



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	80MHz
Connectivity	CANbus, EBI/EMI, I ² C, IrDA, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	59
Program Memory Size	64KB (64K x 8)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°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=mc9s12xd64caa

Section Number	Title	Page
20.4	Functional Description	768
20.4.1	S12XDBG Operation	768
20.4.2	Comparator Modes	769
20.4.3	Trigger Modes	772
20.4.4	State Sequence Control	774
20.4.5	Trace Buffer Operation	775
20.4.6	Tagging	782
20.4.7	Breakpoints	783

Chapter 21

External Bus Interface (S12XEBIV2)

21.1	Introduction	787
21.1.1	Features	787
21.1.2	Modes of Operation	787
21.1.3	Block Diagram	788
21.2	External Signal Description	788
21.3	Memory Map and Register Definition	790
21.3.1	Module Memory Map	790
21.3.2	Register Descriptions	790
21.4	Functional Description	794
21.4.1	Operating Modes and External Bus Properties	794
21.4.2	Internal Visibility	795
21.4.3	Accesses to Port Replacement Registers	798
21.4.4	Stretched External Bus Accesses	798
21.4.5	Data Select and Data Direction Signals	799
21.4.6	Low-Power Options	801
21.5	Initialization/Application Information	801
21.5.1	Normal Expanded Mode	802
21.5.2	Emulation Modes	803

Chapter 22

DP512 Port Integration Module (S12XDP512PIMV2)

22.1	Introduction	807
22.1.1	Features	808
22.1.2	Block Diagram	808
22.2	External Signal Description	810
22.2.1	Signal Properties	810
22.3	Memory Map and Register Definition	817
22.3.1	Module Memory Map	817
22.3.2	Register Descriptions	820
22.4	Functional Description	881
22.4.1	Registers	881

Table 1-3. Device Internal Resources (see Figure 1-4)

Device	RAMSIZE/ RAM_LOW	EEPROMSIZE/ EEPROM_LOW	FLASHSIZE0/ FLASH_LOW	FLASHSIZE1/ FLASH_HIGH
9S12XDT384	20K 0x0F_B000	4K 0x13_F000	128K 0x79_FFFF	256K 0x7C_0000
9S12XDQ256	16K 0x0F_C000	4K 0x13_F000	128K 0x79_FFFF	128K 0x7E_0000
9S12XDT256	16K 0x0F_C000	4K 0x13_F000		
9S12XD256	14K 0x0F_C800	4K 0x13_F000		
9S12XA256	16K 0x0F_C000	4K 0x13_F000		
9S12XB256	10K 0x0F_D800	2K 0x13_F800		

Table 2-2. CRGFLG Field Descriptions (continued)

Field	Description
5 LVRF	Low Voltage Reset Flag — If low voltage reset feature is not available (see device specification) LVRF always reads 0. LVRF is set to 1 when a low voltage reset occurs. This flag can only be cleared by writing a 1. Writing a 0 has no effect. 0 Low voltage reset has not occurred. 1 Low voltage reset has occurred.
4 LOCKIF	PLL Lock Interrupt Flag — LOCKIF is set to 1 when LOCK status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (LOCKIE = 1), LOCKIF causes an interrupt request. 0 No change in LOCK bit. 1 LOCK bit has changed.
3 LOCK	Lock Status Bit — LOCK reflects the current state of PLL lock condition. This bit is cleared in self clock mode. Writes have no effect. 0 PLL VCO is not within the desired tolerance of the target frequency. 1 PLL VCO is within the desired tolerance of the target frequency.
2 TRACK	Track Status Bit — TRACK reflects the current state of PLL track condition. This bit is cleared in self clock mode. Writes have no effect. 0 Acquisition mode status. 1 Tracking mode status.
1 SCMIF	Self Clock Mode Interrupt Flag — SCMIF is set to 1 when SCM status bit changes. This flag can only be cleared by writing a 1. Writing a 0 has no effect. If enabled (SCMIE = 1), SCMIF causes an interrupt request. 0 No change in SCM bit. 1 SCM bit has changed.
0 SCM	Self Clock Mode Status Bit — SCM reflects the current clocking mode. Writes have no effect. 0 MCU is operating normally with OSCCLK available. 1 MCU is operating in self clock mode with OSCCLK in an unknown state. All clocks are derived from PLLCLK running at its minimum frequency f_{SCM} .

