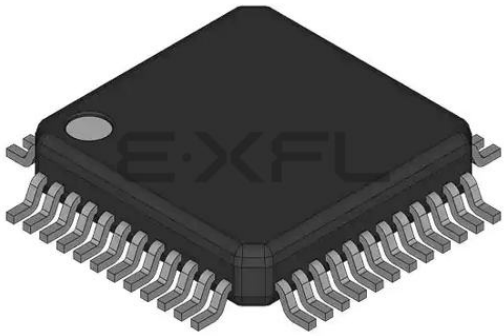


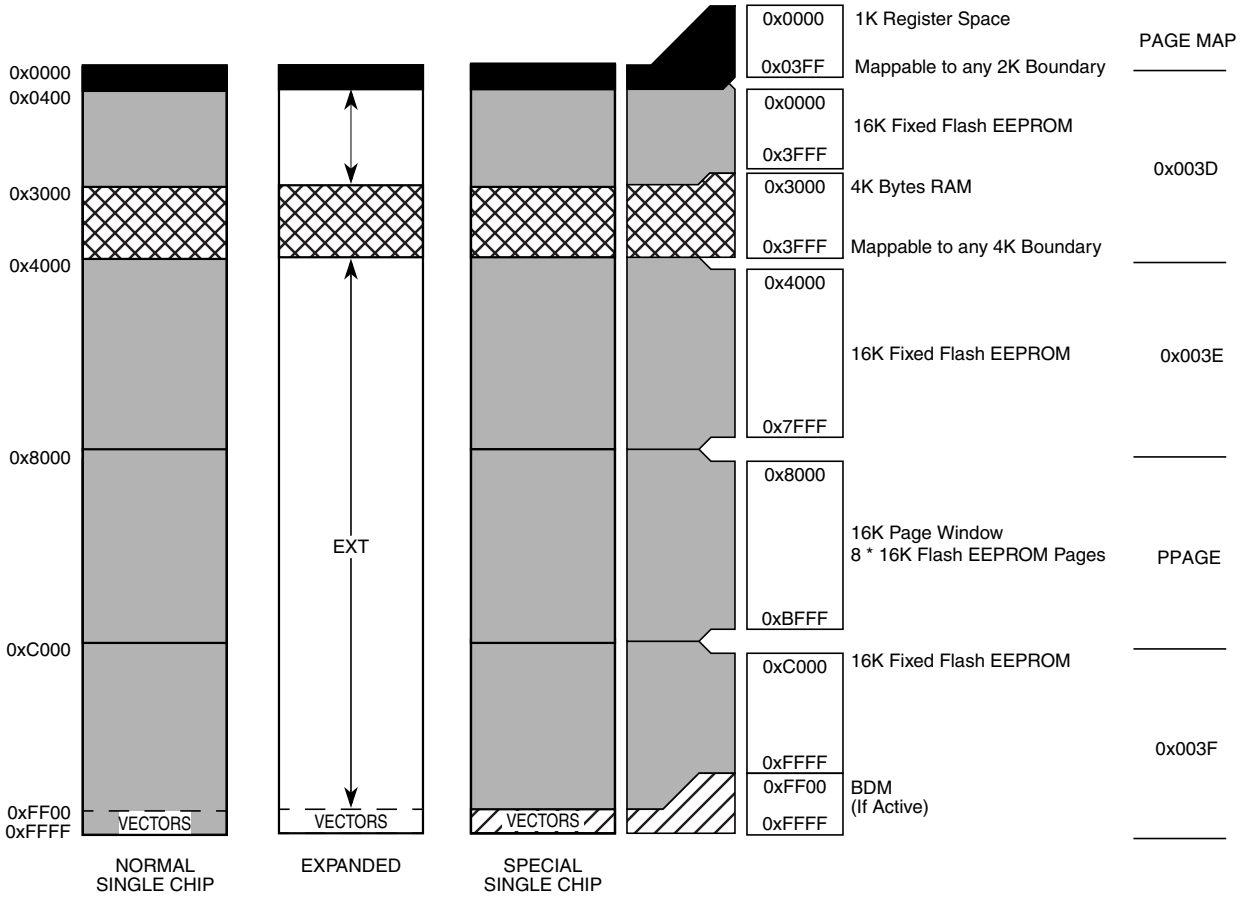
Welcome to E-XFL.COM

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	HCS12
Core Size	16-Bit
Speed	25MHz
Connectivity	EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	31
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.35V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	48-LQFP
Supplier Device Package	48-LQFP (7x7)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc9s12gc32cfae



The figure shows a useful map, which is not the map out of reset. After reset the map is:

0x0000–0x03FF: Register Space
0x0000–0x0FFF: 4K RAM (only 3K visible 0x0400–0x0FFF)

Flash erase sector size is 1024 bytes

Figure 1-2. MC9S12C128 and MC9S12GC128 User Configurable Memory Map

2.3.2.2.6 Port S Polarity Select Register (PPSS)

Module Base + 0x000D

	7	6	5	4	3	2	1	0
R	0	0	0	0	PPSS3	PPSS2	PPSS1	PPSS0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 2-15. Port S Polarity Select Register (PPSS)

Read: Anytime.

