic low part of OFLAG and COMP2 determines the pulse width for the logic high part of OFLAG. The period of the waveform is determined by the COMP1 and COMP2 values and the frequency of the primary clock source. See Figure 20-28.

20.5.2.15 MODULO COUNTING Mode

To create a modulo counter using COMP1 and COMP2 as the counter boundaries (instead of $0000 and $FFFF), set the registers in the following manner. Set CNTMODE to either 100 (quadrature count mode) or 101 (count with direction mode). Use count through roll-over (LENGTH = 0) and continuous count (ONCE = 0). Set COMP1 and CMPLD1 to the upper boundary value. Set COMP2 and CMPLD2 to the lower boundary value. Set CMPMODE = 10 (COMP1 is used when counting up and COMP2 is used when counting down). Set CLC2 = 110 (load CNTR with value of CMPLD2 on COMP1 compare) and CLC1 = 111 (load CNTR with value of CMPLD1 on COMP2 compare).

20.5.3 Other Features

20.5.3.1 Redundant OFLAG Checking

This mode allows the user to bundle two timer functions generating any pattern to compare their resulting OFLAG behaviors (output signal).

The redundant mode is used to support online checks for functional safety reasons. Whenever a mismatch between the two adjacent channels occurs, it is reported via an interrupt to the core and the two outputs are put into their inactive states. An error is flagged via the RCF flag.

This feature can be tested by forcing a transition on one of the OFLAGs using the VAL and FORCE bits of the channel.

20.5.3.2 Loopback Checking

This mode is always available in that one channel can generate an OFLAG while another channel uses the first channels OFLAG as its input to be measured and verified to be as expected.

20.5.3.3 Input Capture Mode

Input capture is used to measure pulse width (by capturing the counter value on two successive input edges) or waveform period (by capturing the counter value on two consecutive rising edges or two consecutive falling edges). The Capture Registers store a copy of the counter’s value when an input edge (positive, negative, or both) is detected. The type of edge to be captured by each circuit is determined by the CPT1MODE and CPT2MODE bits whose functionality is listed in Table 20-14. Also, controlling the

Table 22-5. FCCU_CFG field descriptions (continued)

Field	Description
FCCU_CFG.FOM	Fault Output Mode selection 000 Dual-Rail (default state; FCCU_F[1:0]= outputs). 001 Time Switching (FCCU_F[1:0]= outputs). 010 Bi-Stable (FCCU_F[1:0]= outputs). 011 Reserved. 100 Reserved. 101 Test0 (FCCU_F[0] = input, FCCU_F[1]= output) 110 Test1 (FCCU_F[0] = output, FCCU_F[1]= output) 111 Test2 (FCCU_F[0] = output, FCCU_F[1]= input) **Note:** In Test*n* mode, a simple double-stage resynchronization stage is used to resynchronize the FCCU_F input/outputs on the system/IRCOSC clock.
FCCU_CFG.FOP	Fault Output Prescaler FOP defines the prescaler setting used to generate the FCCU_F protocol frequency. 00 0000 Input clock frequency (IRCOSC clock) is divided by 2 00 0001 Input clock frequency (IRCOSC clock) is divided by 4 00 0010 Input clock frequency (IRCOSC clock) is divided by 6 00 0011 Input clock frequency (IRCOSC clock) is divided by 8 00 0100 Input clock frequency (IRCOSC clock) is divided by 10 00 0101 Input clock frequency (IRCOSC clock) is divided by 12 00 0110 Input clock frequency (IRCOSC clock) is divided by 14 The following equation gives the FCCU_F frequency: $$F_{\text{FCCU\_F}} = F_{IRCOSC} / (1024 \times (FOP + 1) \times 2)$$

22.6.4 FCCU CF Configuration Register (FCCU_CF_CFG0..3)

The FCCU_CF_CFGx register is accessible in write mode only in the CONFIG state. It contains the configuration of each critical fault in terms of fault recovery management.

The configuration depends on the type of signaling following a fault event. Hardware recoverable faults should be configured only if a previous latching stage captures and hold the physical fault otherwise the fault can be lost. All the other faults should be configured as SW fault.

Offset: 0x0E4 Access: User read/write

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	CFG_TO_IEN
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-23. FCCU IRQ Enable Register (FCCU_IRQ_EN)

Table 22-22. FCCU_IRQ_EN field descriptions

Field	Description
CFG_TO_IEN	*Configuration Time-out Interrupt Enable* 0: Configuration time-out interrupt disabled 1: Configuration time-out interrupt enabled This bit can be read and written by the software.

