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	32MHz
Connectivity	CANbus, EBI/EMI, I²C, IrDA, LINbus, SCI, SPI
Peripherals	LCD, Motor control PWM, POR, PWM, WDT
Number of I/O	80
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	4K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	100-LQFP
Supplier Device Package	100-LQFP (14x14)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/s9s12hy32j0mll

Chapter 1
Device Overview MC9S12HY/HA-Family

Chapter 2
Port Integration Module (S12HYPIMV1)

Chapter 3S12P Memory Map Control (S12PMMCV1)

Chapter 4
Interrupt Module (S12SINTV1)

3.6 Initialization/Application Information

3.6.1 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

This sequence is uninterruptable. There is no need to inhibit interrupts during the CALL instruction execution. A CALL instruction can be performed from any address to any other address in the local CPU memory space.

The PPAGE value supplied by the instruction is part of the effective address of the CPU. For all addressing mode variations (except indexed-indirect modes) the new page value is provided by an immediate operand in the instruction. In indexed-indirect variations of the CALL instruction a pointer specifies memory locations where the new page value and the address of the called subroutine are stored. Using indirect addressing for both the new page value and the address within the page allows usage of values calculated at run time rather than immediate values that must be known at the time of assembly.

The RTC instruction terminates subroutines invoked by a CALL instruction. The RTC instruction unstacks the PPAGE value and the return address and refills the queue. Execution resumes with the next instruction after the CALL instruction.

During the execution of an RTC instruction the CPU performs the following steps:

1. Pulls the previously stored PPAGE value from the stack
2. Pulls the 16-bit return address from the stack and loads it into the PC
3. Writes the PPAGE value into the PPAGE register
4. Refills the queue and resumes execution at the return address

This sequence is uninterruptable. The RTC can be executed from anywhere in the local CPU memory space.

The CALL and RTC instructions behave like JSR and RTS instruction, they however require more execution cycles. Usage of JSR/RTS instructions is therefore recommended when possible and

Since the host knows the target serial clock frequency, the SYNC command (used to abort a command) does not need to consider the lower possible target frequency. In this case, the host could issue a SYNC very close to the 128 serial clock cycles length. Providing a small overhead on the pulse length in order to assure the SYNC pulse will not be misinterpreted by the target. See Section 5.4.9, “SYNC — Request Timed Reference Pulse”.

Figure 5-12 shows a SYNC command being issued after a READ_BYTE, which aborts the READ_BYTE command. Note that, after the command is aborted a new command could be issued by the host computer.

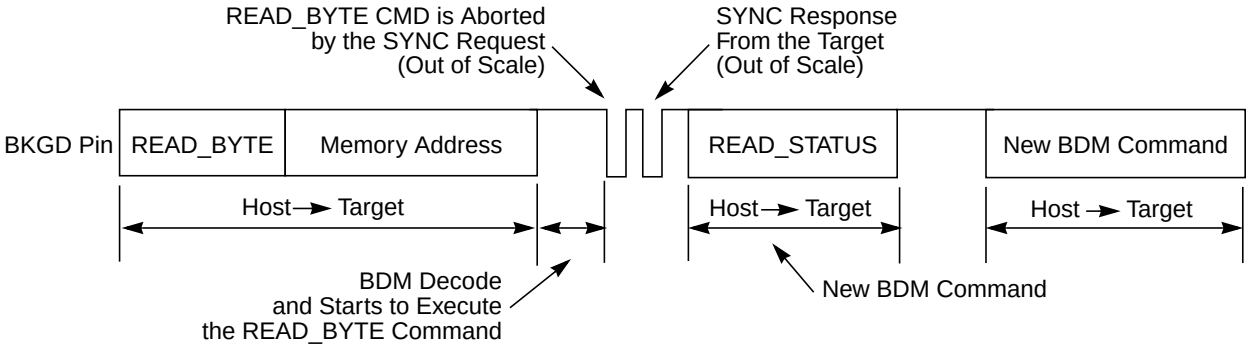


Figure 5-12. ACK Abort Procedure at the Command Level

NOTE

Figure 5-12 does not represent the signals in a true timing scale

Figure 5-13 shows a conflict between the ACK pulse and the SYNC request pulse. This conflict could occur if a POD device is connected to the target BKGD pin and the target is already in debug active mode. Consider that the target CPU is executing a pending BDM command at the exact moment the POD is being connected to the BKGD pin. In this case, an ACK pulse is issued along with the SYNC command. In this case, there is an electrical conflict between the ACK speedup pulse and the SYNC pulse. Since this is not a probable situation, the protocol does not prevent this conflict from happening.