Write: Anytime.

Table 2-14. PPSS Field Descriptions

Field	Description
3–0 PPSS[3:0]	**Pull Select Port S** — This register selects whether a pull-down or a pull-up device is connected to the pin. 0 A pull-up device is connected to the associated port S pin, if enabled by the associated bit in register PERS and if the port is used as input or as wired-or output. 1 A pull-down device is connected to the associated port S pin, if enabled by the associated bit in register PERS and if the port is used as input.

2.3.2.2.7 Port S Wired-OR Mode Register (WOMS)

Module Base + 0x000E

	7	6	5	4	3	2	1	0
R	0	0	0	0	WOMS3	WOMS2	WOMS1	WOMS0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 2-16. Port S Wired-Or Mode Register (WOMS)

Read: Anytime.

Write: Anytime.

Table 2-15. WOMS Field Descriptions

Field	Description
3–0 WOMS[3:0]	**Wired-OR Mode Port S** — This register configures the output pins as wired-or. If enabled the output is driven active low only (open-drain). A logic level of “1” is not driven. This bit has no influence on pins used as inputs. 0 Output buffers operate as push-pull outputs. 1 Output buffers operate as open-drain outputs.

Table 5-2. ITCR Field Descriptions

Field	Description
4 WRTINT	**Write to the Interrupt Test Registers** Read: anytime Write: only in special modes and with I-bit mask and X-bit mask set. 0 Disables writes to the test registers; reads of the test registers will return the state of the interrupt inputs. 1 Disconnect the interrupt inputs from the priority decoder and use the values written into the ITEST registers instead. **Note:** Any interrupts which are pending at the time that WRTINT is set will remain until they are overwritten.
3:0 ADR[3:0]	**Test Register Select Bits** Read: anytime Write: anytime These bits determine which test register is selected on a read or write. The hexadecimal value written here will be the same as the upper nibble of the lower byte of the vector selects. That is, an “F” written into ADR[3:0] will select vectors 0xFFFE–0xFFF0 while a “7” written to ADR[3:0] will select vectors 0xFF7E–0xFF70.

5.3.2.2 Interrupt Test Registers

Module Base + 0x0016
Starting address location affected by INITRG register setting.

	7	6	5	4	3	2	1	0
R W	INTE	INTC	INTA	INT8	INT6	INT4	INT2	INT0
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 5-3. Interrupt TEST Registers (ITEST)

Read: Only in special modes. Reads will return either the state of the interrupt inputs of the interrupt sub-block (WRTINT = 0) or the values written into the TEST registers (WRTINT = 1). Reads will always return 0s in normal modes.

Write: Only in special modes and with WRTINT = 1 and CCR I mask = 1.

Chapter 6
Background Debug Module (BDMV4) Block Description

6.1 Introduction

This section describes the functionality of the background debug module (BDM) sub-block of the HCS12 core platform.

A block diagram of the BDM is shown in Figure 6-1.

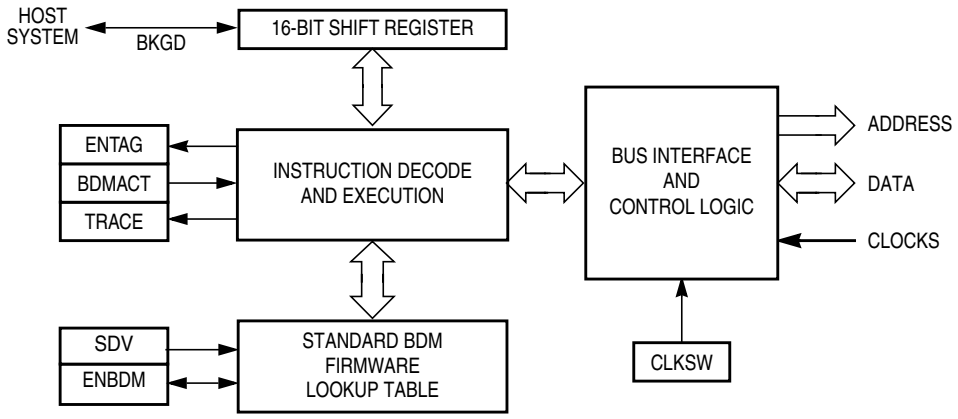


Figure 6-1. BDM Block Diagram

The background debug module (BDM) sub-block is a single-wire, background debug system implemented in on-chip hardware for minimal CPU intervention. All interfacing with the BDM is done via the BKGD pin.

