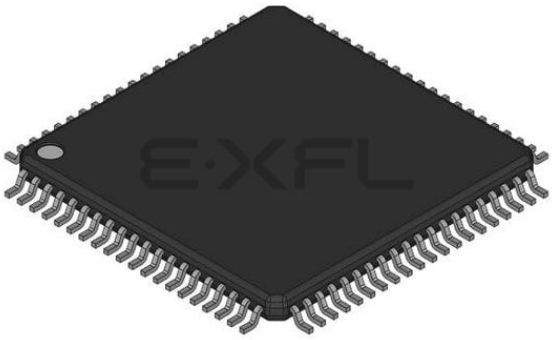


Welcome to E-XFL.COM

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	CPU12
Core Size	16-Bit
Speed	8MHz
Connectivity	CANbus, MI Bus, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	48
Program Memory Size	60KB (60K x 8)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	4.5V ~ 5.5V
Data Converters	A/D 8x8/10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	80-QFP
Supplier Device Package	80-QFP (14x14)
Purchase URL	https://www.e-xfl.com/pro/item?MUrl=&PartUrl=mc912d60amfue8

List of Figures

Section 1. General Description

1.1 Contents

1.2 Introduction

The MC68HC912D60A microcontroller unit (MCU) is a 16-bit device available in two package options, 80-pin QFP and 112-pin TQFP. On-chip peripherals include a 16-bit central processing unit (CPU12), 60K bytes of flash EEPROM, 2K bytes of RAM, 1K bytes of EEPROM, two asynchronous serial communication interfaces (SCI), a serial peripheral interface (SPI), an enhanced capture timer (ECT), two (one on 80QFP) 8-channel,10-bit analog-to-digital converters (ATD), a four-channel pulse-width modulator (PWM), and a CAN 2.0 A, B software compatible module (MSCAN12). System resource mapping, clock generation, interrupt control and bus interfacing are managed by the lite integration module (LIM). The MC68HC912D60A has full 16-bit data paths throughout, however, the external bus can operate in an 8-bit narrow mode so single 8-bit wide memory can be interfaced for lower cost systems. The inclusion of a PLL circuit allows power consumption and performance to be adjusted to suit operational requirements. In addition to the I/O ports available in each module, 16 (2 on 80QFP) I/O port pins are available with Key-Wake-Up capability from STOP or WAIT mode.

2.6 Indexed Addressing Modes

The CPU12 indexed modes reduce execution time and eliminate code size penalties for using the Y index register. CPU12 indexed addressing uses a postbyte plus zero, one, or two extension bytes after the instruction opcode. The postbyte and extensions do the following tasks:

- Specify which index register is used.
- Determine whether a value in an accumulator is used as an offset.
- Enable automatic pre- or post-increment or decrement
- Specify use of 5-bit, 9-bit, or 16-bit signed offsets.

Table 2-2. Summary of Indexed Operations

Postbyte Code (xb)	Source Code Syntax	Comments
rr0nnnnn	,r n,r –n,r	**5-bit constant offset** n = –16 to +15 rr can specify X, Y, SP, or PC
111rr0zs	n,r –n,r	**Constant offset** (9- or 16-bit signed) z-0 = 9-bit with sign in LSB of postbyte(s) 1 = 16-bit if z = s = 1, 16-bit offset indexed-indirect (see below) rr can specify X, Y, SP, or PC
111rr011	[n,r]	**16-bit offset indexed-indirect** rr can specify X, Y, SP, or PC
rr1pnnnn	n,–r n,+r n,r– n,r+	**Auto pre-decrement/increment** or **Auto post-decrement/increment**; p = pre-(0) or post-(1), n = –8 to –1, +1 to +8 rr can specify X, Y, or SP (PC not a valid choice)
111rr1aa	A,r B,r D,r	**Accumulator offset** (unsigned 8-bit or 16-bit) aa-00 = A 01 = B 10 = D (16-bit) 11 = see accumulator D offset indexed-indirect rr can specify X, Y, SP, or PC
111rr111	[D,r]	**Accumulator D offset indexed-indirect** rr can specify X, Y, SP, or PC

	Bit 7	6	5	4	3	2	1	Bit 0
	PUPH	PUPG	0	PUPE	0	0	PUPB	PUPA
RESET:	1	1	0	1	0	0	0	0

PUCR — Pull-Up Control Register **$000C**

These bits select pull-up resistors for any pin in the corresponding port that is currently configured as an input. This register is not in the map in peripheral mode.