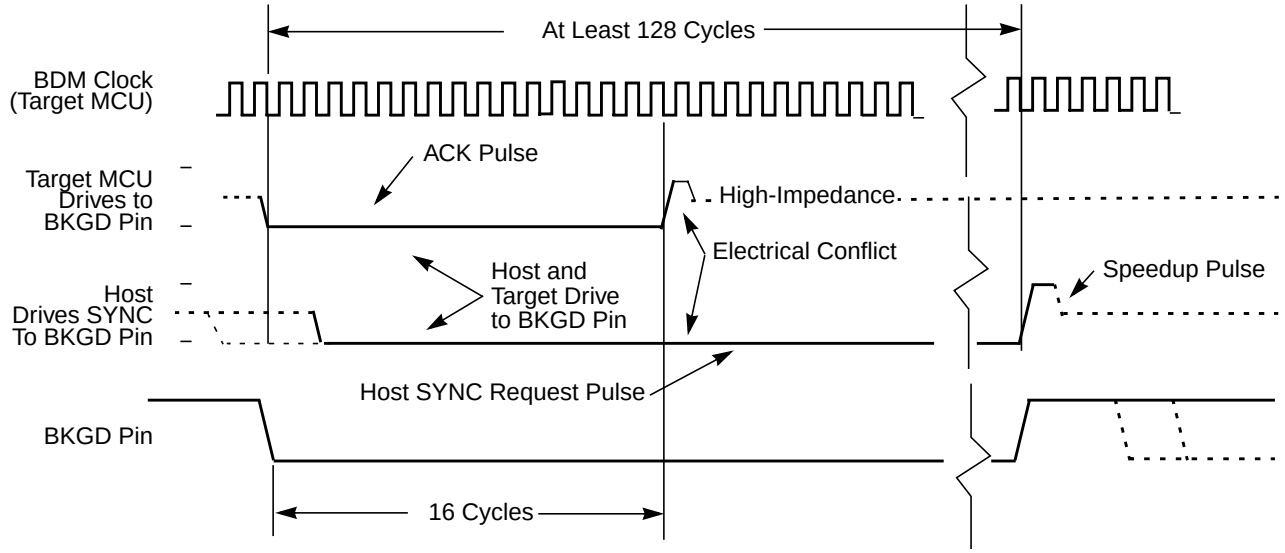


Figure 5-13. ACK Pulse and SYNC Request Conflict

6.3.2.6 Debug Count Register (DBGCNT)

Address: 0x0026

	7	6	5	4	3	2	1	0
R	TBF	0	CNT					
W								
Reset	—	—	—	—	—	—	—	—
POR	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 6-8. Debug Count Register (DBGCNT)

Read: Anytime

Write: Never

Table 6-12. DBGCNT Field Descriptions

Field	Description
7 TBF	**Trace Buffer Full** — The TBF bit indicates that the trace buffer has stored 64 or more lines of data since it was last armed. If this bit is set, then all 64 lines will be valid data, regardless of the value of DBGCNT bits. The TBF bit is cleared when ARM in DBGC1 is written to a one. The TBF is cleared by the power on reset initialization. Other system generated resets have no affect on this bit This bit is also visible at DBGSR[7]
5–0 CNT[5:0]	**Count Value** — The CNT bits indicate the number of valid data 20-bit data lines stored in the Trace Buffer. Table 6-13 shows the correlation between the CNT bits and the number of valid data lines in the Trace Buffer. When the CNT rolls over to zero, the TBF bit in DBGSR is set and incrementing of CNT will continue in end-trigger mode. The DBGCNT register is cleared when ARM in DBGC1 is written to a one. The DBGCNT register is cleared by power-on-reset initialization but is not cleared by other system resets. Thus should a reset occur during a debug session, the DBGCNT register still indicates after the reset, the number of valid trace buffer entries stored before the reset occurred. The DBGCNT register is not decremented when reading from the trace buffer.

Table 6-13. CNT Decoding Table

TBF	CNT[5:0]	Description
0	000000	No data valid
0	000001 000010 000100 000110 .. 111111	1 line valid 2 lines valid 4 lines valid 6 lines valid .. 63 lines valid
1	000000	64 lines valid; if using Begin trigger alignment, ARM bit will be cleared and the tracing session ends.
1	000001 111110	64 lines valid, oldest data has been overwritten by most recent data

Similarly the SZE and SZ bits allow the size of access (word or byte) to be considered in the compare. Only comparators A and B feature SZE and SZ.

The TAG bit in each comparator control register is used to determine the match condition. By setting TAG, the comparator qualifies a match with the output of opcode tracking logic and a state sequencer transition occurs when the tagged instruction reaches the CPU execution stage. Whilst tagging the RW, RWE, SZE, and SZ bits and the comparator data registers are ignored; the comparator address register must be loaded with the exact opcode address.

If the TAG bit is clear (forced type match) a comparator match is generated when the selected address appears on the system address bus. If the selected address is an opcode address, the match is generated when the opcode is fetched from the memory, which precedes the instruction execution by an indefinite number of cycles due to instruction pipelining. For a comparator match of an opcode at an odd address when TAG = 0, the corresponding even address must be contained in the comparator register. Thus for an opcode at odd address (n), the comparator register must contain address (n–1).

Once a successful comparator match has occurred, the condition that caused the original match is not verified again on subsequent matches. Thus if a particular data value is verified at a given address, this address may not still contain that data value when a subsequent match occurs.

Match[0, 1, 2] map directly to Comparators [A, B, C] respectively, except in range modes (see 6.3.2.4, "Debug Control Register2 (DBGC2)"). Comparator channel priority rules are described in the priority section (6.4.3.4, "Channel Priorities").

6.4.2.1 Single Address Comparator Match

With range comparisons disabled, the match condition is an exact equivalence of address bus with the value stored in the comparator address registers. Further qualification of the type of access (R/W, word/byte) and databus contents is possible, depending on comparator channel.

6.4.2.1.1 Comparator C

Comparator C offers only address and direction (R/W) comparison. The exact address is compared, thus with the comparator address register loaded with address (n) a word access of address (n–1) also accesses (n) but does not cause a match.

Table 6-32. Comparator C Access Considerations

Condition For Valid Match	Comp C Address	RWE	RW	Examples
Read and write accesses of ADDR[n]	ADDR[n][1]	0	X	LDAA ADDR[n] STAA #$BYTE ADDR[n]
Write accesses of ADDR[n]	ADDR[n]	1	0	STAA #$BYTE ADDR[n]
Read accesses of ADDR[n]	ADDR[n]	1	1	LDAA #$BYTE ADDR[n]

1 A word access of ADDR[n-1] also accesses ADDR[n] but does not generate a match. The comparator address register must contain the exact address from the code.