4.3.2.15 Port Data Register 1 (PORTAD1)

The data port associated with the ATD is input-only. The port pins are shared with the analog A/D inputs AN7-0.

	7	6	5	4	3	2	1	0
R	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
W								
Reset	1	1	1	1	1	1	1	1
Pin Function	AN 7	AN6	AN5	AN4	AN3	AN2	AN1	AN0


 = Unimplemented or Reserved

Figure 4-17. Port Data Register 1 (PORTAD1)

Read: Anytime

Write: Anytime, no effect

The A/D input channels may be used for general-purpose digital input.

Table 4-26. PORTAD1 Field Descriptions

Field	Description
7:0 PTAD[7:8]	A/D Channel x (ANx) Digital Input Bits — If the digital input buffer on the ANx pin is enabled (IENx=1) or channel x is enabled as external trigger (ETRIGE = 1, ETRIGCH[3-0] = x, ETRIGSEL = 0) read returns the logic level on ANx pin (signal potentials not meeting V _{IL} or V _{IH} specifications will have an indeterminate value)). If the digital input buffers are disabled (IENx = 0) and channel x is not enabled as external trigger, read returns a “1”. Reset sets all PORTAD1 bits to “1”.



BRA

Branch Always

BRA

Operation

$PC + \$0002 + (REL10 \ll 1) \Rightarrow PC$

Branches always

CCR Effects

N	Z	V	C
—	—	—	—

- N: Not affected.
- Z: Not affected.
- V: Not affected.
- C: Not affected.

Code and CPU Cycles

Source Form	Address Mode	Machine Code							Cycles
BRA REL10	REL10	0	0	1	1	1	1	REL10	PP

Register Name		Bit 7	6	5	4	3	2	1	Bit 0
0x0002 CANBTR0	R W	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
0x0003 CANBTR1	R W	SAMP	TSEG22	TSEG21	TSEG20	TSEG13	TSEG12	TSEG11	TSEG10
0x0004 CANRFLG	R W	WUPIF	CSCIF	RSTAT1	RSTAT0	TSTAT1	TSTAT0	OVRIF	RXF
0x0005 CANRIER	R W	WUPIE	CSCIE	RSTATE1	RSTATE0	TSTATE1	TSTATE0	OVRIE	RXFIE
0x0006 CANTFLG	R W	0	0	0	0	0	TXE2	TXE1	TXE0
0x0007 CANTIER	R W	0	0	0	0	0	TXEIE2	TXEIE1	TXEIE0
0x0008 CANTARQ	R W	0	0	0	0	0	ABTRQ2	ABTRQ1	ABTRQ0
0x0009 CANTAACK	R W	0	0	0	0	0	ABTAK2	ABTAK1	ABTAK0
0x000A CANTBSEL	R W	0	0	0	0	0	TX2	TX1	TX0
0x000B CANIDAC	R W	0	0	IDAM1	IDAM0	0	IDHIT2	IDHIT1	IDHIT0
0x000C Reserved	R W	0	0	0	0	0	0	0	0
0x000D CANMISC	R W	0	0	0	0	0	0	0	BOHOLD
0x000E CANRXERR	R W	RXERR7	RXERR6	RXERR5	RXERR4	RXERR3	RXERR2	RXERR1	RXERR0
0x000F CANTXERR	R W	TXERR7	TXERR6	TXERR5	TXERR4	TXERR3	TXERR2	TXERR1	TXERR0
0x0010–0x0013 CANIDAR0–3	R W	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0

= Unimplemented or Reserved
 u = Unaffected

Figure 10-3. MSCAN Register Summary (continued)

Chapter 12

Serial Peripheral Interface (S12SPIV4)

12.1 Introduction

The SPI module allows a duplex, synchronous, serial communication between the MCU and peripheral devices. Software can poll the SPI status flags or the SPI operation can be interrupt driven.

