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	Not For New Designs
Core Processor	CPU16
Core Size	16-Bit
Speed	25MHz
Connectivity	EBI/EMI, SCI, SPI
Peripherals	POR, PWM, WDT
Number of I/O	16
Program Memory Size	-
Program Memory Type	ROMless
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	132-BQFP Bumpered
Supplier Device Package	132-PQFP (24.13x24.13)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc16z1ceh25

Table 4-2 Instruction Set Summary (Continued)

Mnemonic	Operation	Description	Address	Instruction			Condition Codes							
			Mode	Opcode	Operand	Cycles	S	MV	H	EV	N	Z	V	C
RORW	Rotate Right Word		IND16, X IND16, Y IND16, Z EXT	270E 271E 272E 273E	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	Δ	Δ	Δ	Δ
RTI[3]	Return from Interrupt	(SK : SP) + 2 ⇒ SK : SP Pull CCR (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 6 ⇒ PK : PC	INH	2777	—	12	Δ	Δ	Δ	Δ	Δ	Δ	Δ	Δ
RTS[4]	Return from Subroutine	(SK : SP) + 2 ⇒ SK : SP Pull PK (SK : SP) + 2 ⇒ SK : SP Pull PC (PK : PC) – 2 ⇒ PK : PC	INH	27F7	—	12	—	—	—	—	—	—	—	—
SBA	Subtract B from A	(A) – (B) ⇒ A	INH	370A	—	2	—	—	—	—	Δ	Δ	Δ	Δ
SBCA	Subtract with Carry from A	(A) – (M) – C ⇒ A	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	42 52 62 72 1742 1752 1762 1772 2742 2752 2762	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SBCB	Subtract with Carry from B	(B) – (M) – C ⇒ B	IND8, X IND8, Y IND8, Z IMM8 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	C2 D2 E2 F2 17C2 17D2 17E2 17F2 27C2 27D2 27E2	ff ff ff ii gggg gggg gggg hh ll — — —	6 6 6 2 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SBCD	Subtract with Carry from D	(D) – (M : M + 1) – C ⇒ D	IND8, X IND8, Y IND8, Z IMM16 IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	82 92 A2 37B2 37C2 37D2 37E2 37F2 2782 2792 27A2	ff ff ff jj kk gggg gggg gggg hh ll — — —	6 6 6 4 6 6 6 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SBCE	Subtract with Carry from E	(E) – (M : M + 1) – C ⇒ E	IMM16 IND16, X IND16, Y IND16, Z EXT	3732 3742 3752 3762 3772	jj kk gggg gggg gggg hh ll	4 6 6 6 6	—	—	—	—	Δ	Δ	Δ	Δ
SDE	Subtract D from E	(E) – (D)⇒ E	INH	2779	—	2	—	—	—	—	Δ	Δ	Δ	Δ
STAA	Store A	(A) ⇒ M	IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT E, X E, Y E, Z	4A 5A 6A 174A 175A 176A 177A 274A 275A 276A	ff ff ff gggg gggg gggg hh ll — — —	4 4 4 6 6 6 6 4 4 4	—	—	—	—	Δ	Δ	0	—

Table 4-5 Exception Vector Table

Vector Number	Vector Address	Address Space	Type of Exception
0	0000	P	Reset — Initial ZK, SK, and PK
	0002	P	Reset — Initial PC
	0004	P	Reset — Initial SP
	0006	P	Reset — Initial IZ (Direct Page)
4	0008	D	Breakpoint
5	000A	D	Bus Error
6	000C	D	Software Interrupt
7	000E	D	Illegal Instruction
8	0010	D	Division by Zero
9 – E	0012 – 001C	D	Unassigned, Reserved
F	001E	D	Uninitialized Interrupt
10	0020	D	Unassigned, Reserved
11	0022	D	Level 1 Interrupt Autovector
12	0024	D	Level 2 Interrupt Autovector
13	0026	D	Level 3 Interrupt Autovector
14	0028	D	Level 4 Interrupt Autovector
15	002A	D	Level 5 Interrupt Autovector
16	002C	D	Level 6 Interrupt Autovector
17	002E	D	Level 7 Interrupt Autovector
18	0030	D	Spurious Interrupt
19 – 37	0032 – 006E	D	Unassigned, Reserved
38 – FF	0070 – 01FE	D	User-Defined Interrupts

4.13.2 Exception Stack Frame

During exception processing, the contents of the program counter and condition code register are stacked at a location pointed to by SK : SP. Unless it is altered during exception processing, the stacked PK : PC value is the address of the next instruction in the current instruction stream, plus $0006. **Figure 4-6** shows the exception stack frame.

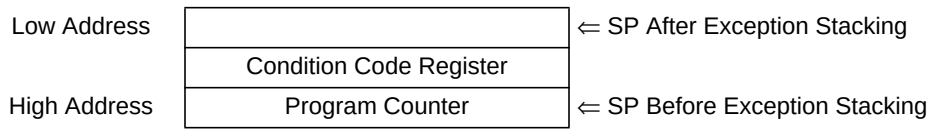

Figure 4-6 Exception Stack Frame Format

4.14.1.1 IPIPE0/IPIPE1 Multiplexing

Six types of information are required to track pipeline activity. To generate the six state signals, eight pipeline states are encoded and multiplexed into IPIPE0 and IPIPE1. The multiplexed signals have two phases. State signals are active low. **Table 4-6** shows the encoding scheme.