22.6.21 FCCU XTMR Register (FCCU_XTMR)

The FCCU_XTMR register contains the read values of the Alarm, Watchdog or Safe Mode Request Timer. These timers are clocked on the IRCOSC clock.

The SW application executes the timer read operation by the following sequence:

- to set the OP17 or OP18 or OP19 operation into the FCCU_CTRL.OPR field
- to wait for the completion of the operation (FCCU_CTRL.OPS field)
- to read the FCCU_XTMR register

Table 22-23. Timer state/value

TIMER	CONFIG state	NORMAL state	ALARM state	FAULT state
ALARM	00000000h	Initial value	Running	Idle/End of count
SMRT	00000001h	Initial value	—	Running/End of count
CFG	Running	0001FFFFh	0001FFFFh	0001FFFFh

Offset: 0x0E8 Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	XTMR[31:16]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	XTMR[15:0]															
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-24. FCCU XTMR Register (FCCU_XTMR)

Table 22-24. FCCU_XTMR field descriptions

Field	Description
XTMR	*Alarm/Watchdog/Safe request timer* The current timer value is measured in IRCOSC clock cycles. These bits can be read by the software.

22.6.22 FCCU MCS Register (FCCU_MCS)

The FCCU_MCS register contains a queue of the last 4 chip modes. MCS0 is the latest one, while MCS3 is the oldest one. In addition a qualifier indicates if the FCCU is in the FAULT state when the chip mode has been captured. The chip mode is synchronous to the system clock and provided by a different module while the FCCU state is synchronous to the IRCOSC clock, therefore some uncertainty must be considered regarding the FAULT state indication.

Offset: 0x0EC Access: User read

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	VL3	FS3	0	0	MCS3				VL2	FS2	0	0	MCS2			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	VL1	FS1	0	0	MCS1				VL0	FS0	0	0	MCS0			
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 22-25. FCCU MCS Register (FCCU_MCS)

Table 22-25. FCCU_MCS field descriptions

Field	Description
VLx	*Valid* It indicates that the correspondent MCSx and FSx fields are valid. 0: MCSx, FSx fields are not significative 1: MCSx, FSx fields are significative These bits can be read by the software.

Table 23-7. SLL Field Descriptions (continued)

Field	Description
12-13	Reserved, reset to 0
14-15 SMLOCK[1:0]	Secondary Mid Address Block Lock. This bit is an alternative method that may be used to lock the Mid Address Space blocks from programs and erases. SMLOCK has the same description as MLOCK. SMLOCK is not writable unless SLE is high.
16-21	Reserved, reset to 0
22-31 SLLOCK[9:0]	Secondary Low Address Block Lock. This bit is an alternative method that may be used to lock the Low Address Space blocks from programs and erases. SLLOCK has the same description as LLOCK. SLLOCK is not writable unless SLE is high.

23.1.6.5 Low/Mid Address Space Block Select Register (LMS)

The Low/Mid Address Space Block Select Register (LMS) provides a means to select blocks to be operated on during erase.