Read and write anytime.

PUPH — Pull-Up or Pull-Down Port H Enable

0 = Port H pull-ups are disabled.
1 = Enable pull-up/down devices for all port H input pins.

PUPG — Pull-Up or Pull-Down Port G Enable

0 = Port G pull-ups are disabled.
1 = Enable pull-up/down devices for all port G input pins.

PUPE — Pull-Up Port E Enable

0 = Port E pull-ups on PE7 and PE[3:0] are disabled.
1 = Enable pull-up devices for port E input pins PE7 and PE[3:0].

PUPB — Pull-Up Port B Enable

0 = Port B pull-ups are disabled.
1 = Enable pull-up devices for all port B input pins.

This bit has no effect if port B is being used as part of the address/data bus (the pull-ups are inactive).

PUPA — Pull-Up Port A Enable

0 = Port A pull-ups are disabled.
1 = Enable pull-up devices for all port A input pins.

This bit has no effect if port A is being used as part of the address/data bus (the pull-ups are inactive).

maskable. The remaining sources are maskable, and any one of them can be given priority over other maskable interrupts.

The priorities of the non-maskable sources are:

1. POR or $\overline{\text{RESET}}$ pin
2. Clock monitor reset
3. COP watchdog reset
4. Unimplemented instruction trap
5. Software interrupt instruction (SWI)
6. $\overline{\text{XIRQ}}$ signal (if X bit in CCR = 0)

9.3 Maskable interrupts

Maskable interrupt sources include on-chip peripheral systems and external interrupt service requests. Interrupts from these sources are recognized when the global interrupt mask bit (I) in the CCR is cleared. The default state of the I bit out of reset is one, but it can be written at any time.

Interrupt sources are prioritized by default but any one maskable interrupt source may be assigned the highest priority by means of the HPRIO register. The relative priorities of the other sources remain the same.

An interrupt that is assigned highest priority is still subject to global masking by the I bit in the CCR, or by any associated local bits. Interrupt vectors are not affected by priority assignment. HPRIO can only be written while the I bit is set (interrupts inhibited). Table 9-1 lists interrupt sources and vectors in default order of priority.