Table 4-6 IPIPE0/IPIPE1 Encoding

Phase	IPIPE1 State	IPIPE0 State	State Signal Name
1	0 0 1 1	0 1 0 1	START and FETCH FETCH START NULL
2	0 0 1 1	0 1 0 1	INVALID ADVANCE EXCEPTION NULL

IPIPE0 and IPIPE1 are timed so that a logic analyzer can capture all six pipeline state signals and address, data, or control bus state in any single bus cycle. Refer to **APPENDIX A ELECTRICAL CHARACTERISTICS** for specifications.

State signals can be latched asynchronously on the falling and rising edges of either address strobe ($\overline{AS}$) or data strobe ($\overline{DS}$). They can also be latched synchronously using the microcontroller CLKOUT signal. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for more information on the CLKOUT signal, state signals, and state signal demux logic.

4.14.1.2 Combining Opcode Tracking with Other Capabilities

Pipeline state signals are useful during normal instruction execution and execution of exception handlers. The signals provide a complete model of the pipeline up to the point a breakpoint is acknowledged.

Breakpoints are acknowledged after an instruction has executed, when it is in pipeline stage C. A breakpoint can initiate either exception processing or background debug mode. IPIPE0/IPIPE1 are not usable when the CPU16 is in background debug mode.

4.14.2 Breakpoints

Breakpoints are set by assertion of the microcontroller $\overline{BKPT}$ pin. The CPU16 supports breakpoints on any memory access. Acknowledged breakpoints can initiate either exception processing or background debug mode. After BDM has been enabled, the CPU16 will enter BDM when the $\overline{BKPT}$ input is asserted.

- If $\overline{BKPT}$ assertion is synchronized with an instruction prefetch, the instruction is tagged with the breakpoint when it enters the pipeline, and the breakpoint occurs after the instruction executes.
- If $\overline{BKPT}$ assertion is synchronized with an operand fetch, breakpoint processing occurs at the end of the instruction during which $\overline{BKPT}$ is latched.

Both writes must occur before time-out in the order listed. Any number of instructions can be executed between the two writes.

Watchdog clock rate is affected by the software watchdog prescale (SWP) bit and the software watchdog timing (SWT[1:0]) field in SYPCR.

SWP determines system clock prescaling for the watchdog timer and determines that one of two options, either no prescaling or prescaling by a factor of 512, can be selected. The value of SWP is affected by the state of the MODCLK pin during reset, as shown in **Table 5-9**. System software can change SWP value.

Table 5-9 MODCLK Pin and SWP Bit During Reset

MODCLK	SWP
0 (External Clock)	1 (÷ 512)
1 (Internal Clock)	0 (÷ 1)

SWT[1:0] selects the divide ratio used to establish the software watchdog time-out period.

The following equation calculates the time-out period for a slow reference frequency, where f_{ref} is equal to the EXTAL crystal frequency.

$$\text{Time-Out Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{ref}}$$

The following equation calculates the time-out period for a fast reference frequency, where f_{ref} is equal to the EXTAL crystal frequency.

$$\text{Time-Out Period} = \frac{(128)(\text{Divide Ratio Specified by SWP and SWT[1:0]})}{f_{ref}}$$

The following equation calculates the time-out period for an externally input clock frequency on both slow and fast reference frequency devices, when f_{sys} is equal to the system clock frequency.

$$\text{Time-Out Period} = \frac{\text{Divide Ratio Specified by SWP and SWT[1:0]}}{f_{sys}}$$

Table 5-10 shows the divide ratio for each combination of SWP and SWT[1:0] bits. When SWT[1:0] are modified, a watchdog service sequence must be performed before the new time-out period can take effect.