12.1.1 Glossary of Terms

SPI	Serial Peripheral Interface
SS	Slave Select
SCK	Serial Clock
MOSI	Master Output, Slave Input
MISO	Master Input, Slave Output
MOMI	Master Output, Master Input
SISO	Slave Input, Slave Output

12.1.2 Features

The SPI includes these distinctive features:

- Master mode and slave mode
- Bidirectional mode
- Slave select output
- Mode fault error flag with CPU interrupt capability
- Double-buffered data register
- Serial clock with programmable polarity and phase
- Control of SPI operation during wait mode

12.1.3 Modes of Operation

The SPI functions in three modes: run, wait, and stop.

- Run mode
This is the basic mode of operation.
- Wait mode

17.4.4.2 Access Conflicts on Target Buses

The arbitration scheme allows only one master to be connected to a target at any given time. The following rules apply when prioritizing accesses from different masters to the same target bus:

- CPU always has priority over XGATE.
- BDM access has priority over XGATE.
- XGATE access to PRU registers constitutes a special case. It is always granted and stalls the CPU and BDM for its duration.
- In emulation modes all internal accesses are visible on the external bus as well.
- During access to the PRU registers, the external bus is reserved.

17.4.5 Interrupts

17.4.5.1 Outgoing Interrupt Requests

The following interrupt requests can be triggered by the MMC module:

CPU access violation: The CPU access violation signals to the CPU detection of an error condition in the CPU application code which is resulted in write access to the protected XGATE RAM area (see [Section 1.4.3.2, “Illegal CPU Accesses”](#)).

17.5 Initialization/Application Information

17.5.1 CALL and RTC Instructions

CALL and RTC instructions are uninterruptable CPU instructions that automate page switching in the program page window. The CALL instruction is similar to the JSR instruction, but the subroutine that is called can be located anywhere in the local address space or in any Flash or ROM page visible through the program page window. The CALL instruction calculates and stacks a return address, stacks the current PPAGE value and writes a new instruction-supplied value to the PPAGE register. The PPAGE value controls which of the 256 possible pages is visible through the 16 Kbyte program page window in the 64 Kbyte local CPU memory map. Execution then begins at the address of the called subroutine.

During the execution of the CALL instruction, the CPU performs the following steps:

1. Writes the current PPAGE value into an internal temporary register and writes the new instruction-supplied PPAGE value into the PPAGE register
2. Calculates the address of the next instruction after the CALL instruction (the return address) and pushes this 16-bit value onto the stack
3. Pushes the temporarily stored PPAGE value onto the stack
4. Calculates the effective address of the subroutine, refills the queue and begins execution at the new address



Loop1 mode only inhibits consecutive duplicate source address entries that would typically be stored in most tight looping constructs. It does not inhibit repeated entries of destination addresses or vector addresses, since repeated entries of these would most likely indicate a bug in the user's code that the DBG module is designed to help find.