Table 9-1. Interrupt Vector Map

Vector Address	Interrupt Source	CCR Mask	Local Enable	HPRIO Value to Elevate
$FFFE, $FFFF	Reset	None	None	–
$FFFC, $FFFD	Clock monitor fail reset	None	COPCTL (CME, FCME)	–
$FFFA, $FFFB	COP failure reset	None	COP rate selected	–
$FFF8, $FFF9	Unimplemented instruction trap	None	None	–
$FFF6, $FFF7	SWI	None	None	–
$FFF4, $FFF5	XIRQ	X bit	None	–
$FFF2, $FFF3	IRQ	I bit	INTCR (IRQEN)	$F2
$FFF0, $FFF1	Real time interrupt	I bit	RTICTL (RTIE)	$F0
$FFEE, $FFEF	Timer channel 0	I bit	TMSK1 (C0I)	$EE
$FFEC, $FFED	Timer channel 1	I bit	TMSK1 (C1I)	$EC
$FFEA, $FFEB	Timer channel 2	I bit	TMSK1 (C2I)	$EA
$FFE8, $FFE9	Timer channel 3	I bit	TMSK1 (C3I)	$E8
$FFE6, $FFE7	Timer channel 4	I bit	TMSK1 (C4I)	$E6
$FFE4, $FFE5	Timer channel 5	I bit	TMSK1 (C5I)	$E4
$FFE2, $FFE3	Timer channel 6	I bit	TMSK1 (C6I)	$E2
$FFE0, $FFE1	Timer channel 7	I bit	TMSK1 (C7I)	$E0
$FFDE, $FFDF	Timer overflow	I bit	TMSK2 (TOI)	$DE
$FFDC, $FFDD	Pulse accumulator overflow	I bit	PACTL (PAOVI)	$DC
$FFDA, $FFDB	Pulse accumulator input edge	I bit	PACTL (PAI)	$DA
$FFD8, $FFD9	SPI serial transfer complete	I bit	SP0CR1 (SPIE)	$D8
$FFD6, $FFD7	SCI 0	I bit	SC0CR2 (TIE, TCIE, RIE, ILIE)	$D6
$FFD4, $FFD5	SCI 1	I bit	SC1CR2 (TIE, TCIE, RIE, ILIE)	$D4
$FFD2, $FFD3	ATD0 or ATD1	I bit	ATDxCTL2 (ASCIE)	$D2
$FFD0, $FFD1	MSCAN wake-up	I bit	CRIER (WUPIE)	$D0
$FFCE, $FFCF	Key wake-up G or H	I bit	KWIEG[6:0] and KWIEH[7:0]	$CE
$FFCC, $FFCD	Modulus down counter underflow	I bit	MCCTL (MCZI)	$CC
$FFCA, $FFCB	Pulse Accumulator B Overflow	I bit	PBCTL (PBOVI)	$CA
$FFC8, $FFC9	MSCAN errors	I bit	CRIER (RWRNIE, TWRNIE, RERRIE, TERRIE, BOFFIE, OVRIE)	$C8
$FFC6, $FFC7	MSCAN receive	I bit	CRIER (RXFIE)	$C6
$FFC4, $FFC5	MSCAN transmit	I bit	CTCR (TXEIE[2:0])	$C4
$FFC2, $FFC3	CGM lock and limp home	I bit	PLLCR (LOCKIE, LHIE)	$C2
$FF80–$FFC1	Reserved	I bit		$80–$C0

During this power up sequence, after the POR pulse falling edge, the VCO supplies the limp-home clock frequency to the 13-stage counter, as the BCSP output is forced high and MCS is forced low. XCLK, BCLK and MCLK are forced to be PCLK, which is supplied by the VCO at f_{VCOMIN}. The initial period taken for the 13-stage counter to reach 4096 defines the internal reset period.

If the clock monitor indicates the presence of an external clock during the internal reset period, limp-home mode is de-asserted and the 13-stage counter is then driven by EXTALi clock. After the 13-stage counter reaches a count of 4096 XCLK cycles, the internal reset is released, the 13-stage counter is reset and the MCU exits reset normally using EXTALi clock.

However, if the crystal start-up time is longer than the initial count of 4096 XCLK cycles, or in the absence of an external clock, the MCU will leave the reset state in limp-home mode. The LHOME flag is set and LHIF limp-home interrupt request is set, to indicate it is not operating at the desired frequency. Then after yet another 4096 XCLK cycles followed regularly by 8192 XCLK cycles (corresponding to the 13-stage counter timing out), a check of the clock monitor status is performed. When the presence of an external clock is detected limp-home mode is exited generating a limp-home interrupt if enabled.

CAUTION: *The clock monitor circuit can be misled by the EXTALi clock into reporting a good signal before it has fully stabilised. Under these conditions improper EXTALi clock cycles can occur on SYSCLK. This may lead to a code runaway. To ensure that this situation does not occur, the external Reset period should be longer than the oscillator stabilisation time - this is an application dependent parameter.*

With the VDDPLL supply voltage at VSS level, the PLL module and hence limp-home mode are disabled, the device will remain effectively in a static state whilst there is no activity on EXTALi. The internal reset period and MCU operation will execute only on EXTALi clock.