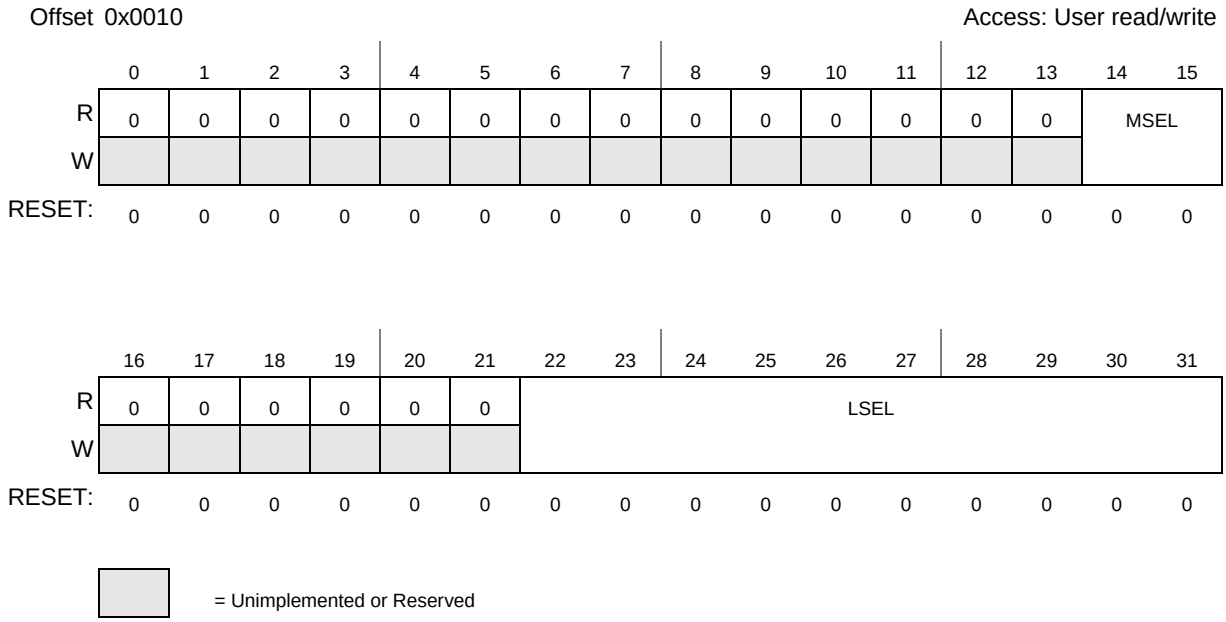


Figure 23-7. LMS Register

LMS register functions are shown in Table 23-8.

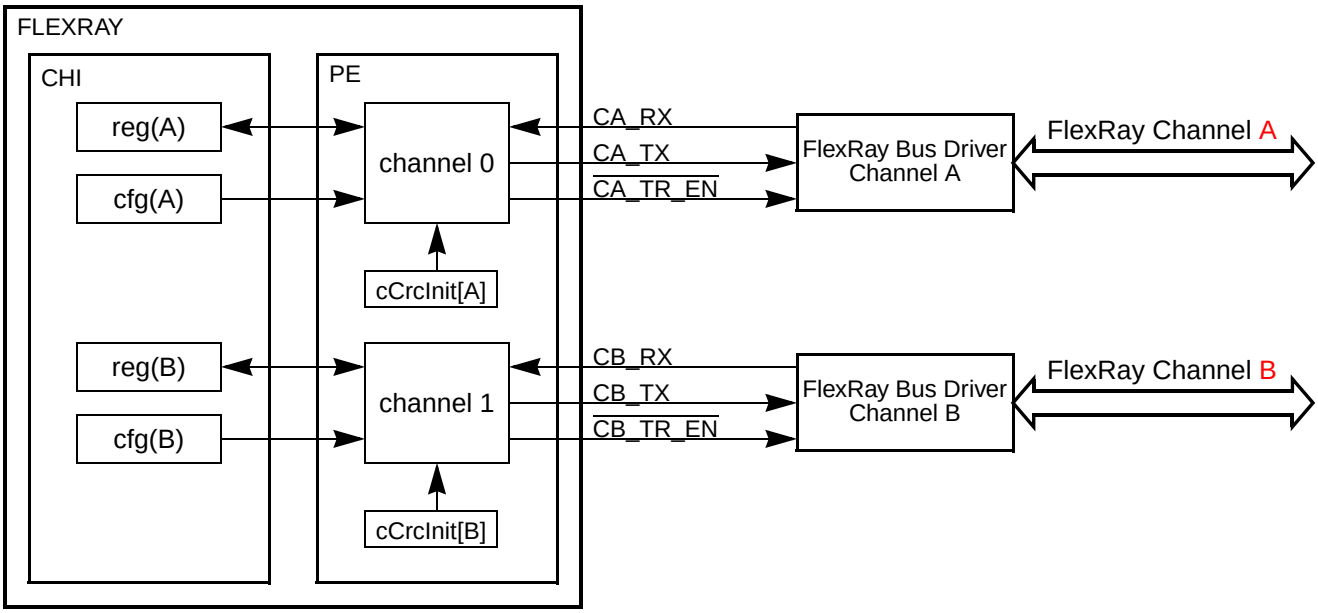


Figure 26-149. Dual Channel Device Mode

26.6.10.2 Single Channel Device Mode

The single channel device mode supports devices that have only one FlexRay port available. This FlexRay port consists of the signals CA_RX, CA_TX, and $\overline{\text{CA\_TR\_EN}}$ and can be connected to either the physical bus channel A (shown in Figure 26-150) or the physical bus channel B (shown in Figure 26-151).

