



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](#)"

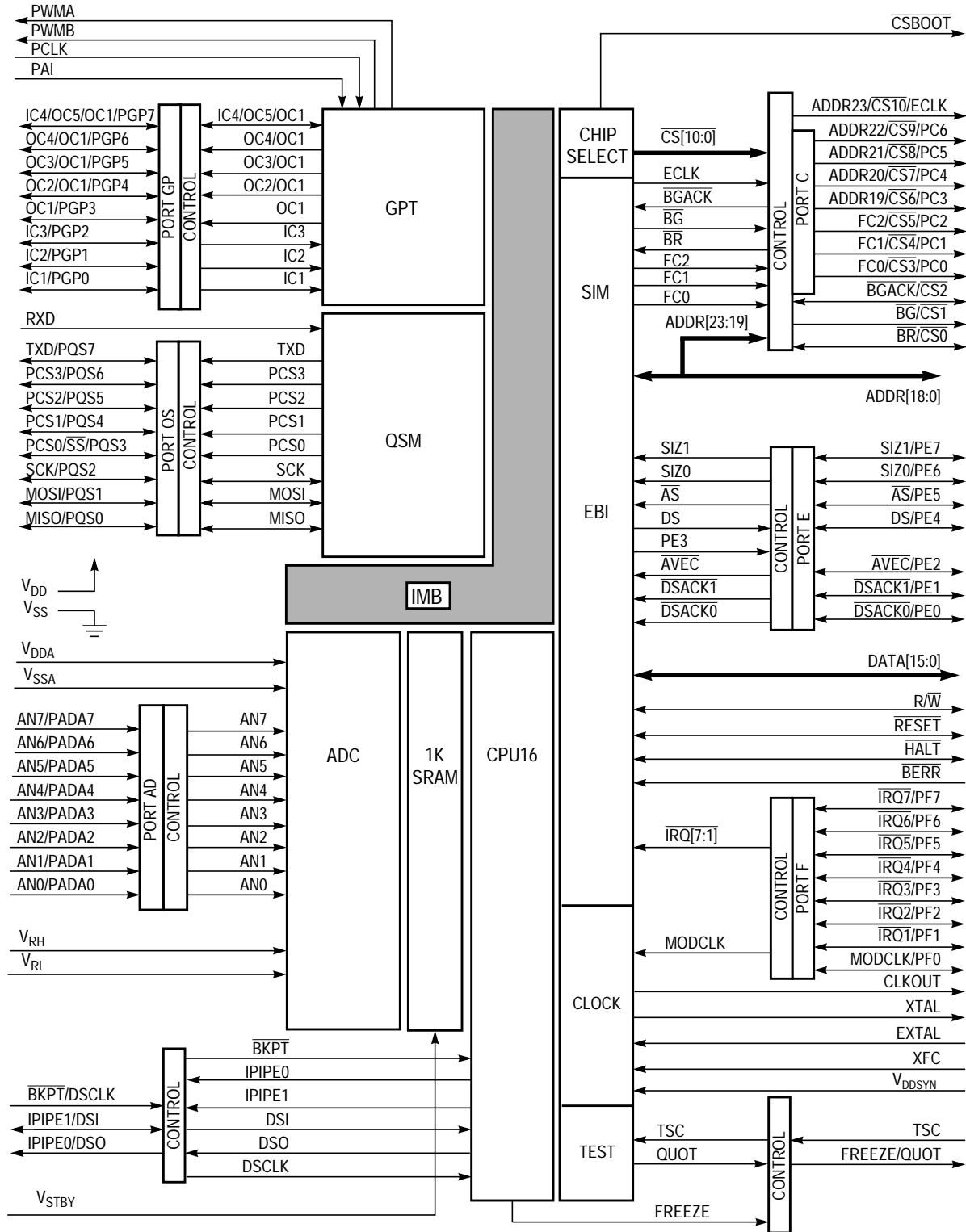
#### Details

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	144-LQFP
Supplier Device Package	144-LQFP (20x20)
Purchase URL	<a href="https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc16z1cag25">https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc16z1cag25</a>



**TABLE OF CONTENTS**  
**(Continued)**












Paragraph	Title	Page
E.1.5	INITRAM.ASM .....	E-11
E.1.6	INITSCI.ASM .....	E-12
E.2	Programming Examples .....	E-12
E.2.1	SIM Programming Examples .....	E-13
E.2.1.1	Example 1 - Using Ports E and F .....	E-13
E.2.1.2	Example 2 - Using Chip-Selects .....	E-14
E.2.1.3	Example 3 - Changing Clock Frequencies .....	E-16
E.2.1.4	Example 4 - Software Watchdog, Periodic Interrupt, and Autovector Demo .....	E-18
E.2.2	CPU16 Programming Example .....	E-23
E.2.2.1	Example 5 - Indexed and Extended Addressing .....	E-23
E.2.3	QSM/SCI Programming Example .....	E-24
E.2.3.1	Example 6 - Using an SCI Port .....	E-24
E.2.4	GPT Programming Example .....	E-25
E.2.4.1	Example 7 - Basic GPT Functions .....	E-25



HC16Z1/CKZ1/CMZ1 BLOCK

Figure 3-1 MC68HC16Z1/CK16Z1/CM16Z1 Block Diagram

**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
PULM <sup>1</sup>	Pull Multiple Registers	For mask bits 0 to 7:  If mask bit set (SK : SP) + 2 $\Rightarrow$ SK : SP Pull register	IMM8	35	ii	4+2(N+1)  N = number of registers pulled	$\Delta$	$\Delta$	$\Delta$	$\Delta$	$\Delta$	$\Delta$	$\Delta$	$\Delta$
PULMAC	Pull MAC State	Stack $\Rightarrow$ MAC Registers	INH	27B9	—	16	—	—	—	—	—	—	—	—
RMAC	Repeating Multiply and Accumulate Signed 16-Bit Fractions	Repeat until (E) < 0 (AM) + (H) * (I) $\Rightarrow$ AM Qualified (IX) $\Rightarrow$ IX; Qualified (IY) $\Rightarrow$ IY; (M : M + 1) <sub>X</sub> $\Rightarrow$ H; (M : M + 1) <sub>Y</sub> $\Rightarrow$ I (E) - 1 $\Rightarrow$ E Until (E) < \$0000	IMM8	FB	xoyo	6 + 12 per iteration	—	$\Delta$	—	$\Delta$	—	—	—	—
ROL	Rotate Left		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0C 1C 2C 170C 171C 172C 173C	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
ROLA	Rotate Left A		INH	370C	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
ROLB	Rotate Left B		INH	371C	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
ROLD	Rotate Left D		INH	27FC	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
ROLE	Rotate Left E		INH	277C	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
ROLW	Rotate Left Word		IND16, X IND16, Y IND16, Z EXT	270C 271C 272C 273C	gggg gggg gggg hh ll	8 8 8 8	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
ROR	Rotate Right Byte		IND8, X IND8, Y IND8, Z IND16, X IND16, Y IND16, Z EXT	0E 1E 2E 170E 171E 172E 173E	ff ff ff gggg gggg gggg hh ll	8 8 8 8 8 8 8	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
RORA	Rotate Right A		INH	370E	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
RORB	Rotate Right B		INH	371E	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
RORD	Rotate Right D		INH	27FE	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$
RORE	Rotate Right E		INH	277E	—	2	—	—	—	—	$\Delta$	$\Delta$	$\Delta$	$\Delta$

## 4.11 Execution Process

Fetches opcodes are latched into stage A, then advanced to stage B. Opcodes are evaluated in stage B. The execution unit can access operands in either stage A or stage B (stage B accesses are limited to 8-bit operands). When execution is complete, opcodes are moved from stage B to stage C, where they remain until the next instruction is complete.