NOTE: *The external clock signal must stabilise within the initial 4096 reset counter cycles.*

	Bit 7	6	5	4	3	2	1	Bit 0	
	CME	FCME	FCMCOP	WCOP	DISR	CR2	CR1	CR0	
RESET:	0/1	0	0	0	0	1	1	1	Normal
RESET:	0/1	0	0	0	1	1	1	1	Special

COPCTL — COP Control Register **$0016**

CME — Clock Monitor Enable

Read and write anytime.

If FCME is set, this bit has no meaning nor effect.

0 = Clock monitor is disabled. Slow clocks and stop instruction may be used.

1 = Slow or stopped clocks (including the stop instruction) will cause a clock reset sequence or limp-home mode. See Limp-Home and Fast STOP Recovery modes.

On reset

CME is 1 if VDDPLL is high
CME is 0 if VDDPLL is low.

NOTE: *The VDDPLL-dependent reset operation is not implemented on first pass products.*
In this case the state of CME on reset is 0.

FCME — Force Clock Monitor Enable

Write once in normal modes, anytime in special modes. Read anytime.

In normal modes, when this bit is set, the clock monitor function cannot be disabled until a reset occurs.

0 = Clock monitor follows the state of the CME bit.

1 = Slow or stopped clocks will cause a clock reset sequence or limp-home mode.

See Limp-Home and Fast STOP Recovery modes.

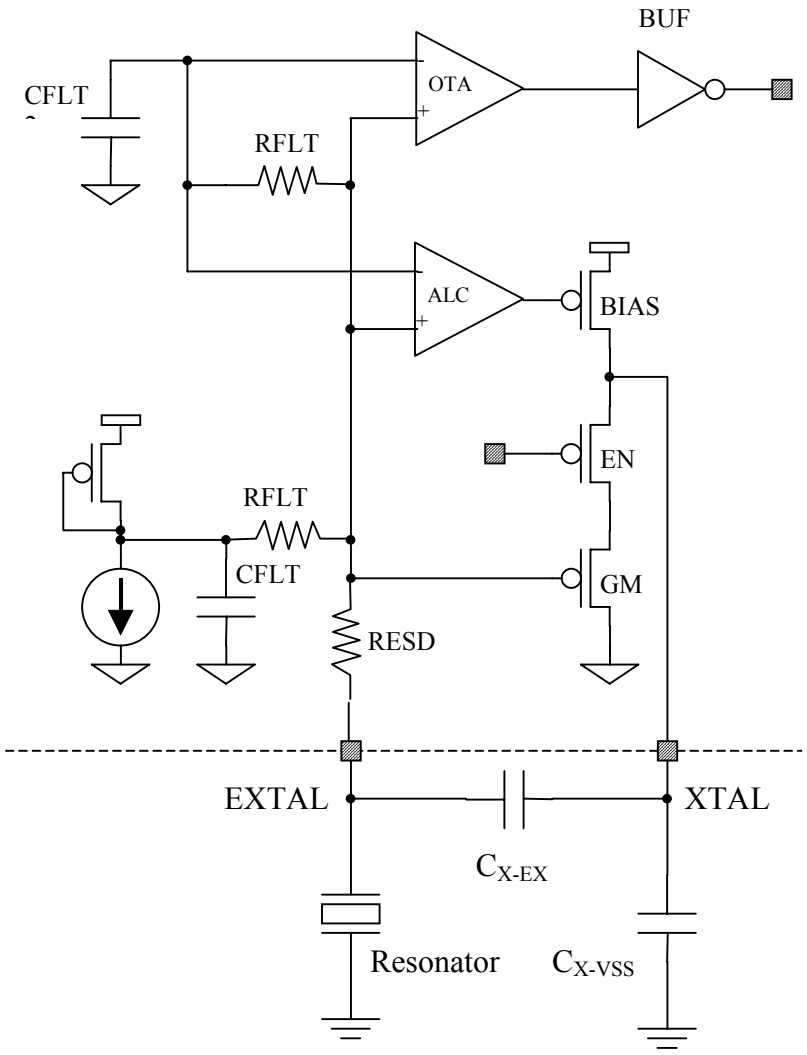


Figure 12-1. MC68HC912D60A Colpitts Oscillator Architecture

12.3.2 MC68HC912D60A Oscillator Design Guidelines

Proper and robust operation of the oscillator circuit requires excellent board layout design practice. Poor layout of the application board can contribute to EMC susceptibility, noise generation, slow starting oscillators, and reaction to noise on the clock input buffer. In addition to published errata for the MC68HC912D60A, the following guidelines must be followed or failure in operation may occur.