If the device is configured as a single channel device by setting FR_MCR.SCD to 1, only the internal channel A and the FlexRay Port A is used. Depending on the setting of FR_MCR.CHA and FR_MCR.CHB, the internal channel A behaves either as a FlexRay Channel A or FlexRay Channel B. The bit FR_MCR.CHA must be set, if the FlexRay Port A is connected to a FlexRay Channel A. The bit FR_MCR.CHB must be set if the FlexRay Port A is connected to a FlexRay Channel B. The two FlexRay channels differ only in the initial value for the frame CRC *cCrcInit*. For a single channel device, the application can access and configure only the registers related to internal channel A.

26.6.24.3.4 PE DRAM Error Response after Application Read in *POC:default config* state

If the module detects an non-corrected memory error during an application triggered read from any PE DRAM address and the protocol is in the *POC:default config* state, this is considered as an fatal protocol error and the module enters the protocol freeze state. This behavior allows for checking the freeze functionality in case of the detection of non-corrected errors.

26.6.24.3.5 PE DRAM Error Response after Application Read out of *POC:default config*

If the module detects an non-corrected memory error during an application triggered read from any PE DRAM address, and the protocol is not in the *POC:default config* state, this error is not considered as an fatal error and the protocol state is not changed. This prevents any interference of the running protocol by PE DRAM error injection reads.

26.6.25 Memory Error Injection

The error injection functionality is used by the application to inject data errors into the memories to trigger and check the memory error detection functionality.

The error injection is enabled only if the ECC functionality enable bit ECCE in the Module Configuration Register (FR_MCR) and the error injection enable control bit EIE in the ECC Error Report and Injection Control Register (FR_EERICR) are set.

The error injection mode is configured by the EIM configuration bit in the ECC Error Report and Injection Control Register (FR_EERICR).When the error injection is enabled, each write access to the configured memory location will be distorted.

The injector has the same behavior for FlexRay module memory writes and application memory writes.

26.6.25.1 CHI LRAM Error Injection

The following sequence describes an error injection sequence for the CHI LRAM. This sequence includes the setup of the error injector followed by an application triggered write access to provoke an distortion of the memory content. When the FlexRay module is in *POC:default config*, there are no limitations and impacts of error injection for the application. For error injection out of *POC:default config* see Section 26.7.2, CHI LRAM Error Injection out of POC:default config.

Injector Setup:

1. FR_MCR[ECCE]:= 1;
   - enable ecc functionality
2. FR_EERICE[EIE]:=I_MODE;
   - configure error injection mode
3. FR_EEIAR[MID]:= 1;
   - select CHI LRAM for error injection
4. FR_EEIAR[BANK]:= I_BANK;
   - define the bank for error injection; I_BANK = {0,1,2,3,4,5}

Address: Base + 0x70

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

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-5. PMUCTRL mask fault register (PMUCTRL_MASKF)

Table 39-6. PMUCTRL_MASKF field descriptions

Field	Description
MF_BB	Mask Fault Bypass Ballast. This field defines the mask for the nBYPASS_BALLAST_LV[3:0] bus. 0 MF_BB[n], means BYPASS_BALLAST_LV[n] = '0'; where n goes from 3 down to 0. 1 MF_BB[n], means BYPASS_BALLAST_LV[n] = NOT nBYPASS_BALLAST_LV[n]; where n goes from 3 down to 0.

39.7.4 PMUCTRL fault monitor register (PMUCTRL_FAULT)

Address: Base + 0x74

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	BB_LV				0	0	0	0	0	0	0	0	0	FLNCF	IONCF	RENCF
W																
Reset	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
R	0	0	0	0	0	0	0	0	0	0	0	0	0	LHCF	LNCF	HNCF
W																
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 39-6. PMUCTRL fault monitor register (PMUCTRL_FAULT)

Table 39-7. PMUCTRL_FAULT field descriptions

Field	Description
BB_LV	Bypass Ballast Low Voltage. This field monitors the NOT of nBYPASS_BALLAST_LV[3:0] bus. 0 BB_LV[n] = NOT (nBYPASS_BALLAST_LV[n] = ‘1’); where n goes from 3 down to 0 1 BB_LV[n] = NOT (nBYPASS_BALLAST_LV[n] = ‘0’); where n goes from 3 down to 0
FLNCF	Flash memory voltage monitor non-critical fault 0 No flash memory voltage monitor fault is active 1 Flash memory voltage monitor fault is active
IONCF	IO voltage monitor non-critical fault 0 No IO voltage monitor fault is active 1 IO voltage monitor fault is active
RENCF	Regulator voltage monitor non-critical fault 0 No regulator voltage monitor fault is active 1 Regulator voltage monitor fault is active
LHCF	Low high voltage detector critical fault 0 No low high voltage fault is active 1 Low high voltage fault is active
LNCF	Low voltage detector non-critical fault 0 No low voltage fault is active 1 Low voltage fault is active
HNCF	High voltage detector non-critical fault 0 No high voltage fault is active 1 High voltage fault is active

39.7.5 PMUCTRL interrupt request status register (PMUCTRL_IRQS)

This register allows you to read and clear the various PMU interrupt status bits. To clear these bits, write a ‘1’ to them.