A prefetch mechanism in the microsequencer reads instruction words from memory and increments the program counter. When instruction execution begins, the program counter points to an address six bytes after the address of the first word of the instruction being executed.

The number of machine cycles necessary to complete an execution sequence varies according to the complexity of the instruction. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for details.

### 4.11.1 Changes in Program Flow

When program flow changes, instructions are fetched from a new address. Before execution can begin at the new address, instructions and operands from the previous instruction stream must be removed from the pipeline. If a change in flow is temporary, a return address must be stored, so that execution of the original instruction stream can resume after the change in flow.

When an instruction that causes a change in program flow executes, PK : PC point to the address of the first word of the instruction + \$0006. During execution of the instruction, PK : PC is loaded with the address of the first instruction word in the new instruction stream. However, stages A and B still contain words from the old instruction stream. Extra processing steps must be performed before execution from the new instruction stream.

## 4.12 Instruction Timing

The execution time of CPU16 instructions has three components:

- Bus cycles required to prefetch the next instruction
- Bus cycles required for operand accesses
- Time required for internal operations

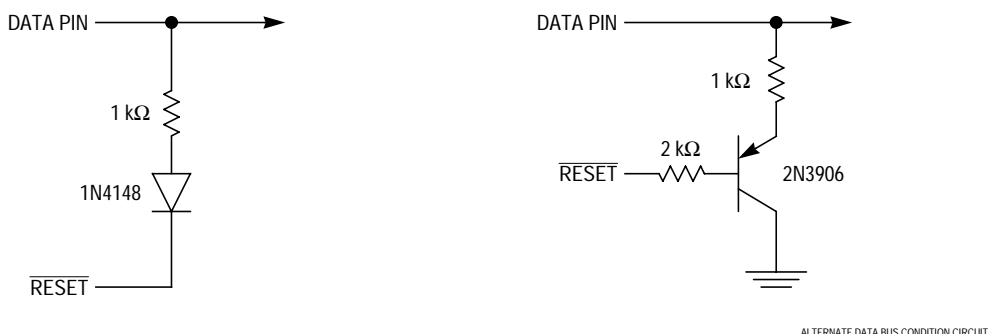
A bus cycle requires a minimum of two system clock periods. If the access time of a memory device is greater than two clock periods, bus cycles are longer. However, all bus cycles must be an integer number of clock periods. CPU16 internal operations are always an integer multiple of two clock periods.

Dynamic bus sizing affects bus cycle time. The integration module manages all accesses. Refer to **SECTION 5 SYSTEM INTEGRATION MODULE** for more information.

The CPU16 does not execute more than one instruction at a time. The total time required to execute a particular instruction stream can be calculated by summing the individual execution times of each instruction in the stream.

The mode configuration drivers are conditioned with  $\overline{R/\overline{W}}$  and  $\overline{DS}$  to prevent conflicts between external devices and the MCU when reset is asserted. If external  $\overline{RESET}$  is asserted during an external write cycle,  $\overline{R/\overline{W}}$  conditioning (as shown in [Figure 5-18](#)) prevents corruption of the data during the write. Similarly,  $\overline{DS}$  conditions the mode configuration drivers so that external reads are not corrupted when  $\overline{RESET}$  is asserted during an external read cycle.

Alternate methods can be used for driving data bus pins low during reset. [Figure 5-19](#) shows two of these options. The simplest is to connect a resistor in series with a diode from the data bus pin to the  $\overline{RESET}$  line. A bipolar transistor can be used for the same purpose, but an additional current limiting resistor must be connected between the base of the transistor and the  $\overline{RESET}$  pin. If a MOSFET is substituted for the bipolar transistor, only the 1 k $\Omega$  isolation resistor is required. These simpler circuits do not offer the protection from potential memory corruption during  $\overline{RESET}$  assertion as does the circuit shown in [Figure 5-18](#).



**Figure 5-19 Alternate Circuit for Data Bus Mode Select Conditioning**

Data bus mode select current is specified in [APPENDIX A ELECTRICAL CHARACTERISTICS](#). Do not confuse pin function with pin electrical state. Refer to [5.7.5 Pin State During Reset](#) for more information.

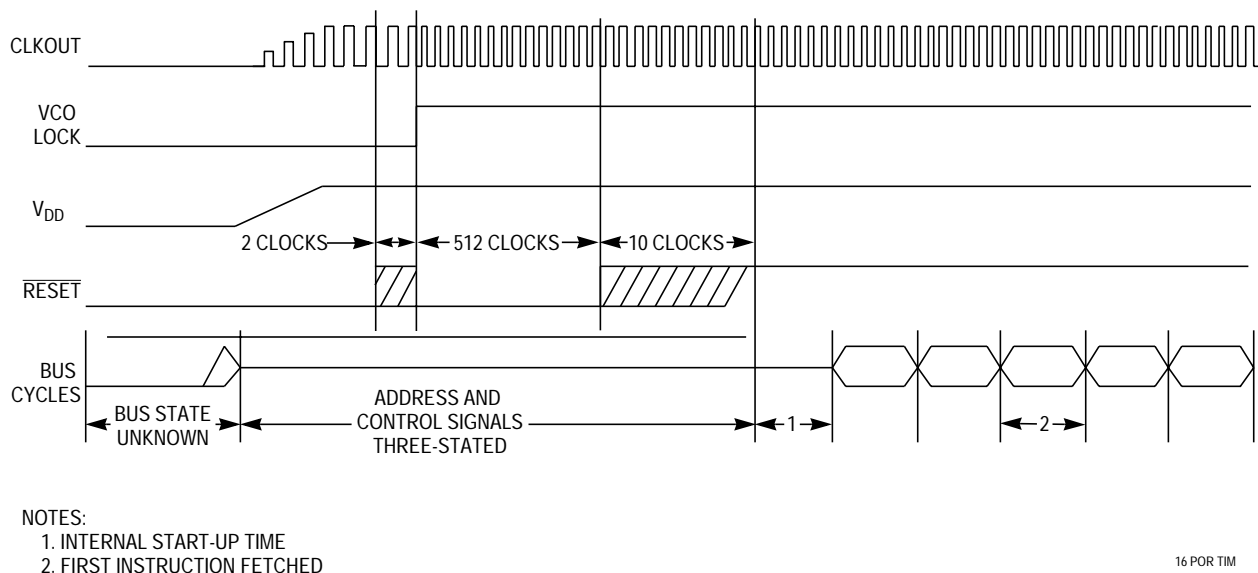
Unlike other chip-select signals, the boot ROM chip-select ( $\overline{CSBOOT}$ ) is active at the release of  $\overline{RESET}$ . During reset exception processing, the MCU fetches initialization vectors beginning at address \$000000 in supervisor program space. An external memory device containing vectors located at these addresses can be enabled by  $\overline{CSBOOT}$  after a reset.