	BIT 7	6	5	4	3	2	1	BIT 0
PORT	PT7	PT6	PT5	PT4	PT3	PT2	PT1	PT0
TIMER	I/OC7	I/OC6	I/OC5	I/OC4	I/OC3	I/OC2	I/OC1	I/OC0
RESET:	0	0	0	0	0	0	0	0

PORTT — Timer Port Data Register **$00AE**

Read: any time (inputs return pin level; outputs return data register contents)

Write: data stored in an internal latch (drives pins only if configured for output)

Since the Output Compare 7 shares the pin with Pulse Accumulator input, the only way for Pulse accumulator to receive an independent input from Output Compare 7 is setting both OM7 & OL7 to be zero, and also OC7M7 in OC7M register to be zero.

OC7 is still able to reset the counter if enabled while PT7 is used as input to Pulse Accumulator.

PORTT can be read anytime. When configured as an input, a read will return the pin level. When configured as an output, a read will return the latched output data.

NOTE: *Writes do not change pin state when the pin is configured for timer output. The minimum pulse width for pulse accumulator input should always be greater than the width of two module clocks due to input synchronizer circuitry. The minimum pulse width for the input capture should always be greater than the width of two module clocks due to input synchronizer circuitry.*

	Bit 7	6	5	4	3	2	1	Bit 0
	—	—	RDRF	—	OR	NF	—	—
RESET:	1	1	0	0	0	0	0	0

SC0SR1 — MI Bus Status Register 1 **$00C4**

The bits in these registers are set by various conditions in the MI Bus hardware and are automatically cleared by special acknowledge sequences. The receive related flag bits in SC0SR1 (RDRF, OR and NF) are all cleared by a read of this register followed by a read of the transmit/receive data register low byte. However, only those bits which were set when SC0SR1 was read will be cleared by the subsequent read of the transmit/receive data register low byte.

Read anytime (used in auto clearing mechanism). Write has no meaning or effect.

RDRF — Receive Data Register Full Flag

0 = Contents of the receiver shift register have not been transferred to the receiver data register.

1 = Contents of the receiver serial shift register have been transferred to the receiver data register.

The EOF (end-of-frame) during an MI Bus pull-field is a continuous square wave, which will result in multiple RDRFs. This may be dealt with in any of the following ways:

– By clearing the RIE mask, ignoring unneeded RDRFs, initiating a push field, waiting for TDRE(1) and then clearing the RDRF

– By clearing the RE bit when a pull field is complete, followed by setting the RE bit after the TDRE[1] flag associated with the next push field is asserted.

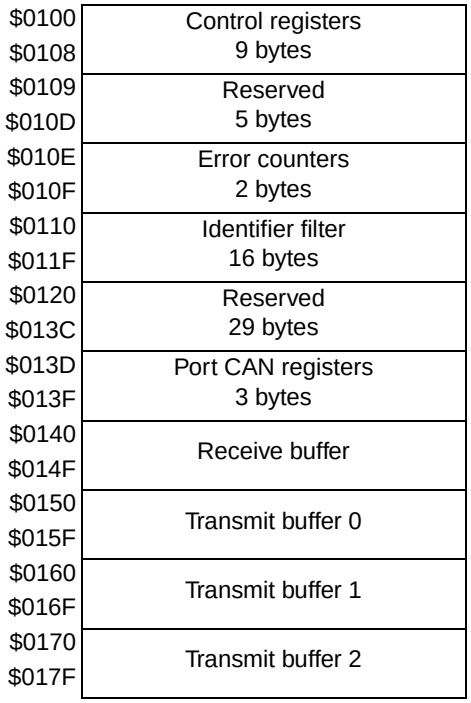
– By disabling the MI Bus.

