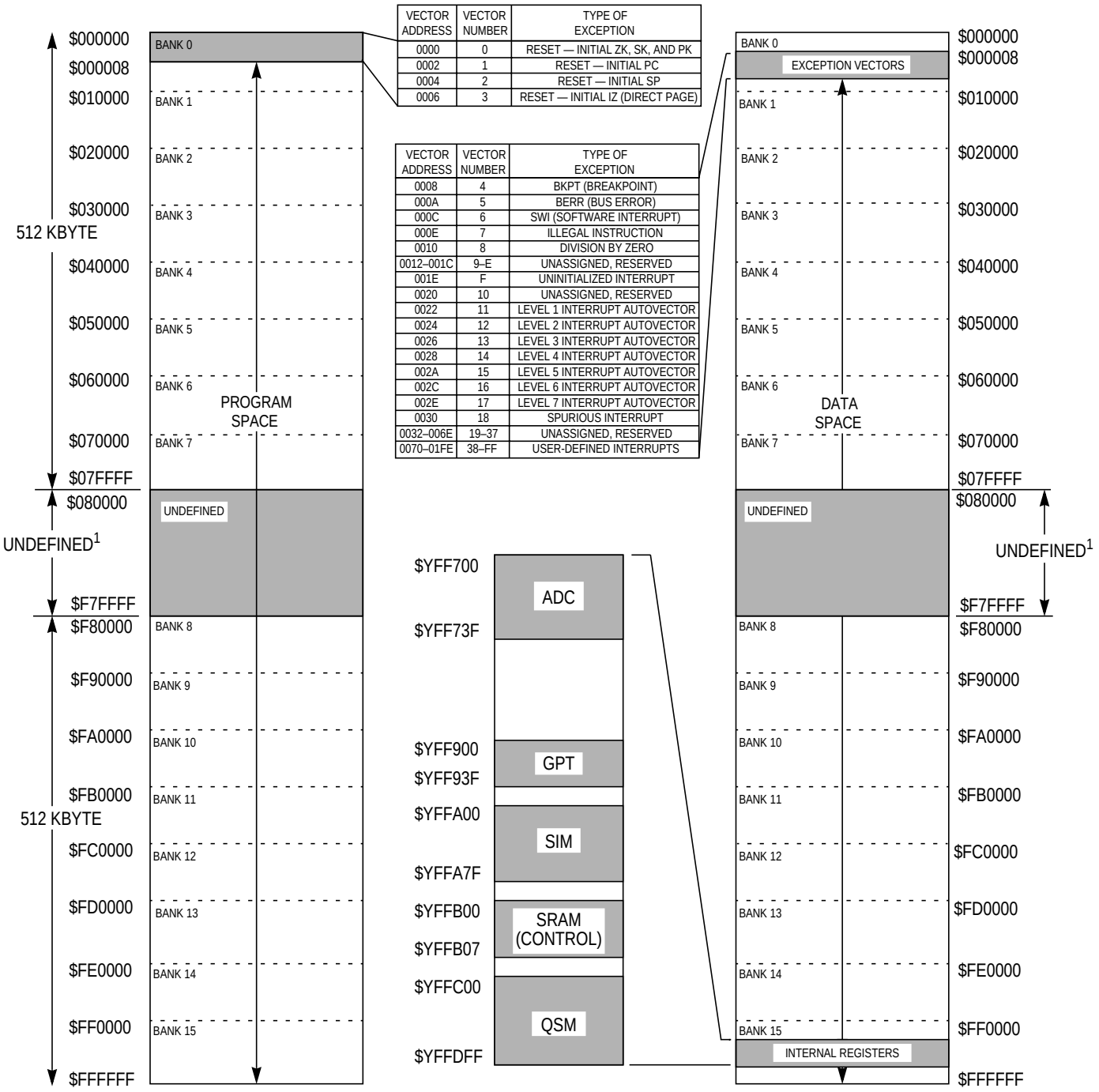
Welcome to E-XFL.COM

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	Obsolete
Core Processor	CPU16
Core Size	16-Bit
Speed	16MHz
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	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc68hc16z1cpv16



NOTE:
1. THE ADDRESSES DISPLAYED IN THIS MEMORY MAP ARE THE FULL 24-BIT IMB ADDRESSES. THE CPU16 ADDRESS BUS IS 20 BITS WIDE, AND CPU16 ADDRESS LINE 19 DRIVES IMB ADDRESS LINES [23:20]. THE BLOCK OF ADDRESSES FROM $080000 TO $F7FFFF MARKED AS UNDEFINED WILL NEVER APPEAR ON THE IMB. MEMORY BANKS 0 TO 15 APPEAR FULLY CONTIGUOUS IN THE CPU16'S FLAT 20-BIT ADDRESS SPACE. THE CPU16 NEED ONLY GENERATE A 20-BIT EFFECTIVE ADDRESS TO ACCESS ANY LOCATION IN THIS RANGE.