The logic level of DATA0 during reset selects boot ROM port size for dynamic bus allocation. When DATA0 is held low, port size is eight bits; when DATA0 is held high, either by the weak internal pull-up driver or by an external pull-up, port size is 16 bits. Refer to [5.9.4 Chip-Select Reset Operation](#) for more information.

DATA1 and DATA2 determine the functions of  $\overline{CS}[2:0]$  and  $\overline{CS}[5:3]$ , respectively. DATA[7:3] determine the functions of an associated chip-select and all lower-numbered chip-selects down through  $\overline{CS}6$ . For example, if DATA5 is pulled low during reset,  $\overline{CS}[8:6]$  are assigned alternate function as ADDR[21:19], and  $\overline{CS}[10:9]$  remain chip-selects. Refer to [5.9.4 Chip-Select Reset Operation](#) for more information.

The SIM clock synthesizer provides clock signals to the other MCU modules. After the clock is running and MSTRST is asserted for at least four clock cycles, these modules reset.  $V_{DD}$  ramp time and VCO frequency ramp time determine how long the four cycles take. Worst case is approximately 15 milliseconds. During this period, module port pins may be in an indeterminate state. While input-only pins can be put in a known state by external pull-up resistors, external logic on input/output or output-only pins during this time must condition the lines. Active drivers require high-impedance buffers or isolation resistors to prevent conflict.

**Figure 5-20** is a timing diagram for power-on reset. It shows the relationships between  $\overline{\text{RESET}}$ ,  $V_{DD}$ , and bus signals.



**Figure 5-20 Power-On Reset**

### 5.7.8 Use of the Three-State Control Pin

Asserting the three-state control (TSC) input causes the MCU to put all output drivers in a disabled, high-impedance state. The signal must remain asserted for approximately ten clock cycles in order for drivers to change state.

When the internal clock synthesizer is used (MODCLK held high during reset), synthesizer ramp-up time affects how long the ten cycles take. Worst case is approximately 20 milliseconds from TSC assertion.

When an external clock signal is applied (MODCLK held low during reset), pins go to high-impedance state as soon after TSC assertion as approximately ten clock pulses have been applied to the EXTAL pin.

Interrupt requests are sampled on consecutive falling edges of the system clock. Interrupt request input circuitry has hysteresis. To be valid, a request signal must be asserted for at least two consecutive clock periods. Valid requests do not cause immediate exception processing, but are left pending. Pending requests are processed at instruction boundaries or when exception processing of higher-priority interrupts is complete.

The CPU16 does not latch the priority of a pending interrupt request. If an interrupt source of higher priority makes a service request while a lower priority request is pending, the higher priority request is serviced. If an interrupt request with a priority equal to or lower than the current IP mask value is made, the CPU16 does not recognize the occurrence of the request. If simultaneous interrupt requests of different priorities are made, and both have a priority greater than the mask value, the CPU16 recognizes the higher-level request.

### 5.8.3 Interrupt Acknowledge and Arbitration

When the CPU16 detects one or more interrupt requests of a priority higher than the interrupt priority mask value, it places the interrupt request level on the address bus and initiates a CPU space read cycle. The request level serves two purposes: it is decoded by modules or external devices that have requested interrupt service, to determine whether the current interrupt acknowledge cycle pertains to them, and it is latched into the interrupt priority mask field in the CPU16 condition code register to preclude further interrupts of lower priority during interrupt service.

Modules or external devices that have requested interrupt service must decode the IP mask value placed on the address bus during the interrupt acknowledge cycle and respond if the priority of the service request corresponds to the mask value. However, before modules or external devices respond, interrupt arbitration takes place.

Arbitration is performed by means of serial contention between values stored in individual module interrupt arbitration (IARB) fields. Each module that can make an interrupt service request, including the SIM, has an IARB field in its configuration register. IARB fields can be assigned values from %0000 to %1111. In order to implement an arbitration scheme, each module that can request interrupt service must be assigned a unique, non-zero IARB field value during system initialization. Arbitration priorities range from %0001 (lowest) to %1111 (highest). If the CPU16 recognizes an interrupt service request from a source that has an IARB field value of %0000, a spurious interrupt exception is processed.

#### WARNING

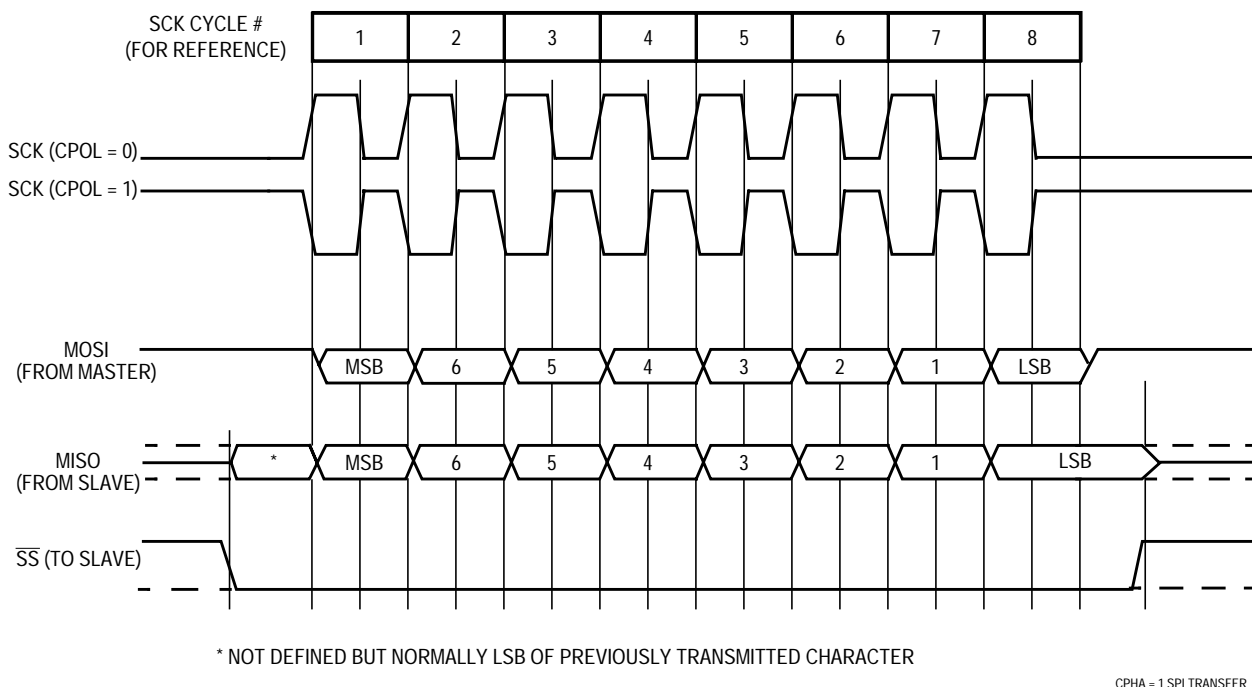
Do not assign the same arbitration priority to more than one module. When two or more IARB fields have the same nonzero value, the CPU16 interprets multiple vector numbers at the same time, with unpredictable consequences.