1. Note that TDRE and TC will both behave in the same way as during normal SCI transmissions. The MI Bus will still be receiving when the TC bit becomes set, hence any queued transmission will not start until the current pull field has finished. See also Register Descriptions.

17.11 Memory Map

The msCAN12 occupies 128 bytes in the CPU12 memory space. The background receive buffer can only be read in test mode.

Figure 17-9. msCAN12 Memory Map



		Bit 7	6	5	4	3	2	1	Bit 0
CIDAR0	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0110	W								
CIDAR1	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0111	W								
CIDAR2	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0112	W								
CIDAR3	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0113	W								
RESET		–	–	–	–	–	–	–	–

		Bit 7	6	5	4	3	2	1	Bit 0
CIDAR4	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0118	W								
CIDAR5	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$0119	W								
CIDAR6	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$011A	W								
CIDAR7	R	AC7	AC6	AC5	AC4	AC3	AC2	AC1	AC0
$011B	W								
RESET		–	–	–	–	–	–	–	–

AC7 – AC0 — Acceptance Code Bits

AC7 – AC0 comprise a user defined sequence of bits with which the corresponding bits of the related identifier register (IDRn) of the receive message buffer are compared. The result of this comparison is then masked with the corresponding identifier mask register.

NOTE: *The CIDAR0–7 registers can only be written if the SFTRES bit in CMCR0 is set.*

17.13.16 msCAN12 Port CAN Data Register (PORTCAN)

		Bit 7	6	5	4	3	2	1	Bit 0
PORTCAN	R	PCAN7	PCAN6	PCAN5	PCAN4	PCAN3	PCAN2	TxCAN	RxCAN
$013E	W								
RESET		0	0	0	0	0	0	0	0

PCAN7 – PCAN2 — Port CAN Data Bits (not available in 80QFP)

Writing to PCANx stores the bit value in an internal bit memory. This value is driven to the respective pin only if DDCANx = 1.

Reading PCANx returns

- the value of the internal bit memory driven to the pin, if DDCANx = 1
- the value of the respective pin, if DDCANx = 0

Reading bits 1 and 0 returns the value of the TxCAN and RxCAN pins, respectively.

17.13.17 msCAN12 Port CAN Data Direction Register (DDRCAN)

DDRCAN register determines the primary direction for the Port CAN pins which are available as general purpose I/O. The value in the DDRCAN also affects the source of data for reads of the corresponding Port CAN register. When the DDCANx = 0 (input), the pin is read. When the DDCANx =1 (output), the internal bit memory is read.

		Bit 7	6	5	4	3	2	1	Bit 0
DDRCAN	R	DDCAN7	DDCAN6	DDCAN5	DDCAN4	DDCAN3	DDCAN2	0	0
$013F	W								
RESET		0	0	0	0	0	0	0	0

DDCAN7 – DDCAN2 — Data Direction Port CAN Bits

0 = Respective I/O pin is configured for input.

1 = Respective I/O pin is configured for output.

ATD0CTL3/ATD1CTL3 — ATD Control Register 3 **$0063/$01E3**

	Bit 7	6	5	4	3	2	1	Bit 0
	0	0	0	0	S1C	FIFO	FRZ1	FRZ0
RESET:	0	0	0	0	0	0	0	0

READ: any time
WRITE: any time

S1C — Conversion Sequence Length (Least Significant Bit)

This control bit works with control bit S8C in ATDCTL5 in determining how many conversion are performed per sequence.

When the S1C bit is set, a sequence length of 1 is defined. However, if the S8C bit is also set, the S8C bit takes precedence. For sequence length coding information see the description for S8C bit in ATDCTL5.