HC16Z1/CK/CM MEM MAP (S)

Figure 3-14 MC68HC16Z1/CKZ1/CMZ1 Separate Program and Data Space Map

4.9 Instruction Format

CPU16 instructions consist of an 8-bit opcode that can be preceded by an 8-bit prebyte and followed by one or more operands.

Opcodes are mapped in four 256-instruction pages. Page 0 opcodes stand alone. Page 1, 2, and 3 opcodes are pointed to by a prebyte code on page 0. The prebytes are $17 (page 1), $27 (page 2), and $37 (page 3).

Operands can be four bits, eight bits or sixteen bits in length. Since the CPU16 fetches 16-bit instruction words from even-byte boundaries, each instruction must contain an even number of bytes.

Operands are organized as bytes, words, or a combination of bytes and words. Operands of four bits are either zero-extended to eight bits, or packed two to a byte. The largest instructions are six bytes in length. Size, order, and function of operands are evaluated when an instruction is decoded.

A page 0 opcode and an 8-bit operand can be fetched simultaneously. Instructions that use 8-bit indexed, immediate, and relative addressing modes have this form. Code written with these instructions is very compact.

Figure 4-4 shows basic CPU16 instruction formats.

of $\overline{\text{BKPT}}$ or execution of the BGND instruction. IPIPE0 and IPIPE1 change function before an exception signal can be generated. The development system must use FREEZE assertion as an indication that BDM has been entered. When BDM is exited, FREEZE is negated before initiation of normal bus cycles. IPIPE0 and IPIPE1 are valid when normal instruction prefetch begins.

4.14.4.4 BDM Commands

Commands consist of one 16-bit operation word and can include one or more 16-bit extension words. Each incoming word is read as it is assembled by the serial interface. The microcode routine corresponding to a command is executed as soon as the command is complete. Result operands are loaded into the output shift register to be shifted out as the next command is read. This process is repeated for each command until the CPU returns to normal operating mode. The BDM command set is summarized in **Table 4-7**. Refer to the *CPU16 Reference Manual* (CPU16RM/AD) for a BDM command glossary.

Table 4-7 Command Summary

Command	Mnemonic	Description
Read Registers from Mask	RREGM	Read contents of registers specified by command word register mask
Write Registers from Mask	WREGM	Write to registers specified by command word register mask
Read MAC Registers	RDMAC	Read contents of entire multiply and accumulate register set
Write MAC Registers	WRMAC	Write to entire multiply and accumulate register set
Read PC and SP	RPCSP	Read contents of program counter and stack pointer
Write PC and SP	WPCSP	Write to program counter and stack pointer
Read Data Memory	RDMEM	Read byte from specified 20-bit address in data space
Write Data Memory	WDMEM	Write byte to specified 20-bit address in data space
Read Program Memory	RPMEM	Read word from specified 20-bit address in program space
Write Program Memory	WPMEM	Write word to specified 20-bit address in program space
Execute from Current PK : PC	GO	Instruction pipeline flushed and refilled; instructions executed from current PC – $0006
Null Operation	NOP	Null command performs no operation

4.14.4.5 Returning from BDM

BDM is terminated when a resume execution (GO) command is received. GO refills the instruction pipeline from address (PK : PC – $0006). FREEZE is negated before the first prefetch. Upon negation of FREEZE, the BDM serial subsystem is disabled and the DSO/DSI signals revert to IPIPE0/IPIPE1 functionality.

The external bus has 24 address lines and 16 data lines. ADDR[19:0] are normal address outputs; ADDR[23:20] follow the output state of ADDR19. The EBI provides dynamic sizing between 8-bit and 16-bit data accesses. It supports byte, word, and long-word transfers. Port width is the maximum number of bits accepted or provided by the external memory system during a bus transfer. Widths of eight and sixteen bits are accessed through the use of asynchronous cycles controlled by the size (SIZ1 and SIZ0) and data size acknowledge ($\overline{\text{DSACK1}}$ and $\overline{\text{DSACK0}}$) pins. Multiple bus cycles may be required for dynamically sized transfers.

To add flexibility and minimize the necessity for external logic, MCU chip-select logic is synchronized with EBI transfers. Refer to **5.9 Chip-Selects** for more information.

5.5.1 Bus Control Signals

The address bus provides addressing information to external devices. The data bus transfers 8-bit and 16-bit data between the MCU and external devices. Strobe signals, one for the address bus and another for the data bus, indicate the validity of an address and provide timing information for data.

Control signals indicate the beginning of each bus cycle, the address space, the size of the transfer, and the type of cycle. External devices decode these signals and respond to transfer data and terminate the bus cycle. The EBI can operate in an asynchronous mode for any port width.

5.5.1.1 Address Bus

Bus signals ADDR[19:0] define the address of the byte (or the most significant byte) to be transferred during a bus cycle. The MCU places the address on the bus at the beginning of a bus cycle. The address is valid while $\overline{\text{AS}}$ is asserted.

5.5.1.2 Address Strobe

Address strobe ($\overline{\text{AS}}$) is a timing signal that indicates the validity of an address on the address bus and of many control signals.