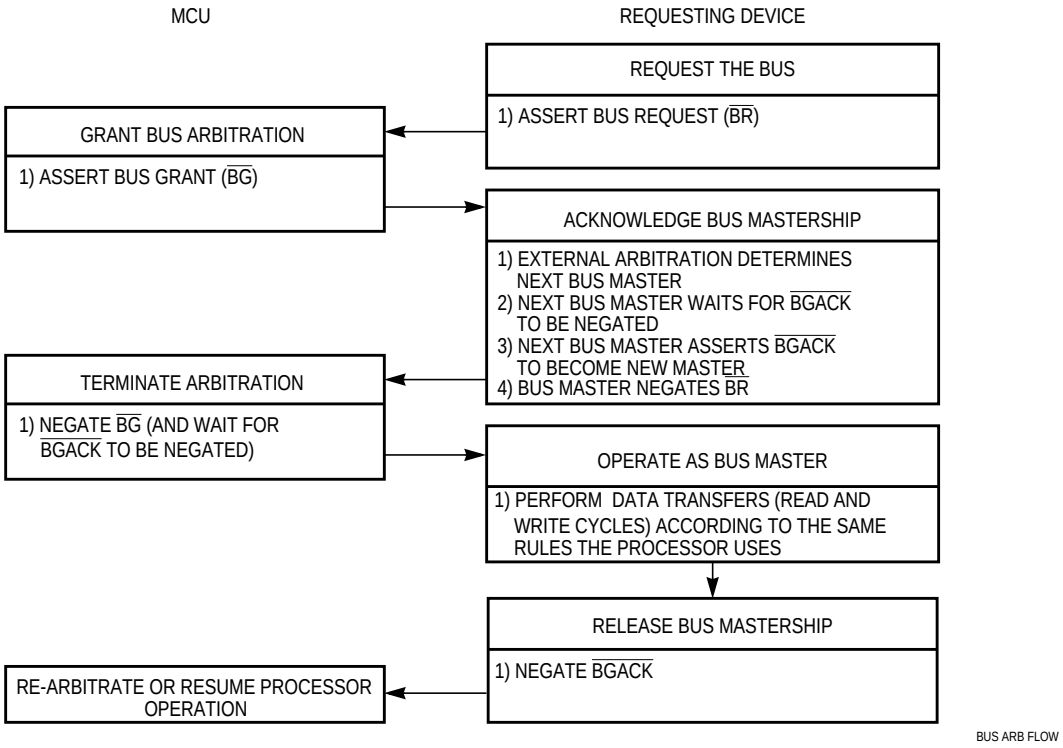

Figure 5-17 Bus Arbitration Flowchart for Single Request

5.6.6.1 Show Cycles

The MCU normally performs internal data transfers without affecting the external bus, but it is possible to show these transfers during debugging. AS is not asserted externally during show cycles.

Show cycles are controlled by the SHEN[1:0] in SIMCR. This field is set to %00 by reset. When show cycles are disabled, the address bus, function codes, size, and read/write signals reflect internal bus activity, but AS and DS are not asserted externally and external data bus pins are in high-impedance state during internal accesses. Refer to **5.2.3 Show Internal Cycles** and the *SIM Reference Manual* (SIMRM/AD) for more information.

When show cycles are enabled, DS is asserted externally during internal cycles, and internal data is driven out on the external data bus. Because internal cycles normally continue to run when the external bus is granted, one SHEN[1:0] encoding halts internal bus activity while there is an external master.

SIZ[1:0] signals reflect bus allocation during show cycles. Only the appropriate portion of the data bus is valid during the cycle. During a byte write to an internal address, the portion of the bus that represents the byte that is not written reflects internal bus conditions, and is indeterminate. During a byte write to an external address, the data multiplexer in the SIM causes the value of the byte that is written to be driven out on both bytes of the data bus.

5.8 Interrupts

Interrupt recognition and servicing involve complex interaction between the SIM, the CPU16, and a device or module requesting interrupt service. This discussion provides an overview of the entire interrupt process. Chip-select logic can also be used to respond to interrupt requests. Refer to **5.9 Chip-Selects** for more information.

5.8.1 Interrupt Exception Processing

The CPU16 handles interrupts as a type of asynchronous exception. An exception is an event that preempts normal processing. Exception processing makes the transition from normal instruction execution to execution of a routine that deals with an exception. Each exception has an assigned vector that points to an associated handler routine. These vectors are stored in a vector table located in the first 512 bytes of address bank 0. The CPU16 uses vector numbers to calculate displacement into the table. Refer to **4.13 Exceptions** for more information.

5.8.2 Interrupt Priority and Recognition

The CPU16 provides for seven levels of interrupt priority (1 – 7), seven automatic interrupt vectors, and 200 assignable interrupt vectors. All interrupts with priorities less than seven can be masked by the interrupt priority (IP) field in the condition code register.

There are seven interrupt request signals ($\overline{\text{IRQ}}$[7:1]). These signals are used internally on the IMB, and there are corresponding pins for external interrupt service requests. The CPU16 treats all interrupt requests as though they come from internal modules; external interrupt requests are treated as interrupt service requests from the SIM. Each of the interrupt request signals corresponds to an interrupt priority level. $\overline{\text{IRQ1}}$ has the lowest priority and $\overline{\text{IRQ7}}$ the highest.

The IP field consists of three bits (CCR[7:5]). Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value (except for $\overline{\text{IRQ7}}$) from being recognized and processed. When IP contains %000, no interrupt is masked. During exception processing, the IP field is set to the priority of the interrupt being serviced.

Interrupt recognition is determined by interrupt priority level and interrupt priority (IP) mask value. The interrupt priority mask consists of three bits in the CPU16 condition code register (CCR[7:5]). Binary values %000 to %111 provide eight priority masks. Masks prevent an interrupt request of a priority less than or equal to the mask value from being recognized and processed. $\overline{\text{IRQ7}}$, however, is always recognized, even if the mask value is %111.

$\overline{\text{IRQ}}$[7:1] are active-low level-sensitive inputs. The low on the pin must remain asserted until an interrupt acknowledge cycle corresponding to that level is detected.