BDMV4 has enhanced capability for maintaining synchronization between the target and host while allowing more flexibility in clock rates. This includes a sync signal to show the clock rate and a handshake signal to indicate when an operation is complete. The system is backwards compatible with older external interfaces.

6.1.1 Features

- Single-wire communication with host development system
- BDMV4 (and BDM2): Enhanced capability for allowing more flexibility in clock rates
- BDMV4: SYNC command to determine communication rate
- BDMV4: GO_UNTIL command
- BDMV4: Hardware handshake protocol to increase the performance of the serial communication
- Active out of reset in special single-chip mode

Table 6-2. BDMSTS Field Descriptions

Field	Description
7 ENBDM	**Enable BDM** — This bit controls whether the BDM is enabled or disabled. When enabled, BDM can be made active to allow firmware commands to be executed. When disabled, BDM cannot be made active but BDM hardware commands are allowed. 0 BDM disabled 1 BDM enabled **Note:** ENBDM is set by the firmware immediately out of reset in special single-chip mode. In secure mode, this bit will not be set by the firmware until after the EEPROM and FLASH erase verify tests are complete.
6 BDMACT	**BDM Active Status** — This bit becomes set upon entering BDM. The standard BDM firmware lookup table is then enabled and put into the memory map. BDMACT is cleared by a carefully timed store instruction in the standard BDM firmware as part of the exit sequence to return to user code and remove the BDM memory from the map. 0 BDM not active 1 BDM active
5 ENTAG	**Tagging Enable** — This bit indicates whether instruction tagging in enabled or disabled. It is set when the TAGGO command is executed and cleared when BDM is entered. The serial system is disabled and the tag function enabled 16 cycles after this bit is written. BDM cannot process serial commands while tagging is active. 0 Tagging not enabled or BDM active 1 Tagging enabled
4 SDV	**Shift Data Valid** — This bit is set and cleared by the BDM hardware. It is set after data has been transmitted as part of a firmware read command or after data has been received as part of a firmware write command. It is cleared when the next BDM command has been received or BDM is exited. SDV is used by the standard BDM firmware to control program flow execution. 0 Data phase of command not complete 1 Data phase of command is complete
3 TRACE	**TRACE1 BDM Firmware Command is Being Executed** — This bit gets set when a BDM TRACE1 firmware command is first recognized. It will stay set as long as continuous back-to-back TRACE1 commands are executed. This bit will get cleared when the next command that is not a TRACE1 command is recognized. 0 TRACE1 command is not being executed 1 TRACE1 command is being executed

The BDM hardware commands are listed in Table 6-5.

Table 6-5. Hardware Commands

Command	Opcode (hex)	Data	Description
BACKGROUND	90	None	Enter background mode if firmware is enabled. If enabled, an ACK will be issued when the part enters active background mode.
ACK_ENABLE	D5	None	Enable handshake. Issues an ACK pulse after the command is executed.
ACK_DISABLE	D6	None	Disable handshake. This command does not issue an ACK pulse.
READ_BD_BYTE	E4	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
READ_BD_WORD	EC	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table in map. Must be aligned access.
READ_BYTE	E0	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.
READ_WORD	E8	16-bit address 16-bit data out	Read from memory with standard BDM firmware lookup table out of map. Must be aligned access.
WRITE_BD_BYTE	C4	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Odd address data on low byte; even address data on high byte.
WRITE_BD_WORD	CC	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table in map. Must be aligned access.
WRITE_BYTE	C0	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Odd address data on low byte; even address data on high byte.
WRITE_WORD	C8	16-bit address 16-bit data in	Write to memory with standard BDM firmware lookup table out of map. Must be aligned access.

NOTE:
If enabled, ACK will occur when data is ready for transmission for all BDM READ commands and will occur after the write is complete for all BDM WRITE commands.

The READ_BD and WRITE_BD commands allow access to the BDM register locations. These locations are not normally in the system memory map but share addresses with the application in memory. To distinguish between physical memory locations that share the same address, BDM memory resources are enabled just for the READ_BD and WRITE_BD access cycle. This allows the BDM to access BDM locations unobtrusively, even if the addresses conflict with the application memory map.

6.4.4 Standard BDM Firmware Commands

Firmware commands are used to access and manipulate CPU resources. The system must be in active BDM to execute standard BDM firmware commands, see Section 6.4.2, “Enabling and Activating BDM.” Normal instruction execution is suspended while the CPU executes the firmware located in the standard BDM firmware lookup table. The hardware command BACKGROUND is the usual way to activate BDM.