Because the EBI manages external interrupt requests, the SIM IARB value is used for arbitration between internal and external interrupt requests. The reset value of IARB for the SIM is %1111, and the reset IARB value for all other modules is %0000.



### 10.3.4.2 CPHA = 1 Transfer Format

**Figure 10-4** is a timing diagram of an 8-bit, MSB-first SPI transfer in which CPHA equals one. Two waveforms are shown for SCK, one for CPOL equal to zero and another for CPOL equal to one. The diagram may be interpreted as a master or slave timing diagram since the SCK, MISO and MOSI pins are directly connected between the master and the slave. The MISO signal shown is the output from the slave and the MOSI signal shown is the output from the master. The  $\overline{SS}$  line is the slave select input to the slave.



**Figure 10-4 CPHA = 1 SPI Transfer Format**

For a master, writing to the SPDR initiates the transfer. For a slave, the first edge of SCK indicates the start of a transfer. The SPI is left-shifted on the first and each succeeding odd clock edge, and data is latched on the second and succeeding even clock edges.

SCK is inactive for the last half of the eighth SCK cycle. For a master, SPIF is set at the end of the eighth SCK cycle (after the seventeenth SCK edge). Since the last SCK edge occurs in the middle of the eighth SCK cycle, however, the slave has no way of knowing when the end of the last SCK cycle occurs. The slave therefore considers the transfer complete after the last bit of serial data has been sampled, which corresponds to the middle of the eighth SCK cycle.

When CPHA is one, the  $\overline{SS}$  line may remain at its active low level between transfers. This format is sometimes preferred in systems having a single fixed master and only one slave that needs to drive the MISO data line.

### 11.8.3.1 Output Compare 1

Output compare 1 can affect any or all of OC[5:1] when an output match occurs. In addition to allowing generation of multiple control signals from a single comparison operation, this function makes it possible for two or more output compare functions to control the state of a single OC pin. Output pulses as short as one timer count can be generated in this way.

The OC1 action mask register (OC1M) and the OC1 action data register (OC1D) control OC1 function. Setting a bit in OC1M selects a corresponding bit in the GPT parallel data port. Bits in OC1D determine whether selected bits are to be set or cleared when an OC1 match occurs. Pins must be configured as outputs in order for the data in the register to be driven out on the corresponding pin. If an OC1 match and another output match occur at the same time and both attempt to alter the same pin, the OC1 function controls the state of the pin.

### 11.8.3.2 Forced Output Compare

Timer compare force register (CFORC) is used to make forced compares. The action taken as a result of a forced compare is the same as when an output compare match occurs, except that status flags are not set. Forced channels take programmed actions immediately after the write to CFORC.

The CFORC register is implemented as the upper byte of a 16-bit register which also contains the PWM control register C (PWMC). It can be accessed as eight bits or a word access can be used. Reads of force compare bits (FOC) have no meaning and always return zeros. These bits are self-negating.

## 11.9 Input Capture 4/Output Compare 5

The IC4/OC5 pin can be used for input capture, output compare, or general-purpose I/O. A function enable bit (I4/O5) in the pulse accumulator control register (PACTL) configures the pin for input capture (IC4) or output compare function (OC5). Both bits are cleared during reset, configuring the pin as an input, but also enabling the OC5 function. IC4/OC5 I/O functions are controlled by DDGP7 in the port GP data direction register (DDRGP).

The 16-bit register (TI4/O5) used with the IC4/OC5 function acts as an input capture register or as an output compare register depending on which function is selected. When used as the input capture 4 register, it cannot be written to except in test or freeze mode.

## 11.10 Pulse Accumulator

The pulse accumulator counter (PACNT) is an 8-bit read/write up-counter. PACNT can operate in external event counting or gated time accumulation modes. **Figure 11-5** is a block diagram of the pulse accumulator.

### 11.11.1 PWM Counter

The 16-bit counter in the PWM unit is similar to the timer counter in the capture/compare unit. During reset, the GPT is configured to use the system clock divided by two to drive the counter. Initialization software can reconfigure the counter to use one of seven prescaler outputs or an external clock input from the PCLK pin.

The PWM count register (PWMCNT) can be read at any time without affecting its value. A read must be a word access to ensure coherence, but byte accesses can be made if coherence is not needed. The counter is cleared to \$0000 during reset and is a read-only register except in freeze or test mode.

Fifteen of the sixteen counter bits are output to multiplexers A and B. The multiplexers provide the fast and slow modes of the PWM unit. Mode for PWMA is selected by the SFA bit in the PWM control register C (PWMC). Mode for PWMB is selected by the SFB bit in the same register.

PWMA, PWMB, and PPR[2:0] bits in PWMC control PWM output frequency. In fast mode, bits [7:0] of PWMCNT are used to clock the PWM logic; in slow mode, bits [14:7] are used. The period of a PWM output in slow mode is 128 times longer than the fast mode period. **Table 11-3** shows a range of PWM output frequencies using 16.78 MHz, 20.97 MHz, and 25.17 MHz system clocks.