5.5.1.3 Data Bus

Signals DATA[15:0] form a bidirectional, non-multiplexed parallel bus that transfers data to or from the MCU. A read or write operation can transfer eight or sixteen bits of data in one bus cycle. For a write cycle, all sixteen bits of the data bus are driven, regardless of the port width or operand size.

5.5.1.4 Data Strobe

Data strobe ($\overline{\text{DS}}$) is a timing signal. For a read cycle, the MCU asserts $\overline{\text{DS}}$ to signal an external device to place data on the bus. $\overline{\text{DS}}$ is asserted at the same time as $\overline{\text{AS}}$ during a read cycle. For a write cycle, $\overline{\text{DS}}$ signals an external device that data on the bus is valid.

4. Set the BYTE field to lower byte when using a 16-bit port, as the external vector for a 16-bit port is fetched from the lower byte. Set the BYTE field to upper byte when using an 8-bit port.

If an interrupting device does not provide a vector number, an autovector acknowledge must be generated, either by asserting the AVEC pin or by generating AVEC internally using the chip-select option register. This terminates the bus cycle.

5.9.4 Chip-Select Reset Operation

The least significant bit of each of the 2-bit chip-select pin assignment fields in CSPAR0 and CSPAR1 each have a reset value of one. The reset values of the most significant bits of each field are determined by the states of DATA[7:1] during reset. There are weak internal pull-up drivers for each of the data lines so that chip-select operation is selected by default out of reset. However, the internal pull-up drivers can be overcome by bus loading effects.

To ensure a particular configuration out of reset, use an active device to put the data lines in a known state during reset. The base address fields in chip-select base address registers CSBAR[0:10] and chip-select option registers CSOR[0:10] have the reset values shown in **Table 5-25**. The BYTE fields of CSOR[0:10] have a reset value of “disable”, so that a chip-select signal cannot be asserted until the base and option registers are initialized.

Table 5-25 Chip-Select Base and Option Register Reset Values

Fields	Reset Values
Base address	$000000
Block size	2 Kbyte
Async/sync mode	Asynchronous mode
Upper/lower byte	Disabled
Read/write	Disabled
AS/DS	AS
DSACK	No wait states
Address space	CPU space
IPL	Any level
Autovector	External interrupt vector

Following reset, the MCU fetches the initial stack pointer and program counter values from the exception vector table, beginning at $000000 in supervisor program space. The CSBOOT chip-select signal is used to select an external boot device mapped to a base address of $000000.

The MSB of the CSBTPA field in CSPAR0 has a reset value of one, so that chip-select function is selected by default out of reset. The BYTE field in chip-select option register CSORBT has a reset value of “both bytes” so that the select signal is enabled out of reset. The LSB of the CSBOOT field, determined by the logic level of DATA0 during reset, selects the boot ROM port size. When DATA0 is held low during reset, port size is eight bits. When DATA0 is held high during reset, port size is 16 bits. DATA0 has a weak internal pull-up driver, so that a 16-bit port is selected by default

9.3.2.3 Command RAM

Command RAM is used by the QSPI in master mode. The CPU16 writes one byte of control information to this segment for each QSPI command to be executed. The QSPI cannot modify information in command RAM.

Command RAM consists of 16 bytes. Each byte is divided into two fields. The peripheral chip-select field enables peripherals for transfer. The command control field provides transfer options.

A maximum of 16 commands can be in the queue. Queue execution by the QSPI proceeds from the address in NEWQP through the address in ENDQP (both of these fields are in SPCR2).

9.3.3 QSPI Pins

The QSPI uses seven pins. These pins can be configured for general-purpose I/O when not needed for QSPI application.

Table 9-2 shows QSPI input and output pins and their functions.

Table 9-2 QSPI Pins

Pin Names	Mnemonics	Mode	Function
Master In Slave Out	MISO	Master Slave	Serial data input to QSPI Serial data output from QSPI
Master Out Slave In	MOSI	Master Slave	Serial data output from QSPI Serial data input to QSPI
Serial Clock	SCK	Master Slave	Clock output from QSPI Clock input to QSPI
Peripheral Chip Selects	PCS[3:1]	Master	Select peripherals
Slave Select	PCS0/$\overline{SS}$	Master Master Slave	Selects peripherals Causes mode fault Initiates serial transfer

9.3.4 QSPI Operation

The QSPI uses a dedicated 80-byte block of static RAM accessible by both the QSPI and the CPU16 to perform queued operations. The RAM is divided into three segments. There are 16 command bytes, 16 transmit data words, and 16 receive data words. QSPI RAM is organized so that one byte of command data, one word of transmit data, and one word of receive data correspond to one queue entry, $0–$F.

The CPU16 initiates QSPI operation by setting up a queue of QSPI commands in command RAM, writing transmit data into transmit RAM, then enabling the QSPI. The QSPI executes the queued commands, sets a completion flag (SPIF), and then either interrupts the CPU16 or waits for intervention.