$\overline{\text{IRQ7}}$ is transition-sensitive as well as level-sensitive: a level-7 interrupt is not detected unless a falling edge transition is detected on the $\overline{\text{IRQ7}}$ line. This prevents redundant servicing and stack overflow. A non-maskable interrupt is generated each time $\overline{\text{IRQ7}}$ is asserted as well as each time the priority mask is written while $\overline{\text{IRQ7}}$ is asserted. If $\overline{\text{IRQ7}}$ is asserted and the IP mask is written to any new value (including %111), $\overline{\text{IRQ7}}$ will be recognized as a new $\overline{\text{IRQ7}}$.

5.10.3 Data Registers

A write to the port E and port F data registers (PORTE[0:1] and PORTF[0:1]) is stored in an internal data latch, and if any pin in the corresponding port is configured as an output, the value stored for that bit is driven out on the pin. A read of a data register returns the value at the pin only if the pin is configured as a discrete input. Otherwise, the value read is the value stored in the register. Both data registers can be accessed in two locations and can be read or written at any time.

5.11 Factory Test

The test submodule supports scan-based testing of the various MCU modules. It is integrated into the SIM to support production test. Test submodule registers are intended for Freescale use only. Register names and addresses are provided in **APPENDIX D REGISTER SUMMARY** to show the user that these addresses are occupied. The QUOT pin is also used for factory test.

Grounding is the most important factor influencing analog circuit performance in mixed signal systems (or in stand-alone analog systems). Close attention must be paid to avoid introducing additional sources of noise into the analog circuitry. Common sources of noise include ground loops, inductive coupling, and combining digital and analog grounds together inappropriately.

The problem of how and when to combine digital and analog grounds arises from the large transients which the digital ground must handle. If the digital ground is not able to handle the large transients, the current from the large transients can return to ground through the analog ground. It is the excess current overflowing into the analog ground which causes performance degradation by developing a differential voltage between the true analog ground and the microcontroller's ground pin. The end result is that the ground observed by the analog circuit is no longer true ground and often ends in skewed results.

Two similar approaches designed to improve or eliminate the problems associated with grounding excess transient currents involve star-point ground systems. One approach is to star-point the different grounds at the power supply origin, thus keeping the ground isolated. Refer to **Figure 8-6**.

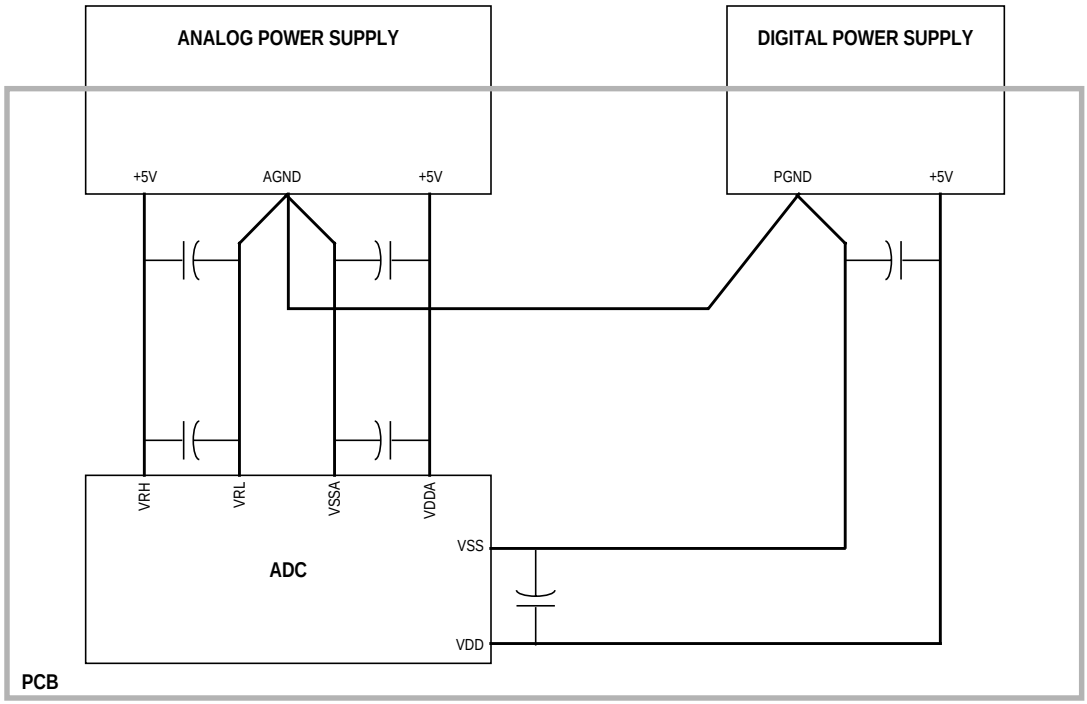

Figure 8-6 Star-Ground at the Point of Power Supply Origin

Another approach is to star-point the different grounds near the analog ground pin on the microcontroller by using small traces for connecting the non-analog grounds to the analog ground. The small traces are meant only to accommodate DC differences, not AC transients.