**Table 11-3 PWM Frequency Ranges**

PPR [2:0]	Prescaler Tap			SFA/B = 0			SFA/B = 1		
	16.78 MHz	20.97 MHz	25.17 MHz	16.78 MHz	20.97 MHz	25.17 MHz	16.78 MHz	20.97 MHz	25.17 MHz
000	Div 2 = 8.39 MHz	Div 2 = 10.5 MHz	Div 2 = 12.6 MHz	32.8 kHz	41 kHz	49.2 kHz	256 Hz	320 Hz	384 Hz
001	Div 4 = 4.19 MHz	Div 4 = 5.25 MHz	Div 4 = 6.29 MHz	16.4 kHz	20.5 kHz	24.6 kHz	128 Hz	160 Hz	192 Hz
010	Div 8 = 2.10 MHz	Div 8 = 2.62 MHz	Div 8 = 3.15 MHz	8.19 kHz	10.2 kHz	12.3 kHz	64.0 Hz	80.0 Hz	96 Hz
011	Div 16 = 1.05 MHz	Div 16 = 1.31 MHz	Div 16 = 1.57 MHz	4.09 kHz	5.15 kHz	6.13 kHz	32.0 Hz	40.0 Hz	48 Hz
100	Div 32 = 524 kHz	Div 32 = 655 kHz	Div 32 = 787 kHz	2.05 kHz	2.56 kHz	3.07 kHz	16.0 Hz	20.0 Hz	24 Hz
101	Div 64 = 262 kHz	Div 64 = 328 kHz	Div 64 = 393 kHz	1.02 kHz	1.28 kHz	1.54 kHz	8.0 Hz	10.0 Hz	12 Hz
110	Div 128 = 131 kHz	Div 128 = 164 kHz	Div 128 = 197 kHz	512 Hz	641 Hz	770 Hz	4.0 Hz	5.0 Hz	6 Hz
111	PCLK	PCLK	PCLK	PCLK/256	PCLK/256	PCLK/256	PCLK/ 32768	PCLK/ 32768	PCLK/ 32768

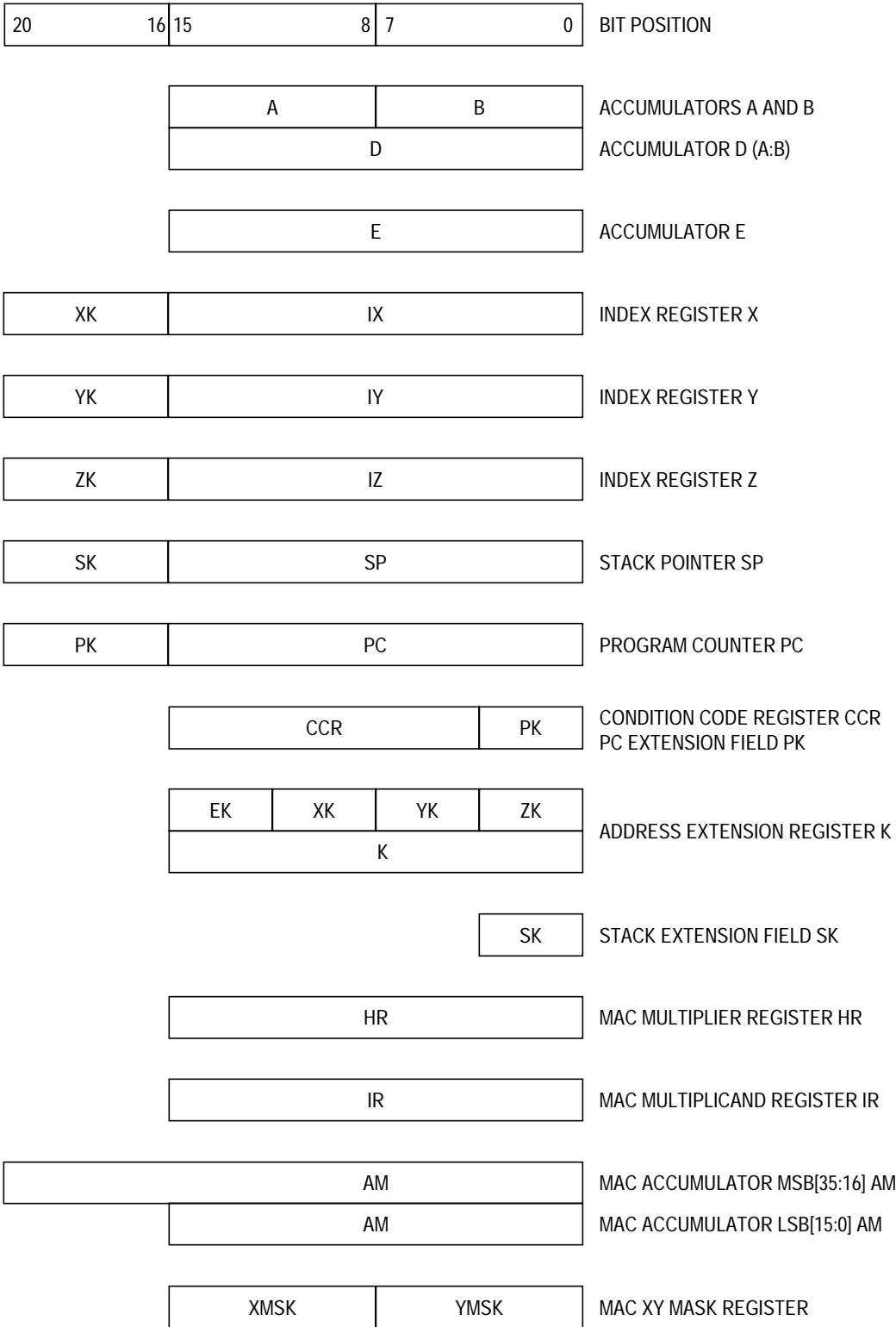
### 11.11.2 PWM Function

The pulse width values of the PWM outputs are determined by control registers PWMA and PWMB. PWMA and PWMB are 8-bit registers implemented as two bytes of a 16-bit register. PWMA and PWMB can be accessed as separate bytes or as one 16-bit register. A value of \$00 loaded into either register causes the corresponding output pin to output a continuous logic level zero signal. A value of \$80 causes the corresponding output signal to have a 50% duty cycle, and so on, to the maximum value of \$FF, which corresponds to an output which is at logic level one for 255/256 of the cycle.

Setting the F1A (for PWMA) or F1B (for PWMB) bits in the CFORC register causes the corresponding pin to output a continuous logic level one signal. The logic level of the associated pin does not change until the end of the current cycle. F1A and F1B are the lower two bits of CFORC, but can be accessed at the same word address as PWMC.