NOTE

In certain very tight loops, the source address will have already been fetched again before the background comparator is updated. This results in the source address being stored twice before further duplicate entries are suppressed. This condition occurs with branch-on-bit instructions when the branch is fetched by the first P-cycle of the branch or with loop-construct instructions in which the branch is fetched with the first or second P cycle. See examples below:

LOOP	INX		;1-byte instruction fetched by 1st P-cycle of BRCLR
	BRCLR	CMPTMP, #0c, LOOP	;the BRCLR instruction also will be fetched by 1st P-cycle of BRCLR
LOOP2	BRN*		; 2-byte instruction fetched by 1st P-cycle of DBNE
	NOP		; 1-byte instruction fetched by 2nd P-cycle of DBNE
	DBNE	A, LOOP2	; this instruction also fetched by 2nd P-cycle of DBNE

19.4.5.2.3 Detail Mode

In detail mode, address and data for all memory and register accesses is stored in the trace buffer. In the case of XGATE tracing this means that initialization of the R1 register during a vector fetch is not traced. This mode is intended to supply additional information on indexed, indirect addressing modes where storing only the destination address would not provide all information required for a user to determine where the code is in error. This mode also features information byte storage to the trace buffer, for each address byte storage. The information byte indicates the size of access (word or byte), the type of access (read or write).

When tracing CPU activity in detail mode, all cycles are traced except those when the CPU is either in a free or opcode fetch cycle. In this mode the XGATE program counter is also traced to provide a snapshot of the XGATE activity. CXINF information byte bits indicate the type of XGATE activity occurring at the time of the trace buffer entry. When tracing CPU activity alone in detail mode, the address range can be limited to a range specified by the TRANGE bits in DBGTCR. This function uses comparators C and D to define an address range inside which CPU activity should be traced (see [Table 19-10](#)). Thus, the traced CPU activity can be restricted to register range accesses.

When tracing XGATE activity in detail mode, all cycles apart from opcode fetch and free cycles are stored to the trace buffer. Additionally the CPU program counter is stored at the time of the XGATE trace buffer entry to provide a snapshot of CPU activity.

19.4.5.3 Trace Buffer Organization

The buffer can be used to trace either from CPU, from XGATE or from both sources. An "X" prefix denotes information from the XGATE module, a "C" prefix denotes information from the CPU. ADRH,ADRM,ADRL denote address high, middle and low byte respectively. INF bytes contain control

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CFREE	CSZ	CRW	COCF	XACK	XSZ	XRW	XOCF

Figure 19-26. Information Byte CXINF

This describes the format of the information byte used only when tracing from CPU or XGATE in detail mode. When tracing from the CPU in detail mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The XGATE entry stored on the same line is a snapshot of the XGATE program counter. In this case the CSZ and CRW bits indicate the type of access being made by the CPU, while the XACK and XOCF bits indicate if the simultaneous XGATE cycle is a free cycle (no bus acknowledge) or opcode fetch cycle. Similarly when tracing from the XGATE in detail mode, information is stored to the trace buffer on all cycles except opcode fetch and free cycles. The CPU entry stored on the same line is a snapshot of the CPU program counter. In this case the XSZ and XRW bits indicate the type of access being made by the XGATE, while the CFREE and COCF bits indicate if the simultaneous CPU cycle is a free cycle or opcode fetch cycle.

Table 19-42. CXINF Field Descriptions

Field	Description
7 CREE	CPU Free Cycle Indicator — This bit indicates if the stored CPU address corresponds to a free cycle. This bit only contains valid information when tracing the XGATE accesses in detail mode. 0 Stored information corresponds to free cycle 1 Stored information does not correspond to free cycle
6 CSZ	Access Type Indicator — This bit indicates if the access was a byte or word size access. This bit only contains valid information when tracing CPU activity in detail mode. 0 Word Access 1 Byte Access
5 CRW	Read Write Indicator — This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing CPU activity in detail mode. 0 Write Access 1 Read Access
4 COCF	CPU Opcode Fetch Indicator — This bit indicates if the stored address corresponds to an opcode fetch cycle. This bit only contains valid information when tracing the XGATE accesses in detail mode. 0 Stored information does not correspond to opcode fetch cycle 1 Stored information corresponds to opcode fetch cycle
3 XACK	XGATE Access Indicator — This bit indicates if the stored XGATE address corresponds to a free cycle. This bit only contains valid information when tracing the CPU accesses in detail mode. 0 Stored information corresponds to free cycle 1 Stored information does not correspond to free cycle
2 XSZ	Access Type Indicator — This bit indicates if the access was a byte or word size access. This bit only contains valid information when tracing XGATE activity in detail mode. 0 Word Access 1 Byte Access