6.4.3.4 Channel Priorities

In case of simultaneous matches the priority is resolved according to Table 6-36. The lower priority is suppressed. It is thus possible to miss a lower priority match if it occurs simultaneously with a higher priority. The priorities described in Table 6-36 dictate that in the case of simultaneous matches, the match pointing to final state has highest priority followed by the lower channel number (0,1,2).

Table 6-36. Channel Priorities

Priority	Source	Action
Highest	**TRIG**	Enter Final State
	Channel pointing to Final State	Transition to next state as defined by state control registers
	Match0 (force or tag hit)	Transition to next state as defined by state control registers
	Match1 (force or tag hit)	Transition to next state as defined by state control registers
Lowest	**Match2 (force or tag hit)**	Transition to next state as defined by state control registers

6.4.4 State Sequence Control

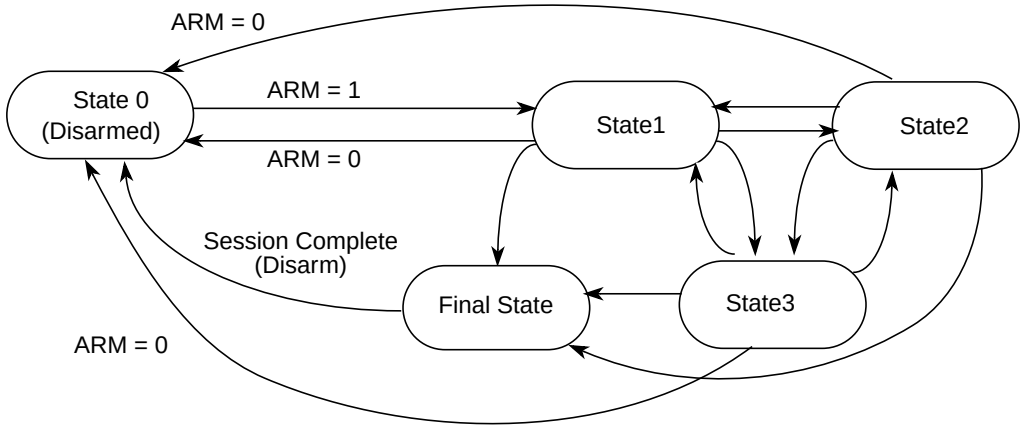


Figure 6-24. State Sequencer Diagram

The state sequencer allows a defined sequence of events to provide a trigger point for tracing of data in the trace buffer. Once the DBG module has been armed by setting the ARM bit in the DBGC1 register, then state1 of the state sequencer is entered. Further transitions between the states are then controlled by the state control registers and channel matches. From Final State the only permitted transition is back to the disarmed state0. Transition between any of the states 1 to 3 is not restricted. Each transition updates the SSF[2:0] flags in DBGSR accordingly to indicate the current state.

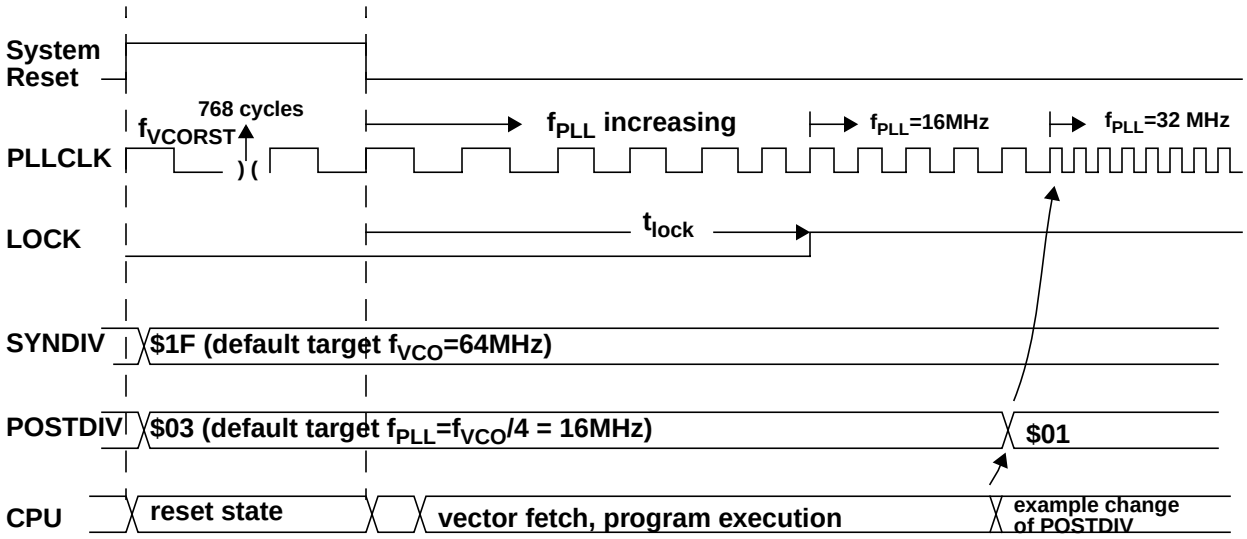
Alternatively writing to the TRIG bit in DBGSC1, provides an immediate trigger independent of comparator matches.