Data transfer is synchronized with the internally-generated serial clock (SCK). Control bits CPHA and CPOL in SPCR control clock phase and polarity. Combinations of CPHA and CPOL determine the SCK edge on which the master MCU drives outgoing data from the MOSI pin and latches incoming data from the MISO pin.

10.3.3.2 Slave Mode

Clearing the MSTR bit in SPCR selects slave mode operation. In slave mode, the SPI is unable to initiate serial transfers. Transfers are initiated by an external bus master. Slave mode is typically used on a multimaster SPI bus. Only one device can be bus master (operate in master mode) at any given time.

When using the SPI in slave mode, include the following steps:

1. Write to the MMCR and interrupt registers. Refer to **10.5 MCCI Initialization** for more information.
2. Write to the MPAR to assign the following pins to the SPI: MISO, MOSI, and $\overline{SS}$. MISO is used for serial data output in slave mode, and MOSI is used for serial data input. Either or both may be necessary, depending on the particular application. SCK is the input serial clock. $\overline{SS}$ selects the SPI when asserted.
3. Write to the MDDR to direct the data flow on SPI pins. Configure the SCK, MOSI, and $\overline{SS}$ pins as inputs. Configure MISO as an output.
4. Write to the SPCR to assign values for CPHA, CPOL, SIZE, LSBF, WOMP, and SPIE. Set the MSTR bit to select master operation. Set the SPE bit to enable the SPI. (The BAUD field in the SPCR of the slave device has no effect on SPI operation.)

When SPE is set and MSTR is clear, a low state on the $\overline{SS}$ pin initiates slave mode operation. The $\overline{SS}$ pin is used only as an input.

After a byte or word of data is transmitted, the SPI sets the SPIF flag. If the SPIE bit in SPCR is set, an interrupt request is generated when SPIF is asserted.

Transfer is synchronized with the externally generated SCK. The CPHA and CPOL bits determine the SCK edge on which the slave MCU latches incoming data from the MOSI pin and drives outgoing data from the MISO pin.

10.3.4 SPI Clock Phase and Polarity Controls

Two bits in the SPCR determine SCK phase and polarity. The clock polarity (CPOL) bit selects clock polarity (high true or low true clock). The clock phase control bit (CPHA) selects one of two transfer formats and affects the timing of the transfer. The clock phase and polarity should be the same for the master and slave devices. In some cases, the phase and polarity may be changed between transfers to allow a master device to communicate with slave devices with different requirements. The flexibility of the SPI system allows it to be directly interfaced to almost any existing synchronous serial peripheral.

10.4.5.4 Parity Checking

The PT bit in SCCR1 selects either even (PT = 0) or odd (PT = 1) parity. PT affects received and transmitted data. The PE bit in SCCR1 determines whether parity checking is enabled (PE = 1) or disabled (PE = 0). When PE is set, the MSB of data in a frame is used for the parity function. For transmitted data, a parity bit is generated for received data; the parity bit is checked. When parity checking is enabled, the PF bit in the SCI status register (SCSR) is set if a parity error is detected.

Enabling parity affects the number of data bits in a frame, which can in turn affect frame size. **Table 10-7** shows possible data and parity formats.

Table 10-7 Effect of Parity Checking on Data Size

M	PE	Result
0	0	8 data bits
0	1	7 data bits, 1 parity bit
1	0	9 data bits
1	1	8 data bits, 1 parity bit

10.4.5.5 Transmitter Operation

The transmitter consists of a serial shifter and a parallel data register (TDR) located in the SCI data register (SCDR). The serial shifter cannot be directly accessed by the CPU16. The transmitter is double-buffered, which means that data can be loaded into the TDR while other data is shifted out. The TE bit in SCCR1 enables (TE = 1) and disables (TE = 0) the transmitter.

Shifter output is connected to the TXD pin while the transmitter is operating (TE = 1, or TE = 0 and transmission in progress). Wired-OR operation should be specified when more than one transmitter is used on the same SCI bus. The WOMS bit in SCCR1 determines whether TXD is an open-drain (wired-OR) output or a normal CMOS output. An external pull-up resistor on TXD is necessary for wired-OR operation. WOMS controls TXD function whether the pin is used by the SCI or as a general-purpose I/O pin.

Data to be transmitted is written to SCDR, then transferred to the serial shifter. The transmit data register empty (TDRE) flag in SCSR shows the status of TDR. When TDRE = 0, the TDR contains data that has not been transferred to the shifter. Writing to SCDR again overwrites the data. TDRE is set when the data in the TDR is transferred to the shifter. Before new data can be written to the SCDR, however, the processor must clear TDRE by writing to SCSR. If new data is written to the SCDR without first clearing TDRE, the data will not be transmitted.

The transmission complete (TC) flag in SCSR shows transmitter shifter state. When TC = 0, the shifter is busy. TC is set when all shifting operations are completed. TC is not automatically cleared. The processor must clear it by first reading SCSR while TC is set, then writing new data to SCDR.

Data written to PWMA and PWMB is not used until the end of a complete cycle. This prevents spurious short or long pulses when register values are changed. The current duty cycle value is stored in the appropriate PWM buffer register (PWMBUFA or PWMBUFB). The new value is transferred from the PWM register to the buffer register at the end of the current cycle.