As the system enters active BDM, the standard BDM firmware lookup table and BDM registers become visible in the on-chip memory map at 0xFF00–0xFFFF, and the CPU begins executing the standard BDM

The commands are described as follows:

- ACK_ENABLE — enables the hardware handshake protocol. The target will issue the ACK pulse when a CPU command is executed by the CPU. The ACK_ENABLE command itself also has the ACK pulse as a response.
- ACK_DISABLE — disables the ACK pulse protocol. In this case, the host needs to use the worst case delay time at the appropriate places in the protocol.

The default state of the BDM after reset is hardware handshake protocol disabled.

All the read commands will ACK (if enabled) when the data bus cycle has completed and the data is then ready for reading out by the BKGD serial pin. All the write commands will ACK (if enabled) after the data has been received by the BDM through the BKGD serial pin and when the data bus cycle is complete. See Section 6.4.3, “BDM Hardware Commands,” and Section 6.4.4, “Standard BDM Firmware Commands,” for more information on the BDM commands.

The ACK_ENABLE sends an ACK pulse when the command has been completed. This feature could be used by the host to evaluate if the target supports the hardware handshake protocol. If an ACK pulse is issued in response to this command, the host knows that the target supports the hardware handshake protocol. If the target does not support the hardware handshake protocol the ACK pulse is not issued. In this case, the ACK_ENABLE command is ignored by the target because it is not recognized as a valid command.

The BACKGROUND command will issue an ACK pulse when the CPU changes from normal to background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO command will issue an ACK pulse when the CPU exits from background mode. The ACK pulse related to this command could be aborted using the SYNC command.

The GO_UNTIL command is equivalent to a GO command with exception that the ACK pulse, in this case, is issued when the CPU enters into background mode. This command is an alternative to the GO command and should be used when the host wants to trace if a breakpoint match occurs and causes the CPU to enter active background mode. Note that the ACK is issued whenever the CPU enters BDM, which could be caused by a breakpoint match or by a BGND instruction being executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TRACE1 command has the related ACK pulse issued when the CPU enters background active mode after one instruction of the application program is executed. The ACK pulse related to this command could be aborted using the SYNC command.

The TAGGO command will not issue an ACK pulse because this would interfere with the tagging function shared on the same pin.

Name[1]		Bit 7	6	5	4	3	2	1	Bit 0
0x0022 DBGTBH	R	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8
	W								
0x0023 DBGTBL	R	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
	W								
0x0024 DBGCNT	R	TBF	0	CNT					
	W								
0x0025 DBGCCX[(2)]	R	PAGSEL		EXTCMP					
	W								
0x0026 DBGCCH[2]	R	Bit 15	14	13	12	11	10	9	Bit 8
	W								
0x0027 DBGCCL[2]	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x0028 DBGC2 BKPCT0	R	BKABEN	FULL	BDM	TAGAB	BKCEN	TAGC	RWCEN	RWC
	W								
0x0029 DBGC3 BKPCT1	R	BKAMBH	BKAMBL	BKBMBH	BKBMBL	RWAEN	RWA	RWBEN	RWB
	W								
0x002A DBGCAX BKP0X	R	PAGSEL		EXTCMP					
	W								
0x002B DBGCAH BKP0H	R	Bit 15	14	13	12	11	10	9	Bit 8
	W								
0x002C DBGCAL BKP0L	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								
0x002D DBGCBX BKP1X	R	PAGSEL		EXTCMP					
	W								
0x002E DBGCBH BKP1H	R	Bit 15	14	13	12	11	10	9	Bit 8
	W								
0x002F DBGCBL BKP1L	R	Bit 7	6	5	4	3	2	1	Bit 0
	W								

= Unimplemented or Reserved

Figure 7-3. DBG Register Summary (continued)

7.4.2.6.3 Detail Mode

In the detail mode, address and data for all cycles except program fetch (P) and free (f) cycles are stored in trace buffer. 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 his code was in error.

7.4.2.6.4 Profile Mode

This mode is intended to allow a host computer to poll a running target and provide a histogram of program execution. Each read of the trace buffer address will return the address of the last instruction executed. The DBGCNT register is not incremented and the trace buffer does not get filled. The ARM bit is not used and all breakpoints and all other debug functions will be disabled.

7.4.2.7 Storage Memory

The storage memory is a 64 words deep by 16-bits wide dual port RAM array. The CPU accesses the RAM array through a single memory location window (DBGTBH:DBGTBL). The DBG module stores trace information in the RAM array in a circular buffer format. As data is read via the CPU, a pointer into the RAM will increment so that the next CPU read will receive fresh information. In all trigger modes except for event-only and detail capture mode, the data stored in the trace buffer will be change-of-flow addresses. change-of-flow addresses are defined as follows:

- Source address of conditional branches (long, short, BRSET, and loop constructs) taken
- Destination address of indexed JMP, JSR, and CALL instruction
- Destination address of RTI, RTS, and RTC instructions
- Vector address of interrupts except for SWI and BDM vectors

In the event-only trigger modes only the 16-bit data bus value corresponding to the event is stored. In the detail capture mode, address and then data are stored for all cycles except program fetch (P) and free (f) cycles.