Independent of the state sequencer, each comparator channel can be individually configured to generate an immediate breakpoint when a match occurs through the use of the BRK bits in the DBGxCTL registers. Thus it is possible to generate an immediate breakpoint on selected channels, whilst a state sequencer transition can be initiated by a match on other channels. If a debug session is ended by a match on a channel the state sequencer transitions through Final State for a clock cycle to state0. This is independent of tracing

7.4.2 Startup from Reset

An example of startup of clock system from Reset is given in Figure 7-31.

Figure 7-31. Startup of clock system after Reset



7.4.3 Stop Mode using PLLCLK as Bus Clock

An example of what happens going into Stop Mode and exiting Stop Mode after an interrupt is shown in Figure 7-32. Disable PLL Lock interrupt (LOCKIE=0) before going into Stop Mode.

Figure 7-32. Stop Mode using PLLCLK as Bus Clock

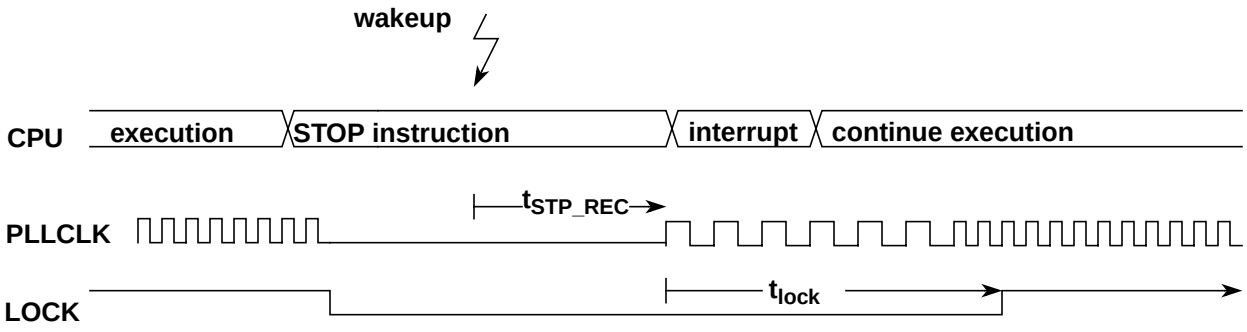


Table 8-21 shows how depending on the A/D resolution the conversion result is transferred to the ATD result registers. Compare is always done using all 12 bits of both the conversion result and the compare value in ATDDRn.

Table 8-21. Conversion result mapping to ATDDRn

A/D resolution	DJM	conversion result mapping to ATDDR*n*
8-bit data	0	Bit[11:4] = result, Bit[3:0]=0000
8-bit data	1	Bit[7:0] = result, Bit[11:8]=0000
10-bit data	0	Bit[11:2] = result, Bit[1:0]=00
10-bit data	1	Bit[9:0] = result, Bit[11:10]=00
12-bit data	X	Bit[11:0] = result

Figure 9-24. Receive/Transmit Message Buffer — Extended Identifier Mapping

Register Name		Bit 7	6	5	4	3	2	1	Bit0
0x00X0 IDR0	R	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21
	W								
0x00X1 IDR1	R	ID20	ID19	ID18	SRR (=1)	IDE (=1)	ID17	ID16	ID15
	W								
0x00X2 IDR2	R	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7
	W								
0x00X3 IDR3	R	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
	W								
0x00X4 DSR0	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X5 DSR1	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X6 DSR2	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X7 DSR3	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X8 DSR4	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00X9 DSR5	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00XA DSR6	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00XB DSR7	R	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
	W								
0x00XC DLR	R					DLC3	DLC2	DLC1	DLC0
	W								

14.3.2.8 Timer Control Register 1/Timer Control Register 2 (TCTL1/TCTL2)

Module Base + 0x0008

	7	6	5	4	3	2	1	0
R W	OM7	OL7	OM6	OL6	OM5	OL5	OM4	OL4
Reset	0	0	0	0	0	0	0	0

Figure 14-14. Timer Control Register 1 (TCTL1)

Module Base + 0x0009

	7	6	5	4	3	2	1	0
R W	OM3	OL3	OM2	OL2	OM1	OL1	OM0	OL0
Reset	0	0	0	0	0	0	0	0

Figure 14-15. Timer Control Register 2 (TCTL2)

Read: Anytime

Write: Anytime

Table 14-8. TCTL1/TCTL2 Field Descriptions

Field	Description
7:0 OMx	**Output Mode** — These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is 1, the pin associated with OCx becomes an output tied to OCx. **Note:** To enable output action by OMx bits on timer port, the corresponding bit in OC7M should be cleared. For an output line to be driven by an OCx the OCPDx must be cleared.
7:0 OLx	**Output Level** — These eight pairs of control bits are encoded to specify the output action to be taken as a result of a successful OCx compare. When either OMx or OLx is 1, the pin associated with OCx becomes an output tied to OCx. **Note:** To enable output action by OLx bits on timer port, the corresponding bit in OC7M should be cleared. For an output line to be driven by an OCx the OCPDx must be cleared.