Registers PWMA, PWMB, and PWMC are reset to $00 during reset. These registers may be written or read at any time. PWMC is implemented as the lower byte of a 16-bit register. The upper byte is the CFORC register. The buffer registers, PWMBUFA and PWMBUFB, are read-only at all times and may be accessed as separate bytes or as one 16-bit register.

Pins PWMA and PWMB can also be used for general-purpose output. The values of the F1A and F1B bits in PWMC are driven out on the corresponding PWM pins when normal PWM operation is disabled. When read, the F1A and F1B bits reflect the states of the PWMA and PWMB pins.

Table A-2 Typical Ratings, 2.7 to 3.6V, 16.78-MHz Operation

Num	Rating	Symbol	Value	Unit
1	Supply Voltage	V_{DD}	3.0	V
2	Operating Temperature	T_A	25	°C
3	V_{DD} Supply Current RUN LPSTOP, VCO off LPSTOP, External clock, max f_{sys}	I_{DD}	 38 70 1	 mA µA mA
4	Clock Synthesizer Operating Voltage	V_{DDSYN}	3.0	V
5	V_{DDSYN} Supply Current 4.194 MHz VCO on, maximum f_{sys} 32.768 kHz VCO on, maximum f_{sys} 4.194 MHz External Clock, maximum f_{sys} 32.768 kHz External Clock, maximum f_{sys} 4.194 MHz LPSTOP, VCO off 32.768 kHz LPSTOP, VCO off 4.194 MHz V_{DD} powered down 32.768 kHz V_{DD} powered down	I_{DDSYN}	 TBD 200 TBD 1 TBD 20 TBD 10	 mA µA µA mA mA µA µA µA
6	RAM Standby Voltage	V_{SB}	3	V
7	RAM Standby Current Normal RAM operation Standby operation	I_{SB}	 3 3	 µA µA
8	Power Dissipation	P_D	120	mW