7.4.2.8 Storing Data in Memory Storage Buffer

7.4.2.8.1 Storing with Begin-Trigger

Storing with begin-trigger can be used in all trigger modes. When DBG mode is enabled and armed in the begin-trigger mode, data is not stored in the trace buffer until the trigger condition is met. As soon as the trigger condition is met, the DBG module will remain armed until 64 words are stored in the trace buffer. If the trigger is at the address of the change-of-flow instruction the change-of-flow associated with the trigger event will be stored in the trace buffer.

7.4.2.8.2 Storing with End-Trigger

Storing with end-trigger cannot be used in event-only trigger modes. When DBG mode is enabled and armed in the end-trigger mode, data is stored in the trace buffer until the trigger condition is met. When the trigger condition is met, the DBG module will become de-armed and no more data will be stored. If

Table 9-12. Outcome of Clock Loss in Pseudo-Stop Mode

CME	SCME	SCMIE	CRG Actions
0	X	X	Clock failure --> No action, clock loss not detected.
1	0	X	Clock failure --> CRG performs Clock Monitor Reset immediately
1	1	0	Clock Monitor failure --> Scenario 1: OSCCLK **recovers** prior to exiting Pseudo-Stop Mode. – MCU remains in Pseudo-Stop Mode, – VREG enabled, – PLL enabled, – SCM activated, – Start Clock Quality Check, – Set SCMIF interrupt flag. *Some time later OSCCLK recovers.* – CM no longer indicates a failure, – 4096 OSCCLK cycles later Clock Quality Check indicates clock o.k., – SCM deactivated, – PLL disabled, – VREG disabled. – MCU remains in Pseudo-Stop Mode. *Some time later either a wakeup interrupt occurs (no SCM interrupt)* – Exit Pseudo-Stop Mode using OSCCLK as system clock (SYSCLK), – Continue normal operation. *or an External Reset is applied.* – Exit Pseudo-Stop Mode using OSCCLK as system clock, – Start reset sequence. Scenario 2: OSCCLK **does not recover** prior to exiting Pseudo-Stop Mode. – MCU remains in Pseudo-Stop Mode, – VREG enabled, – PLL enabled, – SCM activated, – Start Clock Quality Check, – Set SCMIF interrupt flag, – Keep performing Clock Quality Checks (could continue infinitely) while in Pseudo-Stop Mode. *Some time later either a wakeup interrupt occurs (no SCM interrupt)* – Exit Pseudo-Stop Mode in SCM using PLL clock (f_{SCM}) as system clock – Continue to perform additional Clock Quality Checks until OSCCLK is o.k. again. or an External RESET is applied. – Exit Pseudo-Stop Mode in SCM using PLL clock (f_{SCM}) as system clock – Start reset sequence, – Continue to perform additional Clock Quality Checks until OSCCLK is o.k.again.

The MSCAN is able to leave sleep mode (wake up) only when:

- CAN bus activity occurs and WUPE = 1

  or
- the CPU clears the SLPRQ bit

NOTE

The CPU cannot clear the SLPRQ bit before sleep mode (SLPRQ = 1 and SLPAK = 1) is active.

After wake-up, the MSCAN waits for 11 consecutive recessive bits to synchronize to the CAN bus. As a consequence, if the MSCAN is woken-up by a CAN frame, this frame is not received.

The receive message buffers (RxFG and RxBG) contain messages if they were received before sleep mode was entered. All pending actions will be executed upon wake-up; copying of RxBG into RxFG, message aborts and message transmissions. If the MSCAN remains in bus-off state after sleep mode was exited, it continues counting the 128 occurrences of 11 consecutive recessive bits.