There are four queue pointers. The CPU16 can access three of them through fields in QSPI registers. The new queue pointer (NEWQP), contained in SPCR2, points to the first command in the queue. An internal queue pointer points to the command currently being executed. The completed queue pointer (CPTQP), contained in SPSR, points to the last command executed. The end queue pointer (ENDQP), contained in SPCR2, points to the final command in the queue.

Data transfer is synchronized with the internally-generated serial clock SCK. Control bits, CPHA and CPOL, in SPCR0, control clock phase and polarity. Combinations of CPHA and CPOL determine upon which SCK edge to drive outgoing data from the MOSI pin and to latch incoming data from the MISO pin.

Baud rate is selected by writing a value from two to 255 into SPBR[7:0] in SPCR0. The QSPI uses a modulus counter to derive the SCK baud rate from the MCU system clock.

The following expressions apply to the SCK baud rate:

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

or

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

Giving SPBR[7:0] a value of zero or one disables the baud rate generator and SCK assumes its inactive state.

The DSCK bit in each command RAM byte inserts either a standard (DSCK = 0) or user-specified (DSCK = 1) delay from chip-select assertion until the leading edge of the serial clock. The DSCKL field in SPCR1 determines the length of the user-defined delay before the assertion of SCK. The following expression determines the actual delay before SCK:

$$\text{PCS to SCK Delay} = \frac{\text{DSCKL[6:0]}}{f_{sys}}$$

where DSCKL[6:0] equals {1, 2, 3,..., 127}.

When DSCK equals zero, DSCKL[6:0] is not used. Instead, the PCS valid-to-SCK transition is one-half the SCK period.

There are two transfer length options. The user can choose a default value of eight bits, or a programmed value from eight to sixteen bits, inclusive. The programmed value must be written into BITS[3:0] in SPCR0. The BITSE bit in each command RAM byte determines whether the default value (BITSE = 0) or the BITS[3:0] value (BITSE = 1) is used. **Table 9-3** shows BITS[3:0] encoding.

QSPI operation is initiated by setting the SPE bit in SPCR1. Shortly after SPE is set, the QSPI executes the command at the command RAM address pointed to by NEWQP. Data at the pointer address in transmit RAM is loaded into the data serializer and transmitted. Data that is simultaneously received is stored at the pointer address in receive RAM.

When the proper number of bits have been transferred, the QSPI stores the working queue pointer value in CPTQP, increments the working queue pointer, and loads the next data for transfer from transmit RAM. The command pointed to by the incremented working queue pointer is executed next, unless a new value has been written to NEWQP. If a new queue pointer value is written while a transfer is in progress, that transfer is completed normally.

When the CONT bit in a command RAM byte is set, PCS pins are continuously driven to specified states during and between transfers. If the chip-select pattern changes during or between transfers, the original pattern is driven until execution of the following transfer begins. When CONT is cleared, the data in register PORTQS is driven between transfers. The data in PORTQS must match the inactive states of SCK and any peripheral chip-selects used.

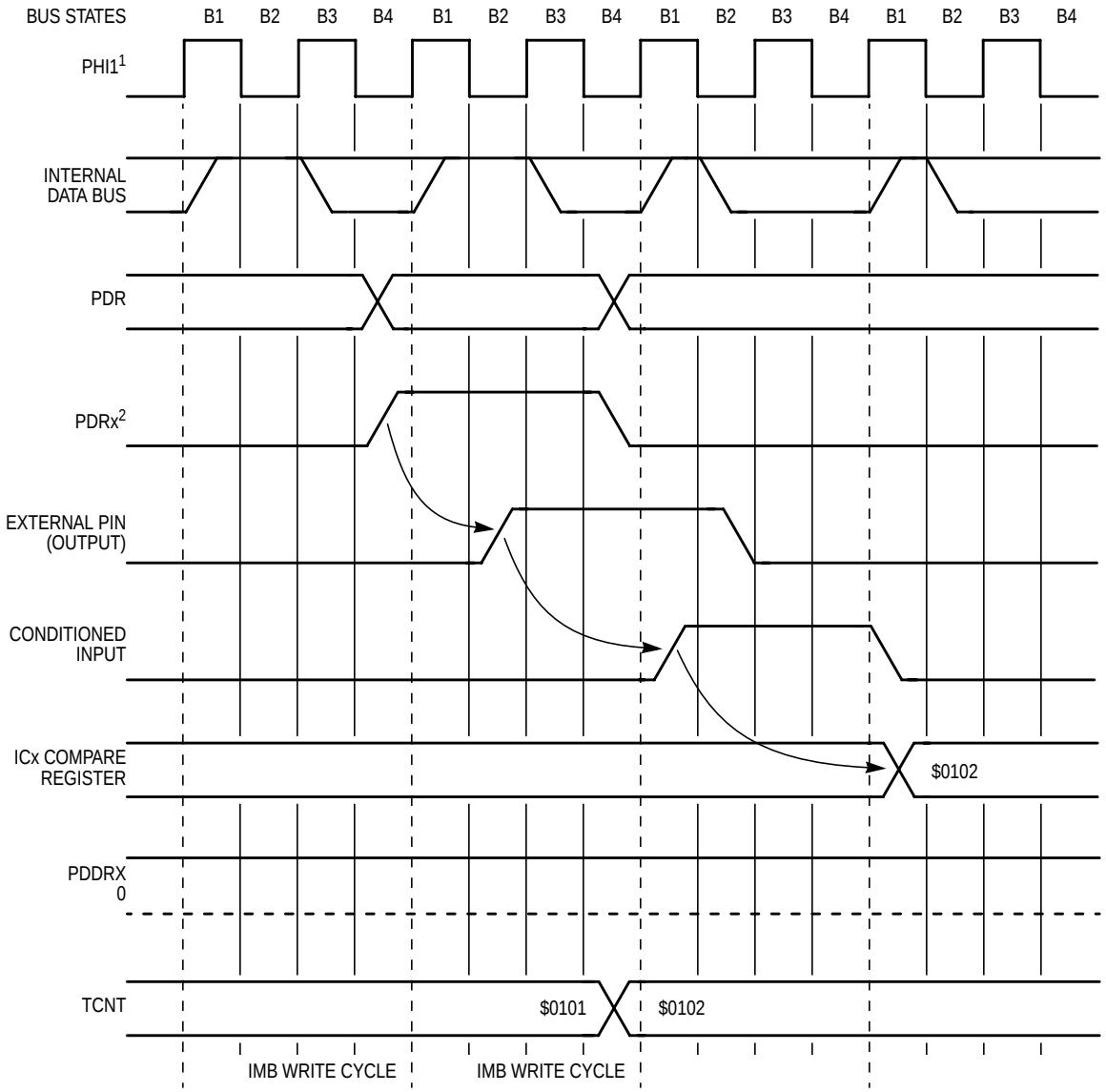
When the QSPI reaches the end of the queue, it sets the SPIF flag. SPIF is set during the final transfer before it is complete. If the SPIFIE bit in SPCR2 is set, an interrupt request is generated when SPIF is asserted. At this point, the QSPI clears SPE and stops unless wrap-around mode is enabled.