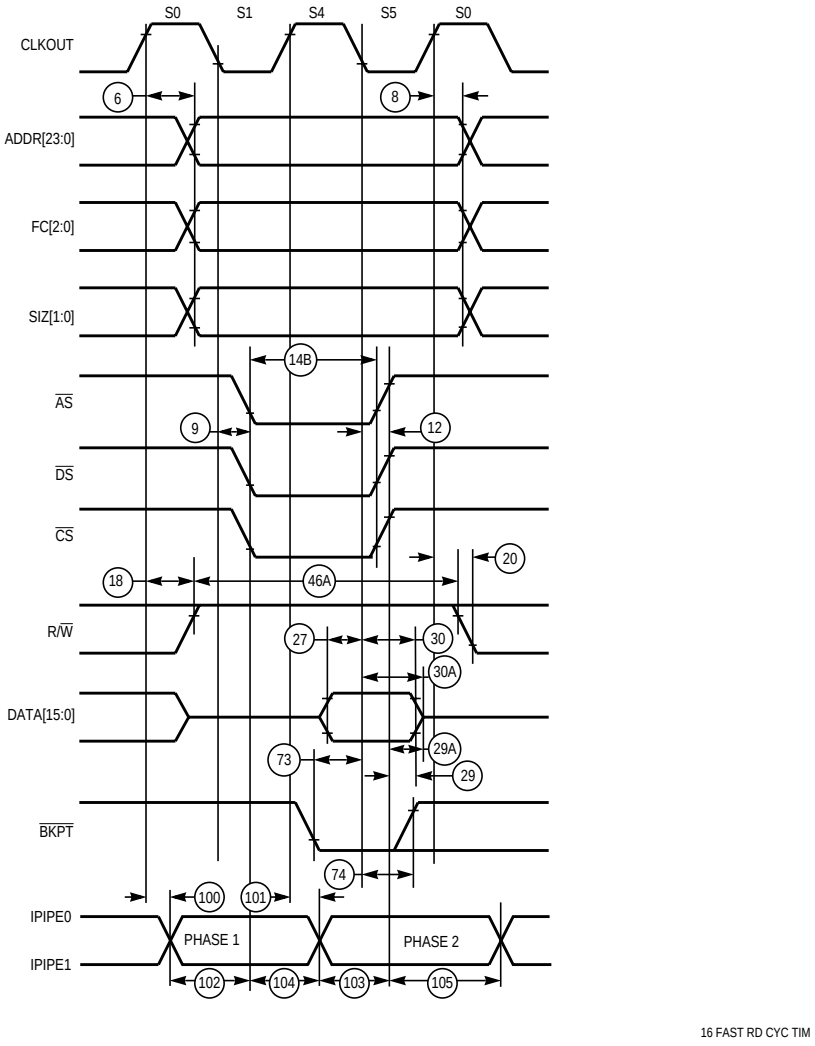

Figure A-6 Fast Termination Read Cycle Timing Diagram

Table A-19 Low Voltage 16.78-MHz Background Debug Mode Timing

(V_{DD} and V_{DDSYN} = 2.7 to 3.6Vdc, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
B0	DSI Input Setup Time	t_{DSISU}	15	—	ns
B1	DSI Input Hold Time	t_{DSIH}	15	—	ns
B2	DSCLK Setup Time	t_{DSCSU}	15	—	ns
B3	DSCLK Hold Time	t_{DSCH}	15	—	ns
B4	DSO Delay Time	t_{DSOD}	—	35	ns
B5	DSCLK Cycle Time	t_{DSCCYC}	2	—	t_{cyc}
B6	CLKOUT High to FREEZE Asserted/Negated	t_{FRZAN}	—	50	ns
B7	CLKOUT High to IPIPE1 High Impedance	t_{IFZ}	—	50	ns
B8	CLKOUT High to IPIPE1 Valid	t_{IF}	—	50	ns
B9	DSCLK Low Time	t_{DSCLO}	1	—	t_{cyc}
B10	IPIPE1 High Impedance to FREEZE Asserted	t_{IPFA}	TBD	—	t_{cyc}
B11	FREEZE Negated to IPIPE[0:1] Active	t_{FRIP}	TBD	—	t_{cyc}

NOTES:
1. All AC timing is shown with respect to V_{IH}/V_{IL} levels unless otherwise noted.

Table A-20 16.78-MHz Background Debug Mode Timing

(V_{DD} and V_{DDSYN} = 5.0 Vdc ± 10%, V_{SS} = 0 Vdc, $T_A = T_L$ to T_H)[1]

Num	Characteristic	Symbol	Min	Max	Unit
B0	DSI Input Setup Time	t_{DSISU}	15	—	ns
B1	DSI Input Hold Time	t_{DSIH}	10	—	ns
B2	DSCLK Setup Time	t_{DSCSU}	15	—	ns
B3	DSCLK Hold Time	t_{DSCH}	10	—	ns
B4	DSO Delay Time	t_{DSOD}	—	25	ns
B5	DSCLK Cycle Time	t_{DSCCYC}	2	—	t_{cyc}
B6	CLKOUT High to FREEZE Asserted/Negated	t_{FRZAN}	—	50	ns
B7	CLKOUT High to IPIPE1 High Impedance	t_{IFZ}	—	TBD	ns
B8	CLKOUT High to IPIPE1 Valid	t_{IF}	—	TBD	ns
B9	DSCLK Low Time	t_{DSCLO}	1	—	t_{cyc}
B10	IPIPE1 High Impedance to FREEZE Asserted	t_{IPFA}	TBD	—	t_{cyc}
B11	FREEZE Negated to IPIPE[0:1] Active	t_{FRIP}	TBD	—	t_{cyc}

NOTES:
1. All AC timing is shown with respect to V_{IH}/V_{IL} levels unless otherwise noted.

D.4.4 ROM Bootstrap Words

ROMBS0 — ROM Bootstrap Word 0 **$YFF830**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED				ZK[3:0]				SK[3:0]				PK[3:0]			

ROMBS1 — ROM Bootstrap Word 1 **$YFF832**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PC[15:0]															

ROMBS2 — ROM Bootstrap Word 2 **$YFF834**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SP[15:0]															

ROMBS3 — ROM Bootstrap Word 3 **$YFF836**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IZ[15:0]															

Typically, CPU16 reset vectors reside in non-volatile memory and are fetched when the CPU16 comes out of reset. These four words can be used as reset vectors with the contents specified at mask time. The content of these words cannot be changed. On generic (blank ROM) MC68HC16Z2 and MC68HC16Z3 devices, ROMBS[0:3] are masked to $0000. When the ROM on the MC68HC16Z2 and MC68HC16Z3 is masked with customer specific code, ROMBS[0:3] respond to system addresses $00000 to $00006 during the reset vector fetch if $\overline{\text{BOOT}}$ = 0.

HMIE — HALTA and MODF Interrupt Enable
0 = HALTA and MODF interrupts disabled.
1 = HALTA and MODF interrupts enabled.
HMIE enables interrupt requests generated by the HALTA status flag or the MODF status flag in SPSR.

HALT — Halt QSPI
0 = QSPI operates normally.
1 = QSPI is halted for subsequent restart.
When HALT is set, the QSPI stops on a queue boundary. It remains in a defined state from which it can later be restarted.

SPIF — QSPI Finished Flag
0 = QSPI is not finished.
1 = QSPI is finished.
SPIF is set after execution of the command at the address in ENDQP[3:0].

MODF — Mode Fault Flag
0 = Normal operation.
1 = Another SPI node requested to become the network SPI master while the QSPI was enabled in master mode.
The QSPI asserts MODF when the QSPI is in master mode (MSTR = 1) and the $\overline{SS}$ input pin is negated by an external driver.

HALTA — Halt Acknowledge Flag
0 = QSPI is not halted.
1 = QSPI is halted.
HALTA is set when the QSPI halts in response to setting the SPCR3 HALT bit.

Bit 4 — Not Implemented

CPTQP[3:0] — Completed Queue Pointer
CPTQP[3:0] points to the last command executed. It is updated when the current command is complete. When the first command in a queue is executing, CPTQP[3:0] contains either the reset value $0 or a pointer to the last command completed in the previous queue.