Table 19-42. CXINF Field Descriptions (continued)

Field	Description
1 XRW	Read Write Indicator — This bit indicates if the corresponding stored address corresponds to a read or write access. This bit only contains valid information when tracing XGATE activity in detail mode. 0 Read/Write Access 1 Access
0 XOCF	XGATE Opcode Fetch Indicator — This bit indicates if the stored address corresponds to an opcode fetch cycle. This bit only contains valid information when tracing the CPU accesses in detail mode. 0 Stored information does not correspond to opcode fetch cycle 1 Stored information corresponds to opcode fetch cycle

19.4.5.3.2 Reading Data from Trace Buffer

The data stored in the trace buffer can be read using either the background debug module (BDM) module or the CPU provided the DBG module is not armed, is configured for tracing (at least one TSOURCE bit is set) and the system not secured. When the ARM bit is written to 1 the trace buffer is locked to prevent reading. The trace buffer can only be unlocked for reading by a single aligned word write to DBGTB when the module is disarmed. Multiple writes to the DBGTB are not allowed since they increment the pointer.

The trace buffer can only be read through the DBGTB register using aligned word reads, any byte or misaligned reads return 0 and do not cause the trace buffer pointer to increment to the next trace buffer address. The trace buffer data is read out first-in first-out. By reading CNT in DBGCNT the number of valid 64-bit lines can be determined. DBGCNT will not decrement as data is read.

Whilst reading an internal pointer is used to determine the next line to be read. After a tracing session, the pointer points to the oldest data entry, thus if no overflow has occurred, the pointer points to line0, otherwise it points to the line with the oldest entry. The pointer is initialized by each aligned write to DBGTBH to point to the oldest data again. This enables an interrupted trace buffer read sequence to be easily restarted from the oldest data entry.

The least significant word of each 64-bit wide array line is read out first. This corresponds to the bytes 1 and 0 of [Table 19-39](#). The bytes containing invalid information (shaded in [Table 19-39](#)) are also read out.

Reading the trace buffer while the DBG module is armed will return invalid data and no shifting of the RAM pointer will occur. Reading the trace buffer is not possible if both TSOURCE bits are cleared.



Stretched accesses are controlled by:

1. EXSTR[2:0] bits in the EBICTL1 register configuring fixed amount of stretch cycles
2. Activation of the external wait feature by EWAITE in EBICTL1 register
3. Assertion of the external $\overline{\text{EWAITE}}$ signal when EWAITE = 1

The EXSTR[2:0] control bits can be programmed for generation of a fixed number of 1 to 8 stretch cycles. If the external wait feature is enabled, the minimum number of additional stretch cycles is 2. An arbitrary amount of stretch cycles can be added using the $\overline{\text{EWAITE}}$ input.

$\overline{\text{EWAITE}}$ needs to be asserted at least for a minimal specified time window within an external access cycle for the internal logic to detect it and add a cycle (refer to electrical characteristics). Holding it for additional cycles will cause the external bus access to be stretched accordingly.

Write accesses are stretched by holding the initiator in its current state for additional cycles as programmed and controlled by external wait after the data have been driven out on the external bus. This results in an extension of time the bus signals and the related control signals are valid externally.

Read data are not captured by the system in normal expanded mode until the specified setup time before the $\overline{\text{RE}}$ rising edge.

Read data are not captured in emulation expanded mode until the specified setup time before the falling edge of ECLK.

In emulation expanded mode, accesses to the internal flash or the emulation memory (determined by EROMON and ROMON bits; see S12X_MMC section for details) always take 1 cycle and stretching is not supported. In case the internal flash is taken out of the map in user applications, accesses are stretched as programmed and controlled by external wait.