9.3.5.2 Master Wrap-Around Mode

Wrap-around mode is enabled by setting the WREN bit in SPCR2. The queue can wrap to pointer address $0 or to the address pointed to by NEWQP, depending on the state of the WRTO bit in SPCR2.

In wrap-around mode, the QSPI cycles through the queue continuously, even while the QSPI is requesting interrupt service. SPE is not cleared when the last command in the queue is executed. New receive data overwrites previously received data in receive RAM. Each time the end of the queue is reached, the SPIF flag is set. SPIF is not automatically reset. If interrupt-driven QSPI service is used, the service routine must clear the SPIF bit to end the current interrupt request. Additional interrupt requests during servicing can be prevented by clearing SPIFIE, but SPIFIE is buffered. Clearing it does not end the current request.

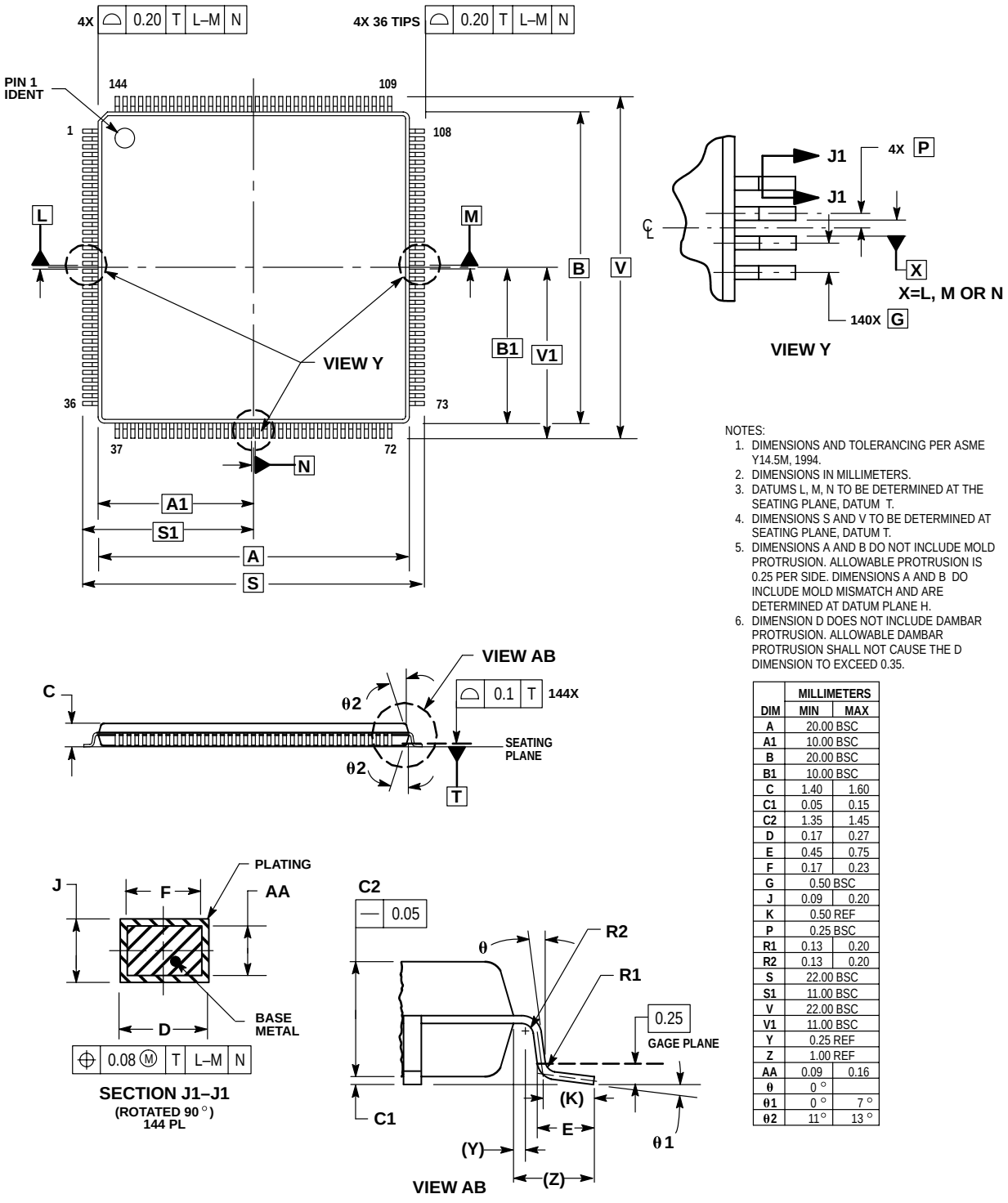
Wrap-around mode is exited by clearing the WREN bit or by setting the HALT bit in SPCR3. Exiting wrap-around mode by clearing SPE is not recommended, as clearing SPE may abort a serial transfer in progress. The QSPI sets SPIF, clears SPE, and stops the first time it reaches the end of the queue after WREN is cleared. After HALT is set, the QSPI finishes the current transfer, then stops executing commands. After the QSPI stops, SPE can be cleared.



NOTES:
1. PHI1 IS THE SAME FREQUENCY AS THE SYSTEM CLOCK; HOWEVER, IT DOES NOT HAVE THE SAME TIMING.
2. WHEN THE BIT VALUE IS DRIVEN ON THE PIN, THE INPUT CIRCUIT SEES THE SIGNAL. AFTER IT IS CONDITIONED, IT CAUSES THE CONTENTS OF THE TCNT TO BE LATCHED INTO THE ICx COMPARE REGISTER.