D.6.14 Receive Data RAM

RR[0:F] — Receive Data RAM **$YFFD00 – $YFFD1F**

Data received by the QSPI is stored in this segment. The CPU16 reads this segment to retrieve data from the QSPI. Data stored in receive RAM is right-justified. Unused bits in a receive queue entry are set to zero by the QSPI upon completion of the individual queue entry. Receive RAM data can be accessed using byte, word, or long-word addressing.

D.8 General-Purpose Timer

Table D-42 GPT Address Map

Address[1]	15 8	7 0
$YFF900	GPT Module Configuration Register (GPTMCR)	
$YFF902	GPT Module Test Register (GPTMTR)	
$YFF904	GPT Interrupt Configuration Register (ICR)	
$YFFE06	Port GP Data Direction Register (DDRGP)	Port GP Data Register (PORTGP)
$YFF908	Output Compare 1 Action Mask Register (OC1M)	Output Compare 1 Action Data Register (OC1D)
$YFF90A	Timer Counter Register (TCNT)	
$YFF90C	Pulse Accumulator Control Register (PACTL)	Pulse Accumulator Counter Register (PACNT)
$YFF90E	Timer Input Capture Register 1 (TIC1)	
$YFF910	Timer Input Capture Register 2 (TIC2)	
$YFF912	Timer Input Capture Register 3 (TIC3)	
$YFF914	Timer Output Compare Register 1 (TOC1)	
$YFF916	Timer Output Compare Register 2 (TOC2)	
$YFF918	Timer Output Compare Register 3 (TOC3)	
$YFF91A	Timer Output Compare Register 4 (TOC4)	
$YFF91C	Timer Input Capture 4/Output Compare Register 5 (TI4/O5)	
$YFF91E	Timer Control Register 1 (TCTL1)	Timer Control Register 2 (TCTL2)
$YFF920	Timer Mask Register 1 (TMSK1)	Timer Mask Register 2 (TMSK2)
$YFF922	Timer Flag Register 1 (TFLG1)	Timer Flag Register 2 (TFLG2)
$YFF924	Compare Force Register (CFORC)	PWM Control Register C (PWMC)
$YFF926	PWM Control Register A (PWMA)	PWM Control Register B (PWMB)
$YFF928	PWM Count Register (PWMCNT)	
$YFF92A	PWM Buffer Register A (PWMBUFA)	PWM Buffer Register B (PWMBUFB)
$YFF92C	GPT Prescaler Register (PRESCL)	
$YFF92E – $YFF93F	Reserved	

NOTES:
1. Y = M111, where M is the logic state of the MM bit in the SIMCR.

D.8.1 GPT Module Configuration Register

GPTMCR — GPT Module Configuration Register **$YFF900**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	FRZ1	FRZ0	STOPP	INCP	0	0	0	SUPV	0	0	0	IARB			
RESET:															
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0

The GPTMCR contains parameters for configuring the GPT.

```
        AIX  #$01           ;increment IX to point to the next byte
        BRA  SEND_STRING    ;loop back and do next byte in string

STRING_DONE:                ;subroutine to implement delay between messages
        LDE  #$FFFF         ;load accumulator E with the delay time
        STE  0,Z            ;set up the counter
LOOP:   DECW 0,Z            ;decrement the counter
        BNE  LOOP           ;count down to zero
        RTS                 ;finish delay loop go back to main

SEND_CH:                    ;subroutine to send out one byte to SCI
        LDAA SCSR           ;read SCI status reg to check/clear TDRE bit
        ANDA #$01           ;check only the TDRE flag bit
        BEQ  SEND_CH        ;if TDR is not empty, go back to check it again
        LDAA #$00           ;clear A to send a full word to SCDR ($FFC0E)
        STD  SCDR           ;transmit one ASCII character to the screen
TC_LOOP:
        LDAB SCSR+1
        ANDB #$80           ;test the TC bit (transfer complete)
        BEQ  TC_LOOP        ;continue to wait until TC is set

        RTS                 ;finish sending out byte

STRING    DC     'I AM A HAPPY EVB16 RUNNING YOUR CODE!!!',0A,0D,00

*****  Interrupts/Exceptions  *****

BDM: BGND                   ;exception vectors point here
                            ;and put the user in background debug mode

*****  Reserve data and stack space  *****

        ORG  $10000         ;start of 1K internal SRAM for data & stack

COUNTER DS     2           ;space for delay counter
```

E.2.4 GPT Programming Example

The following programming example involves demonstrating basic general-purpose timer module (GPT) functions.

Refer to **SECTION 11 GENERAL-PURPOSE TIMER** for more information on the GPT.

E.2.4.1 Example 7 - Basic GPT Functions

```
*       Description : This program demonstrates some basic GPT functions.
*
*                     The 1st demo requires that the pins OC2, IC1, IC2,
*                     and IC3 be tied together so that OC2 may drive
*                     IC1, IC2, & IC3.
*
*                   * In the second demo, the PAI pin should be connected
*                     to the PWMA pin.  A bell on the dummy terminal
*                     will ring when the Pulse Accumulator Counter
```

–F–

–G–