Table 14-9. Compare Result Output Action

OMx	OLx	Action
0	0	No output compare action on the timer output signal
0	1	Toggle OCx output line
1	0	Clear OCx output line to zero
1	1	Set OCx output line to one

Table 17-28. P-Flash Commands

FCMD	Command	Function on P-Flash Memory
0x04	Read Once	Read a dedicated 64 byte field in the nonvolatile information register in P-Flash block that was previously programmed using the Program Once command.
0x06	Program P-Flash	Program a phrase in a P-Flash block.
0x07	Program Once	Program a dedicated 64 byte field in the nonvolatile information register in P-Flash block that is allowed to be programmed only once.
0x08	Erase All Blocks	Erase all P-Flash (and D-Flash) blocks. An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the DPOPEN bit in the DFPROT register are set prior to launching the command.
0x09	Erase Flash Block	Erase a P-Flash (or D-Flash) block. An erase of the full P-Flash block is only possible when FPLDIS, FPHDIS and FPOPEN bits in the FPROT register are set prior to launching the command.
0x0A	Erase P-Flash Sector	Erase all bytes in a P-Flash sector.
0x0B	Unsecure Flash	Supports a method of releasing MCU security by erasing all P-Flash (and D-Flash) blocks and verifying that all P-Flash (and D-Flash) blocks are erased.
0x0C	Verify Backdoor Access Key	Supports a method of releasing MCU security by verifying a set of security keys.
0x0D	Set User Margin Level	Specifies a user margin read level for all P-Flash blocks.
0x0E	Set Field Margin Level	Specifies a field margin read level for all P-Flash blocks (special modes only).

17.4.3.5 D-Flash Commands

Table 17-29 summarizes the valid D-Flash commands along with the effects of the commands on the D-Flash block.

Table 17-29. D-Flash Commands

FCMD	Command	Function on D-Flash Memory
0x01	Erase Verify All Blocks	Verify that all D-Flash (and P-Flash) blocks are erased.
0x02	Erase Verify Block	Verify that the D-Flash block is erased.
0x08	Erase All Blocks	Erase all D-Flash (and P-Flash) blocks. An erase of all Flash blocks is only possible when the FPLDIS, FPHDIS, and FPOPEN bits in the FPROT register and the DPOPEN bit in the DFPROT register are set prior to launching the command.
0x09	Erase Flash Block	Erase a D-Flash (or P-Flash) block. An erase of the full D-Flash block is only possible when DPOPEN bit in the DFPROT register is set prior to launching the command.
0x0B	Unsecure Flash	Supports a method of releasing MCU security by erasing all D-Flash (and P-Flash) blocks and verifying that all D-Flash (and P-Flash) blocks are erased.
0x0D	Set User Margin Level	Specifies a user margin read level for the D-Flash block.
0x0E	Set Field Margin Level	Specifies a field margin read level for the D-Flash block (special modes only).

Table 17-29. D-Flash Commands

FCMD	Command	Function on D-Flash Memory
0x10	Erase Verify D-Flash Section	Verify that a given number of words starting at the address provided are erased.
0x11	Program D-Flash	Program up to four words in the D-Flash block.
0x12	Erase D-Flash Sector	Erase all bytes in a sector of the D-Flash block.

17.4.4 Allowed Simultaneous P-Flash and D-Flash Operations

Only the operations marked ‘OK’ in Table 17-30 are permitted to be run simultaneously on the Program Flash and Data Flash blocks. Some operations cannot be executed simultaneously because certain hardware resources are shared by the two memories. The priority has been placed on permitting Program Flash reads while program and erase operations execute on the Data Flash, providing read (P-Flash) while write (D-Flash) functionality.

Table 17-30. Allowed P-Flash and D-Flash Simultaneous Operations

	Data Flash				
Program Flash	**Read**	**Margin Read[1]**	**Program**	**Sector Erase**	**Mass Erase[3]**
Read		OK	OK	OK	
Margin Read[1]		OK[2]			
Program					
Sector Erase				OK	
Mass Erase[3]					OK

1 A ‘Margin Read’ is any read after executing the margin setting commands ‘Set User Margin Level’ or ‘Set Field Margin Level’ with anything but the ‘normal’ level specified.

2 See the Note on margin settings in Section 17.4.5.12 and Section 17.4.5.13.

3 The ‘Mass Erase’ operations are commands ‘Erase All Blocks’ and ‘Erase Flash Block’

17.4.5 Flash Command Description

This section provides details of all available Flash commands launched by a command write sequence. The ACCERR bit in the FSTAT register will be set during the command write sequence if any of the following illegal steps are performed, causing the command not to be processed by the Memory Controller:

- Starting any command write sequence that programs or erases Flash memory before initializing the FCLKDIV register
- Writing an invalid command as part of the command write sequence
- For additional possible errors, refer to the error handling table provided for each command

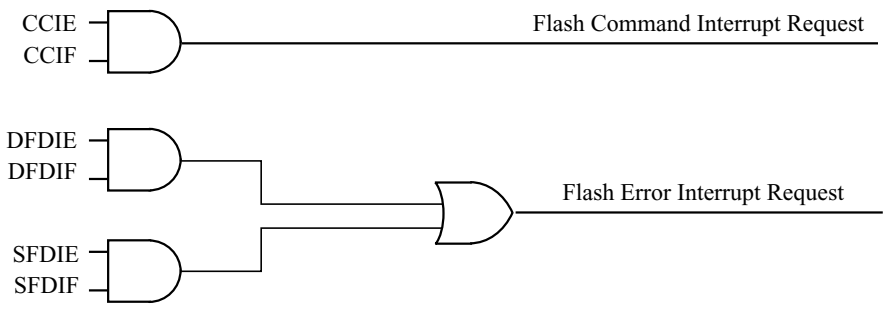


Figure 17-27. Flash Module Interrupts Implementation

17.4.7 Wait Mode

The Flash module is not affected if the MCU enters wait mode. The Flash module can recover the MCU from wait via the CCIF interrupt (see Section 17.4.6, “Interrupts”).