GENERAL PURPOSE OUTPUT

Figure A-32 General-Purpose Output (Causes Input Capture)



DIM	MILLIMETERS MIN	MAX
A	20.00 BSC	
A1	10.00 BSC	
B	20.00 BSC	
B1	10.00 BSC	
C	1.40	1.60
C1	0.05	0.15
C2	1.35	1.45
D	0.17	0.27
E	0.45	0.75
F	0.17	0.23
G	0.50 BSC	
J	0.09	0.20
K	0.50 REF	
P	0.25 BSC	
R1	0.13	0.20
R2	0.13	0.20
S	22.00 BSC	
S1	11.00 BSC	
V	22.00 BSC	
V1	11.00 BSC	
Y	0.25 REF	
Z	1.00 REF	
AA	0.09	0.16
θ	0 °	
θ1	0 °	7 °
θ2	11°	13 °

Figure B-6 Case 918 — 144-Pin Package Dimensions

Table B-1 M68HC16 Z-Series Ordering Information (Continued)

(Shaded cells indicate preliminary part numbers)

Device	Crystal Input	Operating Voltage	Package Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
MC68HC16Z2 (No ROM)	4 MHz	5 V	144-Pin TQFP	–40 to +85°C	16 MHz	2	SPMCM16Z2BCPV16
						60	MCM16Z2BCPV16
						300	MCM16Z2BCPV16B1
					20 MHz	2	SPMCM16Z2BCPV20
						60	MCM16Z2BCPV20
						300	MCM16Z2BCPV20B1
					25 MHz	2	SPMCM16Z2BCPV25
						60	MCM16Z2BCPV25
						300	MCM16Z2BCPV25B1
				–40 to +105°C	16 MHz	2	SPMCM16Z2BVPV16
						60	MCM16Z2BVPV16
						300	MCM16Z2BVPV16B1
					20 MHz	2	SPMCM16Z2BVPV20
						60	MCM16Z2BVPV20
						300	MCM16Z2BVPV20B1
					25 MHz	2	SPMCM16Z2BVPV25
						60	MCM16Z2BVPV25
						300	MCM16Z2BVPV25B1
				–40 to +125°C	16 MHz	2	SPMCM16Z2BMPV16
						60	MCM16Z2BMPV16
						300	MCM16Z2BMPV16B1
					20 MHz	2	SPMCM16Z2BMPV20
						60	MCM16Z2BMPV20
						300	MCM16Z2BMPV20B1
MC68HC16Z3 (ROM)	4 MHz or 32 kHz	5 V	132-Pin PQFP	–40 to +85°C	16 MHz	2	NA
						36	MC68HC16Z3CFC16
						180	NA
					20 MHz	2	NA
						36	MC68HC16Z3CFC20
						180	NA
					25 MHz	2	NA
						36	MC68HC16Z3CFC25
						180	NA