21.4.5 Data Select and Data Direction Signals

The S12X_EBI supports byte and word accesses at any valid external address. The big endian system of the MCU is extended to the external bus; however, word accesses are restricted to even aligned addresses. The only exception is the visibility of misaligned word accesses to addresses in the internal RAM as this module exclusively supports these kind of accesses in a single cycle.

With the above restriction, a fixed relationship is implied between the address parity and the dedicated bus halves where the data are accessed: DATA[15:8] is related to even addresses and DATA[7:0] is related to odd addresses.

In expanded modes the data access type is externally determined by a set of control signals, i.e., data select and data direction signals, as described below. The data select signals are not available if using the external bus interface with an 8-bit data bus.

21.4.5.1 Normal Expanded Mode

In normal expanded mode, the external signals $\overline{\text{RE}}$, $\overline{\text{WE}}$, $\overline{\text{UDS}}$, $\overline{\text{LDS}}$ indicate the access type (read/write), data size and alignment of an external bus access (Table 21-17).



22.3.2.63 Port AD0 Data Direction Register 1 (DDR1AD0)

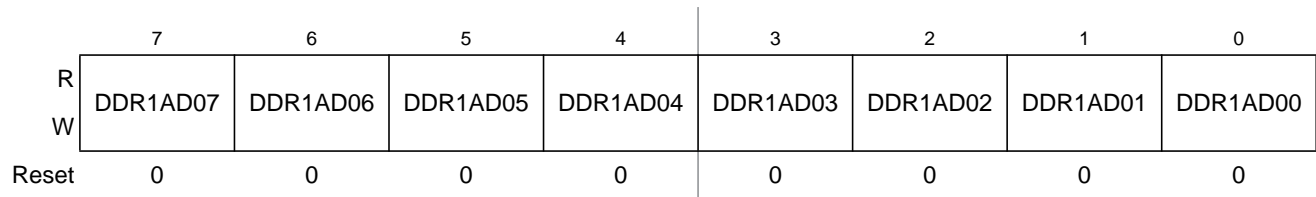


Figure 22-65. Port AD0 Data Direction Register 1 (DDR1AD0)

Read: Anytime.

Write: Anytime.

This register configures pins PAD[07:00] as either input or output.

Table 22-58. DDR1AD0 Field Descriptions

Field	Description
7–0 DDR1AD0[7:0]	<p>Data Direction Port AD0 Register 1</p> <p>0 Associated pin is configured as input. 1 Associated pin is configured as output.</p> <p>Note: Due to internal synchronization circuits, it can take up to 2 bus clock cycles until the correct value is read on PTAD01 register, when changing the DDR1AD0 register.</p> <p>Note: To use the digital input function on port AD0 the ATD0 digital input enable register (ATD0DIEN) has to be set to logic level “1”.</p>

23.0.5.11 S12X_EBI Ports, BKGD, VREGEN Pin Pull-up Control Register (PUCR)

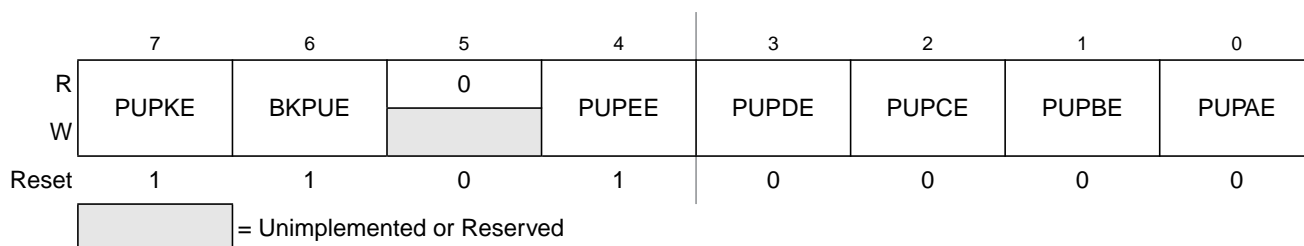


Figure 23-13. S12X_EBI Ports, BKGD, VREGEN Pin Pull-up Control Register (PUCR)

Read: Anytime in single-chip modes.