FIFO — Result Register FIFO Mode

0 = Result registers maps to the conversion sequence
1 = Result registers **do not** map to the conversion sequence

In normal operation, the A/D conversion results map into the result registers based on the conversion sequence; the result of the first conversion appears in the first result register, the second result in the second result register, and so on. In FIFO mode the result register counter is not reset at the beginning or ending of a conversion sequence; conversion results are placed in consecutive result registers between sequences. The result register counter wraps around when it reaches the end of the result register file. The conversion counter value in ATDSTAT0 can be used to determine where in the result register file, the next conversion result will be placed.

The results register counter is initialized to zero on three events: on reset, the beginning of a normal (non-FIFO) conversion sequence, and the end of a normal (non-FIFO) conversion sequence. Therefore, the reset bit in register ATDTEST1 can be toggled to zero the result register counter; any sequence allowed to complete normally will zero the result register counter; a new sequence (non-FIFO) initiated with a write to ATDCTL4/5 followed by a write to ATDCTL3 to set the FIFO bit will start a FIFO sequence with the result register initialized.

S8C	S1C	Number of Conversions per Sequence
0	0	4
0	1	1
1	X	8

Table 18-6. Conversion Sequence Length Coding

The result register assignments made to a conversion sequence follow a few simple rules. Normally, the first result is placed in the first register; the second result is placed in the second register, and so on. **Table 18-7** presents the result register assignments for the various conversion lengths that are normally made. If FIFO mode is used, the result register assignments differ. The results are placed in consecutive registers between conversion sequences; the result register mapping wraps around when the end of the register file is reached.

Number of Conversions per Sequence	Result Register Assignment
1	ADR0
4	ADR0 through ADR3
8	ADR0 through ADR7

Table 18-7. Result Register Assignment for Different Conversion Sequences

SCAN — Continuous Conversion Sequence Mode

0 = Perform a conversion sequence and return to idle mode

1 = Perform conversion sequences continuously (scan mode)

The scan mode bit controls whether or not conversion sequences are performed continuously or not. If this control bit is 0, a write to control register 4 or 5 will initiate a conversion sequence; the conversion sequence will be executed; the sequence complete flag (SCF) will be set, and the module will return to idle mode. In this mode, the module remains powered up but no conversions are performed; the module waits for the next conversion sequence to be initiated.

If this control bit is 1, a single conversion sequence initiation will result in a continuously executed conversion sequence. When a conversion sequence completes, the sequence complete flag (SCF) is set and a new sequence is immediately begun. The conversion mode characteristics of each sequence are identical. If a new conversion

18.9.6 ATDTEST Module Test Register (ATDTEST)

The test registers implement various special (test) modes used to test the ATD module. The reset bit in ATDTEST1 is always read/write. The SAR (successive approximation register) can always be read but only written in special (test) mode.

The functions implemented by the test registers are reserved for factory test.

ATD0TESTH/ATD1TESTH — ATD Test Register **$0068/$01E8**

	Bit 7	6	5	4	3	2	1	Bit 0
	SAR9	SAR8	SAR7	SAR6	SAR5	SAR4	SAR3	SAR2
RESET:	0	0	0	0	0	0	0	0

ATD0TESTL/ATD1TESTL — ATD Test Register **$0069/$01E9**

	Bit 7	6	5	4	3	2	1	Bit 0
	SAR1	SAR0	RST	0	0	0	0	0
RESET:	0	0	0	0	0	0	0	0

SAR[9:0] — Successive Approximation Register

This ten bit value represents the contents of the AD machine's successive approximation register. This value can always be read. It can only be written in special (test) mode. Note that ATDTEST0 acts as a ten bit register since the entire SAR is read/written when accessing this address.

RST — Test Mode Reset Bit

0 = No reset

1 = Reset the ATD module

When set, this bit causes the ATD module to reset itself. This sets all registers to their reset state (note the reset state of the reset bit is zero), the current conversion and conversion sequence are aborted, pending interrupts are cleared, and the module is placed in an idle mode.