Table B-1 M68HC16 Z-Series Ordering Information (Continued)

(Shaded cells indicate preliminary part numbers)

Device	Crystal Input	Operating Voltage	Package Type	Temperature	Frequency (MHz)	Package Order Quantity	Order Number
MC68HC16Z4	32 kHz	5 V	144-Pin TQFP	–40 to +85°C	16 MHz	2	SPMCK16Z4CPV16
						60	MCK68HC16Z4CPV16
						300	MCK16Z4CPV16B1
					20 MHz	2	SPMCK16Z4CPV20
						60	MCK68HC16Z4CPV20
						300	MCK16Z4CPV20B1
					25 MHz	2	SPMCK16Z4CPV25
						60	MCK68HC16Z4CPV25
						300	MCK16Z4CPV25B1
				–40 to +105°C	16 MHz	2	SPMCK16Z4VPV16
						60	MCK68HC16Z4VPV16
						300	MCK16Z4VPV16B1
					20 MHz	2	SPMCK16Z4VPV20
						60	MCK68HC16Z4VPV20
						300	MCK16Z4VPV20B1
					25 MHz	2	SPMCK16Z4VPV25
						60	MCK68HC16Z4VPV25
						300	MCK16Z4VPV25B1
				–40 to +125°C	16 MHz	2	SPMCK16Z4MPV16
						60	MCK68HC16Z4MPV16
						300	MCK16Z4MPV16B1
					20 MHz	2	SPMCK16Z4MPV20
						60	MCK68HC16Z4MPV20
						300	MCK16Z4MPV20B1
		2.7 V	132-Pin PQFP	–40 to +85°C	16 MHz	2	SPMCCK16Z4CFC16
						36	MC68CK16Z4CFC16
						180	MCCK16Z4CFC16B1
			144-Pin TQFP	–40 to +85°C	16 MHz	2	SPMCCK16Z4CPV16
						60	MC68CK16Z4CPV16
						300	MCCK16Z4CPV16B1

PADA[7:0] — Port ADA Data Pins

A read of PADA[7:0] returns the logic level of the port ADA pins. If an input is not at an appropriate logic level (that is, outside the defined levels), the read is indeterminate. Use of a port ADA pin for digital input does not preclude its simultaneous use as an analog input.

D.5.4 ADC Control Register 0

ADCTL0 — ADC Control Register 0 **$YFF70A**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NOT USED								RES10	STS[1:0]		PRS[4:0]				
RESET:								0	0	0	0	0	0	1	1

ADCTL0 is used to select 8- or 10-bit conversions, sample time, and ADC clock frequency. Writes to it have immediate effect.

RES10 — 10-Bit Resolution

0 = 8-bit conversion
1 = 10-bit conversion

Conversion results are appropriately aligned in result registers to reflect the number of bits.

STS[1:0] — Sample Time Selection

Total conversion time is the sum of initial sample time, transfer time, final sample time, and resolution time. Initial sample time is fixed at two ADC clocks. Transfer time is fixed at two ADC clocks. Resolution time is fixed at ten ADC clocks for an 8-bit conversion and twelve ADC clocks for a 10-bit conversion. Final sample time is determined by the STS[1:0] field. Refer to **Table D-26**.

Table D-26 Sample Time Selection

STS[1:0]	Sample Time
00	2 ADC Clock Periods
01	4 ADC Clock Periods
10	8 ADC Clock Periods
11	16 ADC Clock Periods

PRS[4:0] — Prescaler Rate Selection

The ADC clock is derived from the system clock by a programmable prescaler. ADC clock period is determined by the value of the PRS field in ADCTL0. The prescaler has two stages. The first stage is a 5-bit modulus counter. It divides the system clock by any value from two to 32 (PRS[4:0] = %00000 to %11111). The second stage is a divide-by-two circuit. Refer to **Table D-27**.

NF — Noise Error
0 = No noise detected in the received data.
1 = Noise detected in the received data.

FE — Framing Error
0 = No framing error detected in the received data.
1 = Framing error or break detected in the received data.

PF — Parity Error
0 = No parity error detected in the received data.
1 = Parity error detected in the received data.