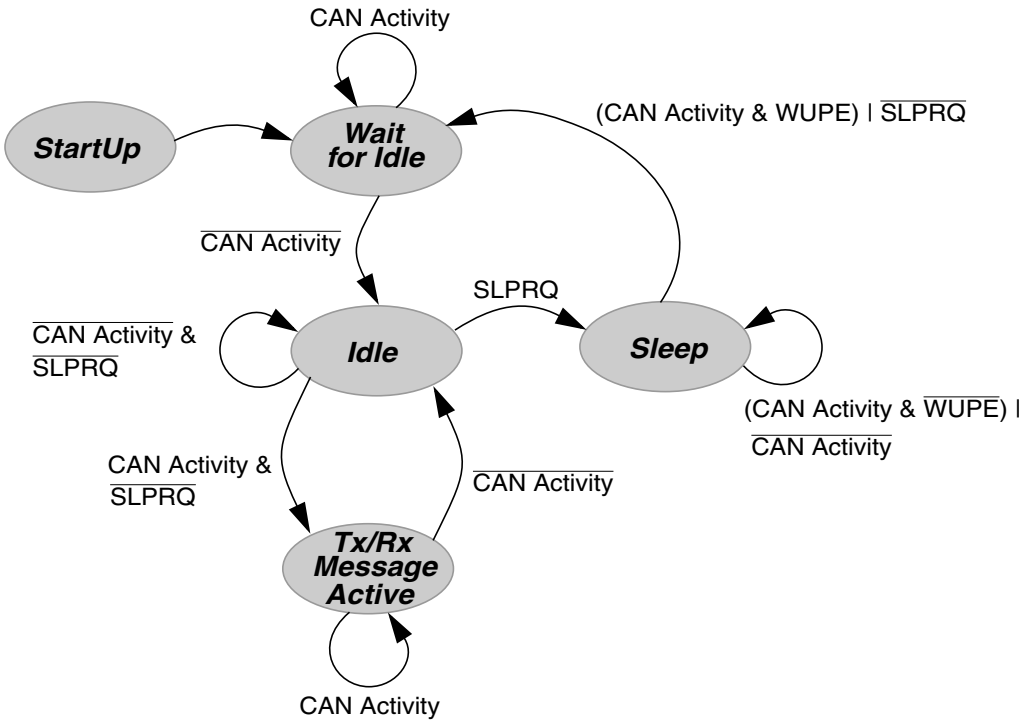


Figure 10-45. Simplified State Transitions for Entering/Leaving Sleep Mode

12.2.4 PWM2 — Pulse Width Modulator Channel 2 Pin

This pin serves as waveform output of PWM channel 2.

12.2.5 PWM1 — Pulse Width Modulator Channel 1 Pin

This pin serves as waveform output of PWM channel 1.

12.2.6 PWM0 — Pulse Width Modulator Channel 0 Pin

This pin serves as waveform output of PWM channel 0.

12.3 Memory Map and Register Definition

This subsection describes in detail all the registers and register bits in the PWM8B6CV1 module.

The special-purpose registers and register bit functions that would not normally be made available to device end users, such as factory test control registers and reserved registers are clearly identified by means of shading the appropriate portions of address maps and register diagrams. Notes explaining the reasons for restricting access to the registers and functions are also explained in the individual register descriptions.

12.3.1 Module Memory Map

The following paragraphs describe the content of the registers in the PWM8B6CV1 module. The base address of the PWM8B6CV1 module is determined at the MCU level when the MCU is defined. The register decode map is fixed and begins at the first address of the module address offset. Table 12-1 shows the registers associated with the PWM and their relative offset from the base address. The register detail description follows the order in which they appear in the register map.

Reserved bits within a register will always read as 0 and the write will be unimplemented. Unimplemented functions are indicated by shading the bit.

Table 12-1 shows the memory map for the PWM8B6CV1 module.

NOTE

Register address = base address + address offset, where the base address is defined at the MCU level and the address offset is defined at the module level.

Module Base + 0x0005

	7	6	5	4	3	2	1	0
R	0	CON45	CON23	CON01	PSWAI	PFRZ	0	0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 12-8. PWM Control Register (PWMCTL)

Read: anytime

Write: anytime

There are three control bits for concatenation, each of which is used to concatenate a pair of PWM channels into one 16-bit channel. When channels 4 and 5 are concatenated, channel 4 registers become the high-order bytes of the double-byte channel. When channels 2 and 3 are concatenated, channel 2 registers become the high-order bytes of the double-byte channel. When channels 0 and 1 are concatenated, channel 0 registers become the high-order bytes of the double-byte channel.

Reference Section 12.4.2.7, “PWM 16-Bit Functions,” for a more detailed description of the concatenation PWM function.

NOTE

Change these bits only when both corresponding channels are disabled.

Table 12-12 is used to summarize which channels are used to set the various control bits when in 16-bit mode.

Table 12-12. 16-bit Concatenation Mode Summary

CONxx	PWMEx	PPOLx	PCLKx	CAEx	PWMx Output
CON45	PWME5	PPOL5	PCLK5	CAE5	PWM5
CON23	PWME3	PPOL3	PCLK3	CAE3	PWM3
CON01	PWME1	PPOL1	PCLK1	CAE1	PWM1

12.4.2.8 PWM Boundary Cases

Table 12-13 summarizes the boundary conditions for the PWM regardless of the output mode (left aligned or center aligned) and 8-bit (normal) or 16-bit (concatenation):

Table 12-13. PWM Boundary Cases

PWMDTYx	PWMPERx	PPOLx	PWMx Output
0x0000 (indicates no duty)	>0x0000	1	Always Low
0x0000 (indicates no duty)	>0x0000	0	Always High
XX	0x0000(1) (indicates no period)	1	Always High
XX	0x0000[1] (indicates no period)	0	Always Low
>= PWMPERx	XX	1	Always High
>= PWMPERx	XX	0	Always Low

1. Counter = 0x0000 and does not count.

12.5 Resets

The reset state of each individual bit is listed within the register description section (see Section 12.3, “Memory Map and Register Definition,” which details the registers and their bit-fields. All special functions or modes which are initialized during or just following reset are described within this section.