Table 19-2. Hardware Commands(1)

Command	Opcode (Hex)	Data	Description
BACKGROUND	90	None	Enter background mode if firmware enabled.
READ_BD_BYTE(1)	E4	16-bit address 16-bit data out	Read from memory with BDM in map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
READ_BD_WORD(1)	EC	16-bit address 16-bit data out	Read from memory with BDM in map (may steal cycles if external access). Must be aligned access.
READ_BYTE	E0	16-bit address 16-bit data out	Read from memory with BDM out of map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
READ_WORD	E8	16-bit address 16-bit data out	Read from memory with BDM out of map (may steal cycles if external access). Must be aligned access.
WRITE_BD_BYTE(1)	C4	16-bit address 16-bit data in	Write to memory with BDM in map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
WRITE_BD_WORD(1)	CC	16-bit address 16-bit data in	Write to memory with BDM in map (may steal cycles if external access). Must be aligned access.
WRITE_BYTE	C0	16-bit address 16-bit data in	Write to memory with BDM out of map (may steal cycles if external access) data for odd address on low byte, data for even address on high byte.
WRITE_WORD	C8	16-bit address 16-bit data in	Write to memory with BDM out of map (may steal cycles if external access). Must be aligned access.

1. Use these commands only for reading/writing to BDM locations.The BDM firmware ROM and BDM registers are not normally in the HC12 MCU memory map.Since these locations have the same addresses as some of the normal application memory map, there needs to be a way to decide which physical locations are being accessed by the hardware BDM commands.This gives rise to needing separate memory access commands for the BDM locations as opposed to the normal application locations.In logic, this is accomplished by momentarily enabling the BDM memory resources, just for the access cycles of the READ_BD and WRITE_BD commands.This logic allows the debugging system to unobtrusively access the BDM locations even if the application program is running out of the same memory area in the normal application memory map.

The second type of BDM commands are firmware commands implemented in a small ROM within the HC12 MCU. The CPU must be in background mode to execute firmware commands. The usual way to get to background mode is by the hardware command BACKGROUND. The BDM ROM is located at $FF20 to $FFFF while BDM is active. There are also seven bytes of BDM registers located at $FF00 to $FF06 when BDM is active. The CPU executes code in the BDM firmware to perform the requested operation. The BDM firmware watches for serial commands and executes them as they are received. The firmware commands are shown in Table 19-3.

The external host should delay about 32 target BDMCLK cycles after the data portion of firmware write commands to allow BDM firmware to complete the requested write operation before a new serial command disturbs the BDM SHIFTER register.

The external host should delay about 64 target BDMCLK cycles after a TRACE1 or GO command before starting any new serial command. This delay is needed because the BDM SHIFTER register is used as a temporary data holding register during the exit sequence to user code.

BDM logic retains control of the internal buses until a read or write is completed. If an operation can be completed in a single cycle, it does not intrude on normal CPU12 operation. However, if an operation requires multiple cycles, CPU12 clocks are frozen until the operation is complete.

19.4.4 BDM Lockout

The access to the MCU resources by BDM may be prevented by enabling the BDM lockout feature. When enabled, the BDM lockout mechanism prevents the BDM from being active. In this case the BDM ROM is disabled and does not appear in the MCU memory map.

BDM lockout is enabled by clearing NOBDML bit of EEMCR register. The NOBDML bit is loaded at reset from the SHADOW byte of EEPROM module. Modifying the state of the NOBDML and corresponding EEPROM SHADOW bit is only possible in special modes.

Please refer to EEPROM Memory for NOBDML information.

19.4.4.1 Enabling BDM lockout

Enabling the BDM lockout feature is only possible in special modes (SMODN=0) and is accomplished by the following steps.

1. Remove the SHADOW byte protection by clearing SHPROT bit in EEPROT register.
2. Clear NOSHB bit in EEMCR register to make the SHADOW byte visible at $0FC0.
3. Program bit 7 of the SHADOW byte like a regular EEPROM location at address $0FC0 (write $7F into address $0FC0). Do not