D.6.7 SCI Data Register

SCDR — SCI Data Register **$YFFC0E**

15 … 9	8	7	6	5	4	3	2	1	0
NOT USED	R8/T8	R7/T7	R6/T6	R5/T5	R4/T4	R3/T3	R2/T2	R1/T1	R0/T0
RESET:	U	U	U	U	U	U	U	U	U

SCDR consists of two data registers located at the same address. The receive data register (RDR) is a read-only register that contains data received by the SCI serial interface. Data comes into the receive serial shifter and is transferred to RDR. The transmit data register (TDR) is a write-only register that contains data to be transmitted. Data is first written to TDR, then transferred to the transmit serial shifter, where additional format bits are added before transmission. R[7:0]/T[7:0] contain either the first eight data bits received when SCDR is read, or the first eight data bits to be transmitted when SCDR is written. R8/T8 are used when the SCI is configured for nine-bit operation. When the SCI is configured for 8-bit operation, R8/T8 has no meaning or effect.

D.6.8 Port QS Data Register

PORTQS — Port QS Data Register **$YFFC14**

15 … 8	7	6	5	4	3	2	1	0
NOT USED	PQS7	PQS6	PQS5	PQS4	PQS3	PQS2	PQS1	PQS0
RESET:	0	0	0	0	0	0	0	0

PORTQS latches I/O data. Writes drive pins defined as outputs. Reads return data present on the pins. To avoid driving undefined data, first write a byte to PORTQS, then configure DDRQS.

OC1M[5:1] — OC1 Mask Field
OC1M[5:1] correspond to OC[5:1].
0 = Corresponding output compare pin is not affected by OC1 compare.
1 = Corresponding output compare pin is affected by OC1 compare.

OC1D[5:1] — OC1 Data Field
OC1D[5:1] correspond to OC[5:1].
0 = If OC1 mask bit is set, clear the corresponding output compare pin on OC1 match.
1 = If OC1 mask bit is set, the set corresponding output compare pin on OC1 match.

D.8.6 Timer Counter Register

TCNT — Timer Counter Register **$YFF90A**

TCNT is the 16-bit free-running counter associated with the input capture, output compare, and pulse accumulator functions of the GPT module.

D.8.7 Pulse Accumulator Control Register/Counter

PACTL/PACNT — Pulse Accumulator Control Register/Counter **$YFF90C**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PAIS	PAEN	PAMOD	PEDGE	PCLKS	I4/O5	PACLK[1:0]		PULSE ACCUMULATOR COUNTER							
RESET:															
U	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

PACTL enables the pulse accumulator and selects either event counting or gated mode. In event counting mode, PACNT is incremented each time an event occurs. In gated mode, it is incremented by an internal clock.

PAIS — PAI Pin State (Read Only)

PAEN — Pulse Accumulator Enable
0 = Pulse accumulator disabled.
1 = Pulse accumulator enabled.

PAMOD — Pulse Accumulator Mode
0 = External event counting.
1 = Gated time accumulation.

PEDGE — Pulse Accumulator Edge Control
The effects of PAMOD and PEDGE are shown in **Table D-44**.

```
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
```

```
        STAB    TMSK2           ;& set the TCNT's prescale to sysclock/128

*       Set up Input Capture and Output Compare

        LDAB    #$27            ;Input Captures
        STAB    TCTL2           ;TIC1=either, TIC2=rise, TIC3=fall, TIC4=off
        LDAB    #$01            ;Output Compares
        STAB    TCTL1           ;TOC2=toggle, TOC3=off, TOC4=off, TOC5=off
        LDD     #$1000          ;set OC2 to toggle every time that
        STD     TOC2            ;TCNT is #$1000

*       Set up the Pulse Width Modulators A and B

        LDD     #$0064          ;set PWM prescaler to div by 128
        STD     PWMC            ;set PWMA fast (512 Hz)
                                ;and PWMB slow (4 Hz)
        LDAB    #$80            ;set 50% duty cycle
        STAB    PWMA            ;in PWMA
        STAB    PWMB            ;in PWMB

*       Set up the Pulse Accumulator

        LDD     #$5000          ;set PAC to sense rising edges in
        STD     PACTL           ;event counting mode

*       Other Initializations

        LDAB    #$00
        TBXK                    ;set XK to bank 0 for STRING access
PAOV_CNT EQU    0               ;counter variable for PAOV_ROUTINE
        LDAB    #$01
        TBZK
        LDZ     #$0000          ;PAOV_CNT will be indexed off ZK:IZ
        LDAB    #$0A
        STAB    PAOV_CNT,Z      ;load a 10 into the variable
        ANDP    #$FF1F          ;set interrupt priority mask level to 0

*****  Start of main program  *****

GO:     NOP
        BRA     GO              ;Let's loop until we're interrupted
*****  Subroutines  *****

SEND_STRING:
                EVEN                    ;subroutine to send out the entire ASCII
string
        LDAB    0,X             ;get next byte in string as pointed to by IX
        BEQ     STRING_DONE     ;if B=00, then the string is done
        JSR     SEND_CH         ;go send out the byte
        AIX     #$01            ;increment IX to point to the next byte
        BRA     SEND_STRING     ;loop back and do next byte in string
STRING_DONE:
        RTS                     ;go back to whence we came
SEND_CH:                        ;subroutine to send out one byte to SCI
        LDAA    SCSR            ;read SCI status reg to check/clear TDRE bit
```

–B–

–C–