**Table A-5 Typical Ratings, 25.17-MHz**

Num	Rating	Symbol	Value	Unit
1	Supply Voltage	$V_{DD}$	5.0	V
2	Operating Temperature	$T_A$	25	°C
3	$V_{DD}$ Supply Current RUN LPSTOP, VCO off LPSTOP, External clock, max $f_{sys}$	$I_{DD}$	110 125 3.75	mA μA mA
4	Clock Synthesizer Operating Voltage	$V_{DDSYN}$	5.0	V
5	$V_{DDSYN}$ Supply Current VCO on, maximum $f_{sys}$ External Clock, maximum $f_{sys}$ LPSTOP, VCO off $V_{DD}$ powered down	$I_{DDSYN}$	1.0 5.0 100 50	mA mA μA μA
6	RAM Standby Voltage	$V_{SB}$	5.0	V
7	RAM Standby Current Normal RAM operation Standby operation	$I_{SB}$	1.0 1.0	μA μA
8	Power Dissipation	$P_D$	555	mW



CPU16 REGISTER MODEL

Figure D-1 CPU16 Register Model

**Table D-27 Prescaler Output**

PRS[4:0]	ADC Clock	Minimum System Clock	Maximum System Clock
%00000	Reserved	—	—
%00001	System Clock/4	2.0 MHz	8.4 MHz
%00010	System Clock/6	3.0 MHz	12.6 MHz
%00011	System Clock/8	4.0 MHz	16.8 MHz
...	...	...	...
%11101	System Clock/60	30.0 MHz	—
%11110	System Clock/62	31.0 MHz	—
%11111	System Clock/64	32.0 MHz	—

### D.5.5 ADC Control Register 1

#### ADCTL1 — ADC Control Register 1

**\$YFF70C**

15	7	6	5	4	3	2	1	0
NOT USED		SCAN	MULT	S8CM	CD	CC	CB	CA
RESET:								
		0	0	0	0	0	0	0

ADCTL1 is used to initiate an A/D conversion and to select conversion modes and a conversion channel or channels. It can be read or written at any time. A write to ADCTL1 initiates a conversion sequence. If a conversion sequence is already in progress, a write to ADCTL1 aborts it and resets the SCF and CCF flags in the ADC status register.

#### SCAN — Scan Mode Selection

0 = Single conversion

1 = Continuous conversions

Length of conversion sequence(s) is determined by S8CM.

#### MULT — Multichannel Conversion

0 = Conversion sequence(s) run on a single channel selected by [CD:CA].

1 = Sequential conversions of four or eight channels selected by [CD:CA].

Length of conversion sequence(s) is determined by S8CM.

#### S8CM — Select Eight-Conversion Sequence Mode

0 = Four-conversion sequence

1 = Eight-conversion sequence

This bit determines the number of conversions in a conversion sequence. [Table D-28](#) displays the different ADC conversion modes.

## D.7 Multichannel Communication Interface Module

The MCCI is used only in the MC68HC16Z4 and the MC68CK16Z4. [Table D-37](#) shows the MCCI address map.

**Table D-37 MCCI Address Map**

Address <sup>1</sup>	15	8	7	0
\$YFFC00	MCCI Module Configuration Register (MMCR)			
\$YFFC02	MCCI Test Register (MTEST)			
\$YFFC04	SCI Interrupt Level Register (ILSCI)		MCCI Interrupt Vector Register (MIVR)	
\$YFFC06	SPI Interrupt Level Register (ILSPI)		Not Used	
\$YFFC08	Not Used		MCCI Pin Assignment Register (MPAR)	
\$YFFC0A	Not Used		MCCI Data Direction Register (MDDR)	
\$YFFC0C	Not Used		MCCI Port Data Register (PORTMC)	
\$YFFC0E	Not Used		MCCI Port Pin State Register (PORTMCP)	
\$YFFC10 – \$YFFC16	Not Used			
\$YFFC18	SCIA Control Register 0 (SCCR0A)			
\$YFFC1A	SCIA Control Register 1 (SCCR1A)			
\$YFFC1C	SCIA Status Register (SCSRA)			
\$YFFC1E	SCIA Data Register (SCDRA)			
\$YFFC20 – \$YFFC26	Not Used			
\$YFFC28	SCIB Control Register 0 (SCCR0B)			
\$YFFC2A	SCIB Control Register 1 (SCCR1B)			
\$YFFC2C	SCIB Status Register (SCSRB)			
\$YFFC2E	SCIB Data Register (SCDRB)			
\$YFFC30 – \$YFFC36	Not Used			
\$YFFC38	SPI Control Register (SPCR)			
\$YFFC3A	Not Used			
\$YFFC3C	SPI Status Register (SPSR)			
\$YFFC3E	SPI Data Register (SPDR)			

**NOTES:**

1. Y = M111, where M is the logic state of the module mapping (MM) bit in the SIMCR.

### D.7.1 MCCI Module Configuration Register

#### MMCR — MCCI Module Configuration Register

**\$YFFC00**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STOP	NOT USED							SUPV	NOT USED			IARB[3:0]			

RESET:

0 1 0 0 0 0

MMCR bits enable stop mode, establish the privilege level required to access certain MCCI registers, and determine the arbitration priority of MCCI interrupt requests.

### D.7.13 SPI Control Register

#### SPCR — SPI Control Register

**\$YFFC38**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPIE	SPE	WOMP	MSTR	CPOL	CPHA	LSBF	SIZE	SPBR[7:0]							

RESET:

0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0

The SPCR contains parameters for configuring the SPI. The register can be read or written at any time.

#### SPIE — SPI Interrupt Enable

0 = SPI interrupts disabled.

1 = SPI interrupts enabled.

#### SPE — SPI Enable

0 = SPI is disabled.

1 = SPI is enabled.

#### WOMP — Wired-OR Mode for SPI Pins

0 = Outputs have normal CMOS drivers.

1 = Pins designated for output by MDDR have open-drain drivers, regardless of whether the pins are used as SPI outputs or for general-purpose I/O, and regardless of whether the SPI is enabled.

#### MSTR — Master/Slave Mode Select

0 = SPI is a slave device.

1 = SPI is system master.

#### CPOL — Clock Polarity

0 = The inactive state value of SCK is logic level zero.

1 = The inactive state value of SCK is logic level one.

CPOL is used to determine the inactive state of the serial clock (SCK). It is used with CPHA to produce a desired clock/data relationship between master and slave devices.

#### CPHA — Clock Phase

0 = Data captured on the leading edge of SCK and changed on the trailing edge of SCK.

1 = Data is changed on the leading edge of SCK and captured on the trailing edge of SCK.

CPHA determines which edge of SCK causes data to change and which edge causes data to be captured. CPHA is used with CPOL to produce a desired clock/data relationship between master and slave devices.

#### LSBF — Least Significant Bit First

0 = Serial data transfer starts with LSB.

1 = Serial data transfer starts with MSB.



SIZE — Transfer Data Size

0 = 8-bit data transfer.

1 = 16-bit data transfer.

SPBR[7:0] — Serial Clock Baud Rate

The SPI uses a modulus counter to derive the SCK baud rate from the MCU system clock. Baud rate is selected by writing a value from 2 to 255 into SPBR[7:0].