Table 40-5. PXS20 register protection (continued)

Module	Register	Size	Offset	Protect size
eDMA	CHCONFIG13	8	0xD	8-bit
eDMA	CHCONFIG14	8	0xE	8-bit
eDMA	CHCONFIG15	8	0xF	8-bit
DSPIA	MCR	32	0x0	32-bit
DSPIA	TCR	32	0x8	32-bit
DSPIA	CTAR0	32	0xC	32-bit
DSPIA	CTAR1	32	0x10	32-bit
DSPIA	CTAR2	32	0x14	32-bit
DSPIA	CTAR3	32	0x18	32-bit
DSPIA	RSER	32	0x30	32-bit
DSPIB	MCR	32	0x0	32-bit
DSPIB	TCR	32	0x8	32-bit
DSPIB	CTAR0	32	0xC	32-bit
DSPIB	CTAR1	32	0x10	32-bit
DSPIB	CTAR2	32	0x14	32-bit
DSPIB	CTAR3	32	0x18	32-bit
DSPIB	RSER	32	0x30	32-bit
DSPIC	MCR	32	0x0	32-bit
DSPIC	TCR	32	0x8	32-bit
DSPIC	CTAR0	32	0xC	32-bit
DSPIC	CTAR1	32	0x10	32-bit
DSPIC	CTAR2	32	0x14	32-bit
DSPIC	CTAR3	32	0x18	32-bit
DSPIC	RSER	32	0x30	32-bit
eTIMER0	COMP10	16	0x0	16-bit
eTIMER0	COMP20	16	0x2	16-bit
eTIMER0	LOAD0	16	0x8	16-bit
eTIMER0	CTRL10	16	0xE	16-bit
eTIMER0	CTRL20	16	0x10	16-bit
eTIMER0	CTRL30	16	0x12	16-bit
eTIMER0	INTeDMA0	16	0x16	16-bit
eTIMER0	CMPLD10	16	0x18	16-bit
eTIMER0	CMPLD20	16	0x1A	16-bit
eTIMER0	CCCTRL0	16	0x1C	16-bit
eTIMER0	FILT0	16	0x1E	16-bit

reset mechanism requires two consecutive writes with predefined data patterns from the same processor to force the clearing of the IRQ notification(s). The required access pattern is:

1. A processor performs a 16-bit write to the SEMA4_RSTNTF memory location. The most significant byte (SEMA4_RSTNTF[RSTNDP]) must be 0x47; the least significant byte is a “don’t care” for this reference.
2. The same processor performs a second 16-bit write to the SEMA4_RSTNTF location. For this write, the upper byte (SEMA4_RSTNTF[RSTNDP]) is the logical complement of the first data pattern (0xb8) and the lower byte (SEMA4_RSTNTF[RSTNTN]) specifies the notification(s) to be reset. This field can specify a single notification be cleared or that all notifications are cleared.
3. Reads of the SEMA4_RSTNTF location return information on the 2-bit state machine (SEMA4_RSTNTF[RSTNSM]) that implements this function, the bus master performing the reset (SEMA4_RSTNTF[RSTNMS]) and the notification number(s) last cleared (SEMA4_RSTNTF[RSTNTN]). Reads of the SEMA4_RSTNTF register do not affect the secure reset finite state machine in any manner.

Offset: SEMA4_BASE + 0x0104 Access: User read/write

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
R	0	0	RSTNSM		0	RSTNMS			RSTNTN							
W	RSTNDP															
Reset	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 43-6. Semaphores (secure) Reset IRQ notification (SEMA4_RSTNTF)

46.4.2 Reset effects on SRAM accesses on cut1

On cut1, asynchronous reset will possibly corrupt RAM if it asserts during a read or write operation to SRAM. The completion of that access depends on the cycle at which the reset occurs. Data read from or written to SRAM before the reset event occurred is retained, and no other address locations are accessed or changed. In case of no access ongoing when reset occurs, the RAM corruption does not happen.