- The 8-bit up/down counter is configured as an up counter out of reset.
- All the channels are disabled and all the counters don’t count.

12.6 Interrupts

The PWM8B6CV1 module has only one interrupt which is generated at the time of emergency shutdown, if the corresponding enable bit (PWMIE) is set. This bit is the enable for the interrupt. The interrupt flag PWMIF is set whenever the input level of the PWM5 channel changes while PWM5ENA=1 or when PWMENA is being asserted while the level at PWM5 is active.

A description of the registers involved and affected due to this interrupt is explained in Section 12.3.2.15, “PWM Shutdown Register (PWMSDN).”

Write: Anytime

Table 15-3. TIOS Field Descriptions

Field	Description
7:0 IOS[7:0]	**Input Capture or Output Compare Channel Configuration** 0 The corresponding channel acts as an input capture. 1 The corresponding channel acts as an output compare.

15.3.2.2 Timer Compare Force Register (CFORC)

Module Base + 0x0001

	7	6	5	4	3	2	1	0
R	0	0	0	0	0	0	0	0
W	FOC7	FOC6	FOC5	FOC4	FOC3	FOC2	FOC1	FOC0
Reset	0	0	0	0	0	0	0	0

Figure 15-7. Timer Compare Force Register (CFORC)

Read: Anytime but will always return 0x0000 (1 state is transient)

Write: Anytime

Table 15-4. CFORC Field Descriptions

Field	Description
7:0 FOC[7:0]	**Force Output Compare Action for Channel 7:0** — A write to this register with the corresponding data bit(s) set causes the action which is programmed for output compare “x” to occur immediately. The action taken is the same as if a successful comparison had just taken place with the TCx register except the interrupt flag does not get set. **Note:** A successful channel 7 output compare overrides any channel 6:0 compares. If forced output compare on any channel occurs at the same time as the successful output compare then forced output compare action will take precedence and interrupt flag won’t get set.

15.3.2.3 Output Compare 7 Mask Register (OC7M)

Module Base + 0x0002

	7	6	5	4	3	2	1	0
R W	OC7M7	OC7M6	OC7M5	OC7M4	OC7M3	OC7M2	OC7M1	OC7M0
Reset	0	0	0	0	0	0	0	0

Figure 15-8. Output Compare 7 Mask Register (OC7M)

Read: Anytime

Write: Anytime

addresses sequentially staring with 0xFF00-0xFF01 and ending with 0xFF06–0xFF07. The values 0x0000 and 0xFFFF are not permitted as keys. When the KEYACC bit is set, reads of the Flash array will return invalid data.

The user code stored in the Flash array must have a method of receiving the backdoor key from an external stimulus. This external stimulus would typically be through one of the on-chip serial ports.

If KEYEN[1:0] = 1:0 in the FSEC register, the MCU can be unsecured by the backdoor key access sequence described below:

1. Set the KEYACC bit in the FCNFG register
2. Write the correct four 16-bit words to Flash addresses 0xFF00–0xFF07 sequentially starting with 0xFF00
3. Clear the KEYACC bit in the FCNFG register
4. If all four 16-bit words match the backdoor key stored in Flash addresses 0xFF00–0xFF07, the MCU is unsecured and bits SEC[1:0] in the FSEC register are forced to the unsecure state of 1:0

The backdoor key access sequence is monitored by the internal security state machine. An illegal operation during the backdoor key access sequence will cause the security state machine to lock, leaving the MCU in the secured state. A reset of the MCU will cause the security state machine to exit the lock state and allow a new backdoor key access sequence to be attempted. The following illegal operations will lock the security state machine:

1. If any of the four 16-bit words does not match the backdoor key programmed in the Flash array
2. If the four 16-bit words are written in the wrong sequence
3. If more than four 16-bit words are written
4. If any of the four 16-bit words written are 0x0000 or 0xFFFF
5. If the KEYACC bit does not remain set while the four 16-bit words are written

After the backdoor key access sequence has been correctly matched, the MCU will be unsecured. The Flash security byte can be programmed to the unsecure state, if desired.

In the unsecure state, the user has full control of the contents of the four word backdoor key by programming bytes 0xFF00–0xFF07 of the Flash configuration field.