The following expressions apply to SCK baud rate:

$$\text{SCK Baud Rate} = \frac{f_{\text{sys}}}{2 \times \text{SPBR}[7:0]}$$

or

$$\text{SPBR}[7:0] = \frac{f_{\text{sys}}}{2 \times \text{SCK Baud Rate Desired}}$$

Giving SPBR[7:0] a value of zero or one disables SCK (disable state determined by CPOL). At reset, the SCK baud rate is initialized to one-eighth of the system clock frequency.

#### D.7.14 SPI Status Register

**SPSR — SPI Status Register**

**\$YFFC3C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SPIF	WCOL	0	MODF	0	0	0	0	0	0	0	0	0	0	0	0

RESET:

0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0

SPSR contains information concerning the current serial transmission. Only the SPI can set bits in SPSR. The CPU16 reads SPSR to obtain SPI status information and writes it to clear status flags.

SPIF — SPI Finished Flag

0 = SPI is not finished.

1 = SPI is finished.

WCOL — Write Collision

0 = No attempt to write to the SPDR happened during the serial transfer.

1 = Write collision occurred.

Clearing WCOL is accomplished by reading the SPSR while WCOL is set and then either reading the SPDR prior to SPIF being set, or reading or writing the SPDR after SPIF is set.

## E.1.1 EQUATES.ASM

```
*DESCRIPTION :THIS IS A TABLE OF EQUATES FOR ALL M68HC16 Z-SERIES
*
*          REGISTERS.
*****
***** SIM MODULE REGISTERS *****
SIMMCR EQU $FA00    ;SIM MODULE CONFIGURATION REGISTER
SIMTR  EQU $FA02    ;SYSTEM INTEGRATION TEST REGISTER
SYNCR  EQU $FA04    ;CLOCK SYNTHESIZER CONTROL REGISTER
RSR    EQU $FA07    ;RESET STATUS REGISTER
SIMTRE EQU $FA08    ;SYSTEM INTEGRATION TEST REGISTER (E CLOCK)
PORTE0 EQU $FA11    ;PORTE DATA REGISTER (SAME DATA AS PORTE1)
PORTE1 EQU $FA13    ;PORTE DATA REGISTER (SAME DATA AS PORTE0)
DDRE   EQU $FA15    ;PORTE DATA DIRECTION REGISTER
PEPAR  EQU $FA17    ;PORTE PIN ASSIGNMENT REGISTER
PORTF0 EQU $FA19    ;PORT F DATA REGISTER (SAME DATA AS PORTF1)
PORTF1 EQU $FA1B    ;PORT F DATA REGISTER (SAME DATA AS PORTF0)
DDRF   EQU $FA1D    ;PORT F DATA DIRECTION REGISTER
PFPAR  EQU $FA1F    ;PORT F PIN ASSIGNMENT REGISTER
SYPCR  EQU $FA21    ;SYSTEM PROTECTION CONTROL REGISTER
PICR   EQU $FA22    ;PERIODIC INTERRUPT CONTROL REGISTER
PITR   EQU $FA24    ;PERIODIC INTERRUPT TIMING REGISTER
SWSR   EQU $FA27    ;SOFTWARE SERVICE REGISTER
TSTMSRA EQU $FA30   ;MASTER SHIFT REGISTER A
TSTMSRB EQU $FA32   ;MASTER SHIFT REGISTER B
TSTSC  EQU $FA34   ;TEST MODULE SHIFT COUNT
TSTRC  EQU $FA36   ;TEST MODULE REPETITION COUNT
CREG   EQU $FA38   ;TEST SUBMODULE CONTROL REGISTER
DREG   EQU $FA3A   ;DISTRIBUTED REGISTER
CSPDR  EQU $FA41   ;PORT C DATA REGISTER
CSPAR0 EQU $FA44   ;CHIP-SELECT PIN ASSIGNMENT REGISTER 0
CSPAR1 EQU $FA46   ;CHIP-SELECT PIN ASSIGNMENT REGISTER 1
CSBARBT EQU $FA48  ;CHIP-SELECT BOOT BASE ADDRESS REGISTER
CSORBT EQU $FA4A   ;CHIP-SELECT BOOT OPTION REGISTER
CSBAR0 EQU $FA4C   ;CHIP-SELECT 0 BASE ADDRESS REGISTER
CSOR0  EQU $FA4E   ;CHIP SELECT 0 OPTION REGISTER
CSBAR1 EQU $FA50   ;CHIP-SELECT 1 BASE ADDRESS REGISTER
CSOR1  EQU $FA52   ;CHIP-SELECT 1 OPTION REGISTER
CSBAR2 EQU $FA54   ;CHIP-SELECT 2 BASE ADDRESS REGISTER
CSOR2  EQU $FA56   ;CHIP-SELECT 2 OPTION REGISTER
CSBAR3 EQU $FA58   ;CHIP-SELECT 3 BASE ADDRESS REGISTER
CSOR3  EQU $FA5A   ;CHIP-SELECT 3 OPTION REGISTER
CSBAR4 EQU $FA5C   ;CHIP-SELECT 4 BASE ADDRESS REGISTER
CSOR4  EQU $FA5E   ;CHIP-SELECT 4 OPTION REGISTER
CSBAR5 EQU $FA60   ;CHIP-SELECT 5 BASE ADDRESS REGISTER
CSOR5  EQU $FA62   ;CHIP-SELECT 5 OPTION REGISTER
CSBAR6 EQU $FA64   ;CHIP-SELECT 6 BASE ADDRESS REGISTER
CSOR6  EQU $FA66   ;CHIP-SELECT 6 OPTION REGISTER
CSBAR7 EQU $FA68   ;CHIP-SELECT 7 BASE ADDRESS REGISTER
CSOR7  EQU $FA6A   ;CHIP-SELECT 7 OPTION REGISTER
CSBAR8 EQU $FA6C   ;CHIP-SELECT 8 BASE ADDRESS REGISTER
CSOR8  EQU $FA6E   ;CHIP-SELECT 8 OPTION REGISTER
CSBAR9 EQU $FA70   ;CHIP-SELECT 9 BASE ADDRESS REGISTER
CSOR9  EQU $FA72   ;CHIP-SELECT 9 OPTION REGISTER
CSBAR10 EQU $FA74  ;CHIP-SELECT 10 BASE ADDRESS REGISTER
CSOR10 EQU $FA76   ;CHIP-SELECT 10 OPTION REGISTER
```

```

TR7    EQU $FD2E    ;SPI TXD.RAM 7
TR8    EQU $FD30    ;SPI TXD.RAM 8
TR9    EQU $FD32    ;SPI TXD.RAM 9
TRA    EQU $FD34    ;SPI TXD.RAM A
TRB    EQU $FD36    ;SPI TXD.RAM B
TRC    EQU $FD38    ;SPI TXD.RAM C
TRD    EQU $FD3A    ;SPI TXD.RAM D
TRE    EQU $FD3C    ;SPI TXD.RAM E
TRF    EQU $FD3E    ;SPI TXD.RAM F
CR0    EQU $FD40    ;SPI CMD.RAM 0
CR1    EQU $FD41    ;SPI CMD.RAM 1
CR2    EQU $FD42    ;SPI CMD.RAM 2
CR3    EQU $FD43    ;SPI CMD.RAM 3
CR4    EQU $FD44    ;SPI CMD.RAM 4
CR5    EQU $FD45    ;SPI CMD.RAM 5
CR6    EQU $FD46    ;SPI CMD.RAM 6
CR7    EQU $FD47    ;SPI CMD.RAM 7
CR8    EQU $FD48    ;SPI CMD.RAM 8
CR9    EQU $FD49    ;SPI CMD.RAM 9
CRA    EQU $FD4A    ;SPI CMD.RAM A
CRB    EQU $FD4B    ;SPI CMD.RAM B
CRC    EQU $FD4C    ;SPI CMD.RAM C
CRD    EQU $FD4D    ;SPI CMD.RAM D
CRE    EQU $FD4E    ;SPI CMD.RAM E
CRF    EQU $FD4F    ;SPI CMD.RAM F