17.4.8 Stop Mode

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

17.5 Security

The Flash module provides security information to the MCU. The Flash security state is defined by the SEC bits of the FSEC register (see Table 17-10). During reset, the Flash module initializes the FSEC register using data read from the security byte of the Flash configuration field at global address 0x3_FF0F. The security state out of reset can be permanently changed by programming the security byte assuming that the MCU is starting from a mode where the necessary P-Flash erase and program commands are available and that the upper region of the P-Flash is unprotected. If the Flash security byte is successfully programmed, its new value will take affect after the next MCU reset.

The following subsections describe these security-related subjects:

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

17.5.1 Unsecuring the MCU using Backdoor Key Access

The MCU may be unsecured by using the backdoor key access feature which requires knowledge of the contents of the backdoor keys (four 16-bit words programmed at addresses 0x3_FF00-0x3_FF07). If the KEYEN[1:0] bits are in the enabled state (see Section 17.3.2.2), the Verify Backdoor Access Key command (see Section 17.4.5.11) allows the user to present four prospective keys for comparison to the keys stored in the Flash memory via the Memory Controller. If the keys presented in the Verify Backdoor Access Key command match the backdoor keys stored in the Flash memory, the SEC bits in the FSEC

Offset Module Base + 0x0020 . . . 0x002F — Access: User read/write

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	S	D8	D7	D6	D5	D4	D3	D2	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

= Unimplemented or Reserved

Figure 19-9. Motor Controller Duty Cycle Register x (MCDCx) with FAST = 1

Table 19-10. MCDCx Field Descriptions

Field	Description
0 S	SIGN — The SIGN bit is used to define which output will drive the PWM signal in (dual) full-H-bridge modes. The SIGN bit has no effect in half-bridge modes. See Section 19.4.1.3.2, “Sign Bit (S)”, and table Table 19-12 for detailed information about the impact of RECIRC and SIGN bit on the PWM output.

Whenever FAST = 1, the bits D10, D9, D1, and D0 will be set to 0 if the duty cycle register is written.

For example setting MCDCx = 0x0158 with FAST = 0 gives the same output waveform as setting MCDCx = 0x5600 with FAST = 1 (with FAST = 1, the low byte of MCDCx needs not to be written).

The state of the FAST bit has impact only during write and read operations. A change of the FAST bit (set or clear) without writing a new value does not impact the internal interpretation of the duty cycle values.

To prevent the output from inconsistent signals, the duty cycle registers are double buffered. The motor controller module will use working registers to generate the output signals. The working registers are copied from the bus accessible registers at the following conditions:

- MCPER is set to 0 (all channels are disabled in this case)
- MCAM[1:0] of the respective channel is set to 0 (channel is disabled)
- A PWM timer counter overflow occurs while in half H-bridge or full H-bridge mode
- A PWM channel pair is configured to work in Dual Full H-Bridge mode and a PWM timer counter overflow occurs after the odd[1] duty cycle register of the channel pair has been written.

In this way, the output of the PWM will always be either the old PWM waveform or the new PWM waveform, not some variation in between.

Reads of this register return the most recent value written. Reads do not necessarily return the value of the currently active sign, duty cycle, and dither functionality due to the double buffering scheme.

1. Odd duty cycle register: MCDCx+1, x = 2·n

19.4.1.3.5 Dither Bit (DITH)

The purpose of the dither mode is to increase the minimum length of output pulses without decreasing the PWM resolution, in order to limit the pulse distortion introduced by the slew rate control of the outputs. If dither mode is selected the output pattern will repeat after two timer counter overflows. For the same output frequency, the shortest output pulse will have twice the length while dither feature is selected. To achieve the same output frame frequency, the prescaler of the MC10B8C module has to be set to twice the division rate if dither mode is selected; e.g., with the same prescaler division rate the repeat rate of the output pattern is the same as well as the shortest output pulse with or without dither mode selected.

The DITH bit in control register 0 enables or disables the dither function.

DITH = 0: dither function is disabled.

When DITH is cleared and assuming left aligned operation and RECIRC = 0, the PWM output will start at a logic low level at the beginning of the PWM period (motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 11-bit PWM duty cycle value, DUTY, contained in D[10:0] in MCDCx. When a match (output compare between motor controller timer counter and DUTY) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the contents of MCPER – 1). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes one PWM period. The PWM period repeats every P counts (as defined by the bits P[10:0] in the motor controller period register) of the motor controller timer counter. If DUTY >= P, the output will be static low. If DUTY = 0x0000, the output will be continuously at a logic high level. The relationship between the motor controller timer counter clock, motor controller timer counter value, and PWM output while DITH = 0 is shown in Figure 19-17.

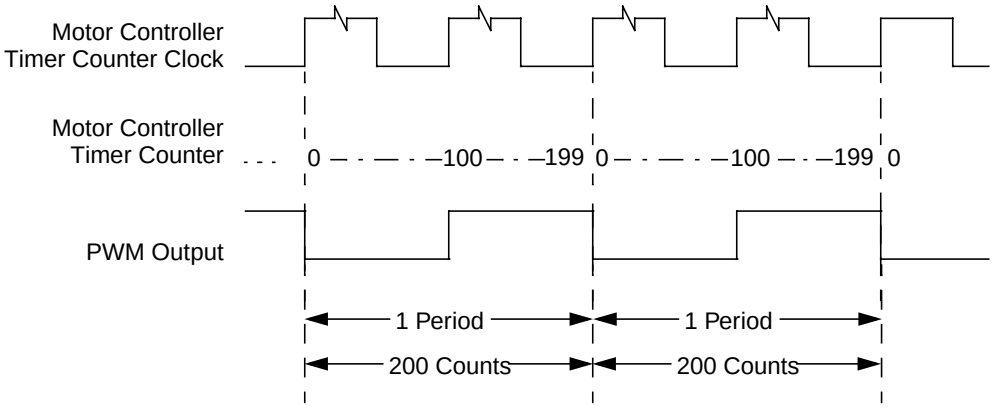


Figure 19-17. PWM Output: DITH = 0, MCAM[1:0] = 01, MCDC = 100, MCPER = 200, RECIRC = 0

DITH = 1: dither function is enabled

Please note if DITH = 1, the bit P0 in the motor controller period register will be internally forced to 0 and read always as 0.