Instead synchronous reset (SW reset) should be used in controlled function (without RAM accesses) in case initialization procedure is needed without RAM initialization.

46.5 Functional description

ECC checks are performed during the read portion of an SRAM ECC read/write (R/W) operation, and ECC calculations are performed during the write portion of a read/write (R/W) operation. Because the ECC bits can contain random data after the device is powered on, the SRAM must be initialized by executing 32-bit write operations prior any read accesses. This is also true for implicit read accesses caused by any write accesses of less than 32-bit as discussed in "ECC Mechanism".

46.6 Initialization and Application Informations

To use the SRAM, the ECC must check all bits that require initialization after power on. All writes must specify an even number of registers performed on 32-bit word-aligned boundaries. If the write is not the entire 32-bits (8-, or 16-bits), a read / modify / write operation is generated that checks the ECC value upon the read. Refer to Section 46.4, SRAM ECC mechanism.

NOTE

You *must* initialize SRAM, even if the application does not use ECC reporting.

47.5.2.7 Interrupt Filter Enable Register (IFER)

This register is used to enable or disable a digital filter counter on the interrupt pads to filter out glitches on the inputs.

Address: Base + 0x0030 Access: User read/write

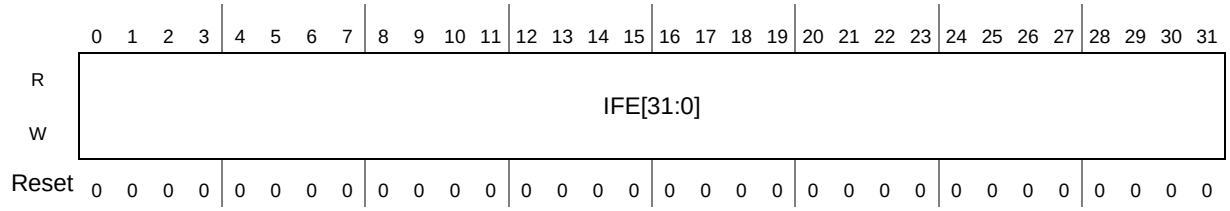


Figure 47-9. Interrupt Filter Enable Register (IFER)

Table 47-9. IFER Field Descriptions

Field	Description
IFE[x]	Enable digital glitch filter on the interrupt pad input. 1: Filter is enabled 0: Filter is disabled

47.5.2.8 Pad Configuration Registers (PCR0–PCR132)

The Pad Configuration Registers allow configuration of the static electrical and functional characteristics associated with I/O pads. Each PCR controls the characteristics of a single pad. See Table 3-5 for the mapping of PCR to pads.

Address: Base + 0x0040 (PCR0)(122 registers) Access: User read/write
Base + 0x0042 (PCR1)
...
Base + 0x0148 (PCR132)

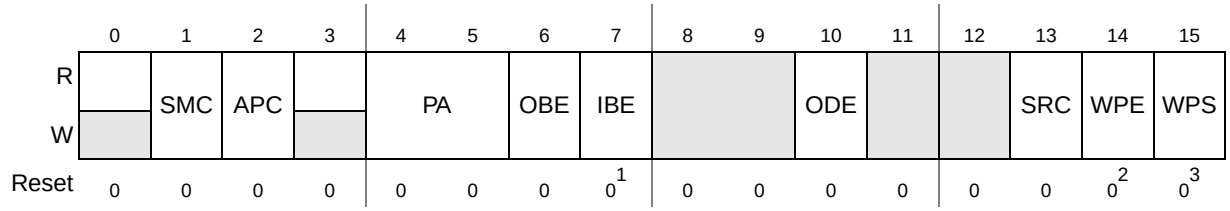


Figure 47-10. Pad Configuration Registers (PCR0)

NOTES:
1 The reset value of the IBE- bit of the Pad Configuration Registers PCR2, 3, 4, 21 is '1' in distinction from the remaining PCR registers
2 The reset value of the WPE- bit of the Pad Configuration Registers PCR2, 3, 4, 21 is '1' in distinction from the remaining PCR registers
3 The reset value of the WPS- bit of the Pad Configuration Registers PCR21 is '1' in distinction from the remaining PCR registers

NOTE

16- and 32-bit accesses are supported.

Appendix A
Revision History

This appendix describes corrections to the *PXS20 Microcontroller Reference Manual*. For convenience, the corrections are grouped by revision.

Since this is the first revision of this document, there are no changes.