```

\*\*\*\*\* MCCI MODULE REGISTERS \*\*\*\*\*

```

MMCR    EQU $FC00    ;MCCI MODULE CONFIGURATION REGISTER
MTEST    EQU $FC02    ;MCCI TEST REGISTER
ILSCI    EQU $FC04    ;SCI INTERRUPT LEVEL REGISTER
MIVR    EQU $FC05    ;MCCI INTERRUPT VECTOR REGISTER
ILSPI    EQU $FC06    ;SPI INTERRUPT LEVEL REGISTER
MPAR    EQU $FC09    ;MCCI PIN ASSIGNMENT REGISTER
MDDR    EQU $FC0B    ;MCCI DATA DIRECTION REGISTER
PORTMC    EQU $FC0D    ;MCCI PORT DATA REGISTER
PORTMCP    EQU $FC0F    ;MCCI PORT PIN STATE REGISTER
SCCR0A    EQU $FC18    ;SCIA CONTROL REGISTER 0
SCCR1A    EQU $FC1A    ;SCIA CONTROL REGISTER 1
SCSRA    EQU $FC1C    ;SCIA STATUS REGISTER
SCDRA    EQU $FC1E    ;SCIA DATA REGISTER
SCCR0B    EQU $FC28    ;SCIB CONTROL REGISTER 0
SCCR1B    EQU $FC2A    ;SCIB CONTROL REGISTER 1
SCSRB    EQU $FC2C    ;SCIB STATUS REGISTER
SCDRB    EQU $FC2E    ;SCIB DATA REGISTER
SPCR    EQU $FC38    ;SPI CONTROL REGISTER
SPSR    EQU $FC3C    ;SPI STATUS REGISTER
SPDR    EQU $FC3E    ;SPI DATA REGISTER

```

\*\*\*\*\* GPT MODULE REGISTERS \*\*\*\*\*

```

GPTMCR    EQU $F900    ;GPT MODULE CONFIGURATION REGISTER
GPTMTR    EQU $F902    ;GPT MODULE TEST REGISTER (RESERVED)
ICR    EQU $F904    ;GPT INTERRUPT CONFIGURATION REGISTER
PDDR    EQU $F906    ;PARALLEL DATA DIRECTION REGISTER
GPTPDR    EQU $F907    ;PARALLEL DATA REGISTER
OC1M    EQU $F908    ;OC1 ACTION MASK REGISTER
OC1D    EQU $F909    ;OC1 ACTION DATA REGISTER
TCNT    EQU $F90A    ;TIMER COUNTER REGISTER

```

```

*           has overflowed ten times.
*
*****

INCLUDE      'EQUATES.ASM'      ;table of EQUates for common register
                                   ;addresses
INCLUDE      'ORG00000.ASM'      ;initialize reset vector
INCLUDE      'ORG00008.ASM'      ;initialize interrupt vectors

*
*   We are choosing User Defined Interrupt Vector 9 (interrupt vector 64
*   at address $0080) to be the base vector number (VBA) for the GPT
*   because the least significant nibble in the address must be a $0.
*
*   The VBA should be reflected in the GPT Interrupt Configuration
*   Register (ICR) at $YFF904.

ORG          $0080                ;Address for interrupt vector 64

DC.W         PAOV_ROUTINE          ;Adjusted Priority Channel -- PAC
DC.W         IC1_ROUTINE           ;Input Capture 1
DC.W         IC2_ROUTINE           ;Input Capture 2
DC.W         IC3_ROUTINE           ;Input Capture 3
DC.W         BDM                   ;Output Compare 1
DC.W         OC2_ROUTINE           ;Output Compare 2
DC.W         BDM                   ;Output Compare 3
DC.W         BDM                   ;Output Compare 4
DC.W         BDM                   ;Input Capture 4 / Output Compare 5
DC.W         BDM                   ;Timer Overflow
DC.W         BDM                   ;Pulse Accumulator Overflow -- elevated
DC.W         BDM                   ;Pulse Accumulator Input

ORG          $0200                ;start program after interrupt vectors

***** Initialization Routines *****

INCLUDE      'INITSYS.ASM'         ;initially set EK=F, XK=0, YK=0, ZK=0
                                   ;set sys clock at 16.78 MHz, disable
COP
INCLUDE      'INITRAM.ASM'         ;turn on 1k internal SRAM at $10000
                                   ;set stack in bank 1 (SK=1, SP=03FE)
INCLUDE      'INITSCI.ASM'         ;set SCI baud rate at 9600
                                   ;enable SCI transmitter and receiver

*
*   Set up the interrupts

LDD          #$008E                ;Give the GPT an IARB of $E
STD          GPTMCR                ;so we can generate interrupts
LDD          #$A640                ;elevate interrupt priority of PAOV,
STD          ICR                   ;set GPT IRQ level to 6,
                                   ;& assign vector 64 (User vector 9) of the
                                   ;interrupt/exception vector table as the
                                   ;GPT's Interrupt Vector Base Address
LDAB         #$17                  ;set OC2, IC1, IC2, IC3 to generate interrupts
STAB         TMSK1
LDAB         #$25                  ;set PAC Overflows to generate interrupts

```



**Freescale Semiconductor, Inc.**

**Freescale Semiconductor, Inc.**

**For More Information On This Product,  
Go to: [www.freescale.com](http://www.freescale.com)**