The security as defined in the Flash security/options byte at address 0xFF0F is not changed by using the backdoor key access sequence to unsecure. The backdoor key stored in addresses 0xFF00–0xFF07 is unaffected by the backdoor key access sequence. After the next reset sequence, the security state of the Flash module is determined by the Flash security/options byte at address 0xFF0F. The backdoor key access sequence has no effect on the program and erase protection defined in the FPROT register.

It is not possible to unsecure the MCU in special single chip mode by executing the backdoor key access sequence in background debug mode.

18.4.1.3 Valid Flash Commands

Table 18-16 summarizes the valid Flash commands along with the effects of the commands on the Flash array.

Table 18-16. Valid Flash Commands

FCMD	Meaning	Function on Flash Array
0x05	Erase Verify	Verify all bytes in the Flash array are erased. If the Flash array is erased, the BLANK bit will set in the FSTAT register upon command completion.
0x20	Program	Program a word (2 bytes) in the Flash array.
0x40	Sector Erase	Erase all 512 bytes in a sector of the Flash array.
0x41	Mass Erase	Erase all bytes in the Flash array. A mass erase of the full Flash array is only possible when FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register are set prior to launching the command.

CAUTION

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

Chapter 19
64 Kbyte Flash Module (S12FTS64KV4)

19.1 Introduction

The FTS128K1FTS64K module implements a 12864 Kbyte Flash (nonvolatile) memory. The Flash memory contains one array of 12864 Kbytes organized as 1024512 rows of 128128 bytes with an erase sector size of eight rows (10241024 bytes). The Flash array may be read as either bytes, aligned words, or misaligned words. Read access time is one bus cycle for byte and aligned word, and two bus cycles for misaligned words.

The Flash array is ideal for program and data storage for single-supply applications allowing for field reprogramming without requiring external voltage sources for program or erase. Program and erase functions are controlled by a command driven interface. The Flash module supports both mass erase and sector erase. An erased bit reads 1 and a programmed bit reads 0. The high voltage required to program and erase is generated internally. It is not possible to read from a Flash array while it is being erased or programmed.

CAUTION

A Flash word must be in the erased state before being programmed. Cumulative programming of bits within a Flash word is not allowed.

19.1.1 Glossary

Command Write Sequence — A three-step MCU instruction sequence to program, erase, or erase verify the Flash array memory.

19.1.2 Features

- 12864 Kbytes of Flash memory comprised of one 12864 Kbyte array divided into 12864 sectors of 10241024 bytes
- Automated program and erase algorithm
- Interrupts on Flash command completion and command buffer empty
- Fast sector erase and word program operation
- 2-stage command pipeline for faster multi-word program times
- Flexible protection scheme to prevent accidental program or erase
- Single power supply for Flash program and erase operations
- Security feature to prevent unauthorized access to the Flash array memory

Table 20-5. FSEC Field Descriptions

Field	Description
7–6 KEYEN[1:0]	**Backdoor Key Security Enable Bits** — The KEYEN[1:0] bits define the enabling of the backdoor key access to the Flash module as shown in Table 20-6.
5–2 NV[5:2]	**Nonvolatile Flag Bits** — The NV[5:2] bits are available to the user as nonvolatile flags.
1–0 SEC[1:0]	**Flash Security Bits** — The SEC[1:0] bits define the security state of the MCU as shown in Table 20-7. If the Flash module is unsecured using backdoor key access, the SEC[1:0] bits are forced to 1:0.

Table 20-6. Flash KEYEN States

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

1. Preferred KEYEN state to disable Backdoor Key Access.

Table 20-7. Flash Security States

SEC[1:0]	Status of Security
00	Secured
01(1)	Secured
10	Unsecured
11	Secured

1. Preferred SEC state to set MCU to secured state.

The security function in the Flash module is described in Section 20.4.3, “Flash Module Security”.

20.3.2.3 RESERVED1

This register is reserved for factory testing and is not accessible to the user.

Module Base + 0x0002

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

= Unimplemented or Reserved

Figure 20-8. RESERVED1

All bits read 0 and are not writable.

20.4.1.3.2 Program Command

The program operation will program a previously erased word in the Flash array using an embedded algorithm.

An example flow to execute the program operation is shown in Figure 20-25. The program command write sequence is as follows:

1. Write to a Flash array address to start the command write sequence for the program command. The data written will be programmed to the Flash array address written.
2. Write the program command, 0x20, to the FCMD register.
3. Clear the CBEIF flag in the FSTAT register by writing a 1 to CBEIF to launch the program command.

If a word to be programmed is in a protected area of the Flash array, the PVIOL flag in the FSTAT register will set and the program command will not launch. Once the program command has successfully launched, the CCIF flag in the FSTAT register will set after the program operation has completed unless a new command write sequence has been buffered. By executing a new program command write sequence on sequential words after the CBEIF flag in the FSTAT register has been set, up to 55% faster programming time per word can be effectively achieved than by waiting for the CCIF flag to set after each program operation.