Write: Anytime, except BKPUE which is writable in special test mode only.

This register is used to enable pull-up devices for the associated ports A, B, C, D, E, and K. Pull-up devices are assigned on a per-port basis and apply to any pin in the corresponding port currently configured as an input.

Table 23-14. PUCR Field Descriptions

Field	Description
7 PUPKE	Pull-up Port K Enable 0 Port K pull-up devices are disabled. 1 Enable pull-up devices for Port K input pins.
6 BKPUE	BKGD and VREGEN Pin Pull-up Enable 0 BKGD and V _{REGEN} pull-up devices are disabled. 1 Enable pull-up devices on BKGD and V _{REGEN} pins.
4 PUPEE	Pull-up Port E Enable 0 Port E pull-up devices on bit 7, 4–0 are disabled. 1 Enable pull-up devices for Port E input pins bits 7, 4–0. Note: Bits 5 and 6 of Port E have pull-down devices which are only enabled during reset. This bit has no effect on these pins.
3 PUPDE	Pull-up Port D Enable 0 Port D pull-up devices are disabled. 1 Enable pull-up devices for all Port D input pins.
2 PUPCE	Pull-up Port C Enable 0 Port C pull-up devices are disabled. 1 Enable pull-up devices for all Port C input pins.
1 PUPBE	Pull-up Port B Enable 0 Port B pull-up devices are disabled. 1 Enable pull-up devices for all Port B input pins.
0 PUPAE	Pull-up Port A Enable 0 Port A pull-up devices are disabled. 1 Enable pull-up devices for all Port A input pins.

26.4.2.5 Sector Erase Abort Command

The sector erase abort operation will terminate the active sector erase or sector modify operation so that other sectors in an EEPROM block are available for read and program operations without waiting for the sector erase or sector modify operation to complete.

An example flow to execute the sector erase abort operation is shown in [Figure 26-22](#). The sector erase abort command write sequence is as follows:

1. Write to any EEPROM memory address to start the command write sequence for the sector erase abort command. The address and data written are ignored.
2. Write the sector erase abort command, 0x47, to the ECMD register.
3. Clear the CBEIF flag in the ESTAT register by writing a 1 to CBEIF to launch the sector erase abort command.

If the sector erase abort command is launched resulting in the early termination of an active sector erase or sector modify operation, the ACCERR flag will set once the operation completes as indicated by the CCIF flag being set. The ACCERR flag sets to inform the user that the EEPROM sector may not be fully erased and a new sector erase or sector modify command must be launched before programming any location in that specific sector. If the sector erase abort command is launched but the active sector erase or sector modify operation completes normally, the ACCERR flag will not set upon completion of the operation as indicated by the CCIF flag being set. If the sector erase abort command is launched after the sector modify operation has completed the sector erase step, the program step will be allowed to complete. The maximum number of cycles required to abort a sector erase or sector modify operation is equal to four EECLK periods (see [Section 26.4.1.1, “Writing the ECLKDIV Register”](#)) plus five bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set.

NOTE

Since the ACCERR bit in the ESTAT register may be set at the completion of the sector erase abort operation, a command write sequence is not allowed to be buffered behind a sector erase abort command write sequence. The CBEIF flag will not set after launching the sector erase abort command to indicate that a command should not be buffered behind it. If an attempt is made to start a new command write sequence with a sector erase abort operation active, the ACCERR flag in the ESTAT register will be set. A new command write sequence may be started after clearing the ACCERR flag, if set.

NOTE

The sector erase abort command should be used sparingly since a sector erase operation that is aborted counts as a complete program/erase cycle.