When DITH is set and assuming left aligned operation and RECIRC = 0, the PWM output will start at a logic low level at the beginning of the PWM period (when the motor controller timer counter = 0x000). The PWM output remains low until the motor controller timer counter matches the 10-bit PWM duty cycle

value, DUTY, contained in D[10:1] in MCDCx. When a match (output compare between motor controller timer counter and DUTY) occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by P[10:1] – 1 in MCPER). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level. This completes the first half of the PWM period. During the second half of the PWM period, the PWM output will remain at a logic low level until either the motor controller timer counter matches the 10-bit PWM duty cycle value, DUTY, contained in D[10:1] in MCDCx if D0 = 0, or the motor controller timer counter matches the 10-bit PWM duty cycle value + 1 (the value of D[10:1] in MCDCx is increment by 1 and is compared with the motor controller timer counter value) if D0 = 1 in the corresponding duty cycle register. When a match occurs, the PWM output will toggle to a logic high level and will remain at a logic high level until the motor controller timer counter overflows (reaches the value defined by P[10:1] – 1 in MCPER). After the motor controller timer counter resets to 0x000, the PWM output will return to a logic low level.

This process will repeat every number of counts of the motor controller timer counter defined by the period register contents (P[10:0]). If the output is neither set to 0% nor to 100% there will be four edges on the PWM output per PWM period in this case. Therefore, the PWM output compare function will alternate between DUTY and DUTY + 1 every half PWM period if D0 in the corresponding duty cycle register is set to 1. The relationship between the motor controller timer counter clock (f_{TC}), motor controller timer counter value, and left aligned PWM output if DITH = 1 is shown in Figure 19-18 and Figure 19-19. Figure 19-20 and Figure 19-21 show right aligned and center aligned PWM operation respectively, with dither feature enabled and D0 = 1. Please note: In the following examples, the MCPER value is defined by the bits P[10:0], which is, if DITH = 1, always an even number.

NOTE

The DITH bit must be changed only if the motor controller is disabled (all channels disabled or period register cleared) to avoid erroneous waveforms.

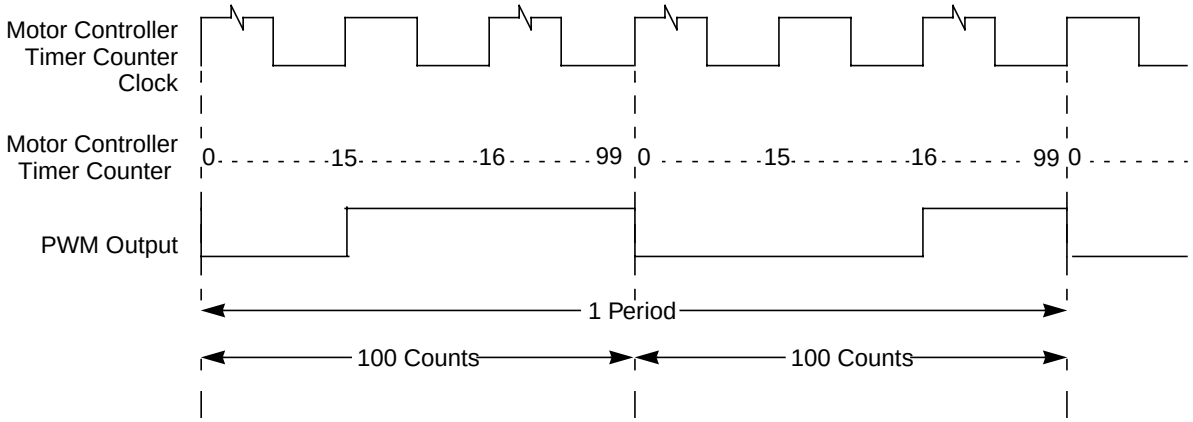


Figure 19-18. PWM Output: DITH = 1, MCAM[1:0] = 01, MCDC = 31, MCPER = 200, RECIRC = 0

Table A-6. 5-V I/O Characteristics

18	D	Port T, S, R, AD interrupt input pulse passed ($\overline{STOP}$)	t_{PULSE}	4	—	—	tcyc
19	D	$\overline{IRQ}$ pulse width, edge-sensitive mode ($\overline{STOP}$)	PW_{IRQ}	1	—	—	tcyc

1. Maximum leakage current occurs at maximum operating temperature. Current decreases by approximately one-half for each 8 C to 12°C in the temperature range from 50°C to 125°C.
2. Refer to Section A.1.4, “Current Injection” for more details
3. Parameter only applies in stop or pseudo stop mode.

A.1.10 Supply Currents

This section describes the current consumption characteristics of the device as well as the conditions for the measurements.