27.4.2.6 Sector Erase Abort Command

The sector erase abort operation will terminate the active sector erase operation so that other sectors in a Flash block are available for read and program operations without waiting for the sector erase operation to complete.

An example flow to execute the sector erase abort operation is shown in [Figure 27-31](#). The sector erase abort command write sequence is as follows:

1. Write to any Flash block address to start the command write sequence for the sector erase abort command. The address and data written are ignored.
2. Write the sector erase abort command, 0x47, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the sector erase abort command.

If the sector erase abort command is launched resulting in the early termination of an active sector erase operation, the ACCERR flag will set once the operation completes as indicated by the CCIF flag being set. The ACCERR flag sets to inform the user that the Flash sector may not be fully erased and a new sector erase command must be launched before programming any location in that specific sector. If the sector erase abort command is launched but the active sector erase operation completes normally, the ACCERR flag will not set upon completion of the operation as indicated by the CCIF flag being set. Therefore, if the ACCERR flag is not set after the sector erase abort command has completed, a Flash sector being erased when the abort command was launched will be fully erased. The maximum number of cycles required to abort a sector erase operation is equal to four FCLK periods (see [Section 27.4.1.1, “Writing the FCLKDIV Register”](#)) plus five bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. If sectors in multiple Flash blocks are being simultaneously erased, the sector erase abort operation will be applied to all active Flash blocks without writing to each Flash block in the sector erase abort command write sequence.

NOTE

Since the ACCERR bit in the FSTAT register may be set at the completion of the sector erase abort operation, a command write sequence is not allowed to be buffered behind a sector erase abort command write sequence. The CBEIF flag will not set after launching the sector erase abort command to indicate that a command should not be buffered behind it. If an attempt is made to start a new command write sequence with a sector erase abort operation active, the ACCERR flag in the FSTAT register will be set. A new command write sequence may be started after clearing the ACCERR flag, if set.

NOTE

The sector erase abort command should be used sparingly since a sector erase operation that is aborted counts as a complete program/erase cycle.

29.4.2.1 Erase Verify Command

The erase verify operation will verify that a Flash block is erased.

An example flow to execute the erase verify operation is shown in [Figure 29-23](#). The erase verify command write sequence is as follows:

1. Write to a Flash block address to start the command write sequence for the erase verify command. The address and data written will be ignored.
2. Write the erase verify command, 0x05, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the erase verify command.

After launching the erase verify command, the CCIF flag in the FSTAT register will set after the operation has completed unless a new command write sequence has been buffered. The number of bus cycles required to execute the erase verify operation is equal to the number of addresses in a Flash block plus 14 bus cycles as measured from the time the CBEIF flag is cleared until the CCIF flag is set. Upon completion of the erase verify operation, the BLANK flag in the FSTAT register will be set if all addresses in the selected Flash block are verified to be erased. If any address in a selected Flash block is not erased, the erase verify operation will terminate and the BLANK flag in the FSTAT register will remain clear. The MRDS bits in the FTSTMOD register will determine the sense-amp margin setting during the erase verify operation.

Appendix D Using L15Y Silicon

The following items should be considered when using L15Y Silicon:

- Do not write or read to registers which are marked “Reserved” in Table 1-1.
- Fill the interrupt vector locations which are marked “Reserved” in Table 1-12. according to your coding policies for unused interrupts
- L15Y Silicon includes two analog to digital converters ATD0 and ATD1. ATD0 channels 7 to 0 are connected to PAD07 to PAD00 and ATD1 channels 7 to 0 are connected to PAD15 to PAD08.
- L15Y Silicon integrates the S12X_DBG module Version 2. L15Y Silicon integrates the S12X_MMC module Version 2. This Version doesn’t support the following enhancement which is available on S12X_MMC Version 3:
 - S12XCPU and S12XBDM can access MCU resources which are on different target busses at the same time. I.E S12XCPU can access XSRAM and S12XBDM can access Register Space at the same time.