A.1.10.1 Measurement Conditions

IDD value is measured on VDDR pin. It does not include the current to drive external loads. Unless otherwise noted the currents are measured in special single chip mode and the CPU code is executed from RAM. For Run and Wait current measurements PLL is on and the reference clock is the IRC trimmed to 1 MHz. The bus frequency is 32 MHz and the CPU frequency is 64 MHz. Table A-7, Table A-8 and Table A-9 show the configuration of the CPMU module and the peripherals for Run, Wait and Stop current measurement.

Table A-7. CPMU Configuration for Pseudo Stop Current Measurement

CPMU REGISTER	Bit settings/Conditions
CPMUCLKS	PLLSEL=0, PSTP=1, PRE=PCE=RTIOSCSEL=COPOSCSEL=1
CPMUOSC	OSCE=1, External Square wave on EXTAL f_{EXTAL}=16MHz, V_{IH}= 1.8V, V_{IL}=0V
CPMURTI	RTDEC=0, RTR[6:4]=111, RTR[3:0]=1111;
CPMUCOP	WCOP=1, CR[2:0]=111

Table A-8. CPUM Configuration for Run/Wait and Full Stop Current Measurement

CPMU REGISTER	Bit settings/Conditions
CPMUSYNR	VCOFRQ[1:0]=01,SYNDIV[5:0] = 32
CPMUPOSTDIV	POSTDIV[4:0]=0,
CPMUCLKS	PLLSEL=1
CPMUOSC	OSCE=0, Reference clock for PLL is f_{ref}=f_{irc1m} trimmed to 1MHz

Appendix B
Ordering Information

The following figure provides an ordering partnumber example for the devices covered by this data book. There are two options when ordering a device. Customers must choose between ordering either the mask-specific partnumber or the generic / mask-independent partnumber. Ordering the mask-specific partnumber enables the customer to specify which particular maskset they will receive whereas ordering the generic maskset means that FSL will ship the currently preferred maskset (which may change over time).

In either case, the marking on the device will always show the generic / mask-independent partnumber and the mask set number.

NOTE

The mask identifier suffix and the Tape & Reel suffix are always both omitted from the partnumber which is actually marked on the device.

For specific partnumbers to order, please contact your local sales office. The below figure illustrates the structure of a typical mask-specific ordering number for the MC9S12HY/HA-Family devices

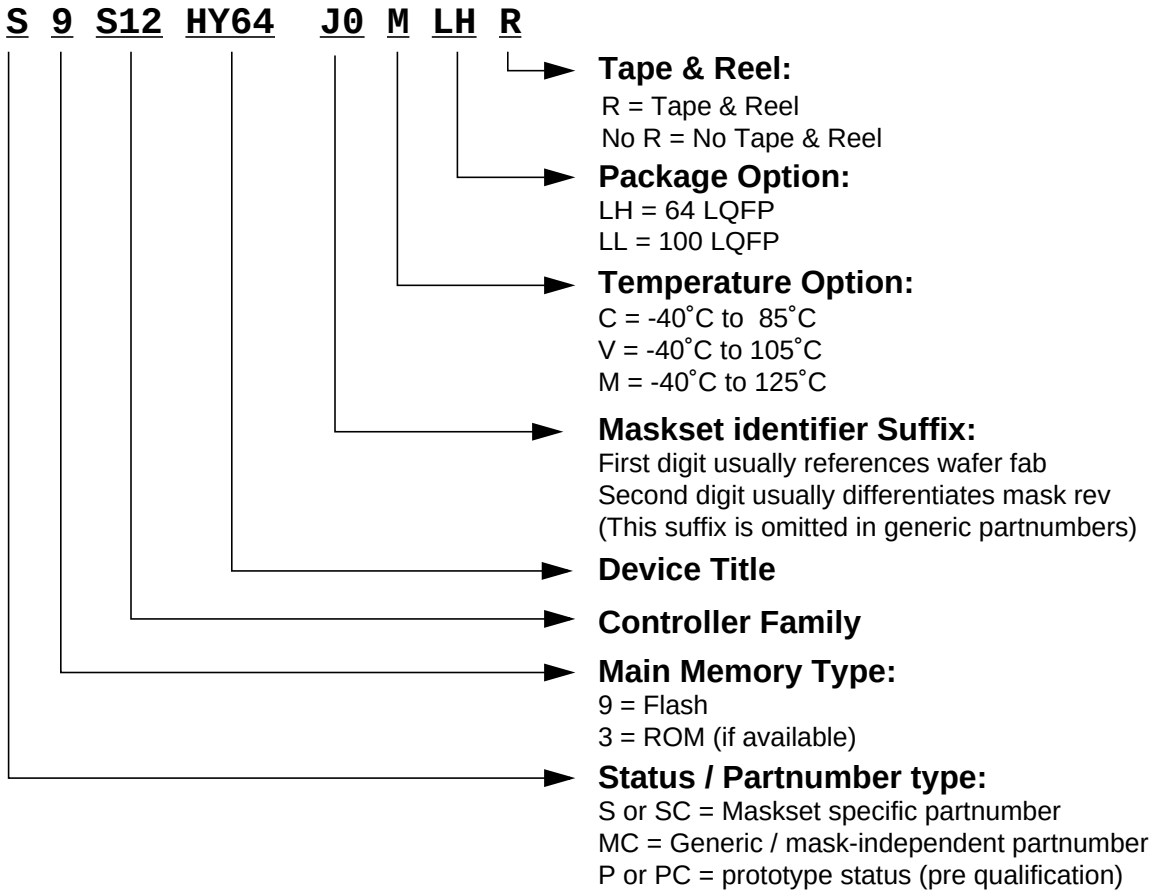


Figure B-1. Order Part Number Example

Figure C-3. 100-pin LQFP (case no. 983) - page 3