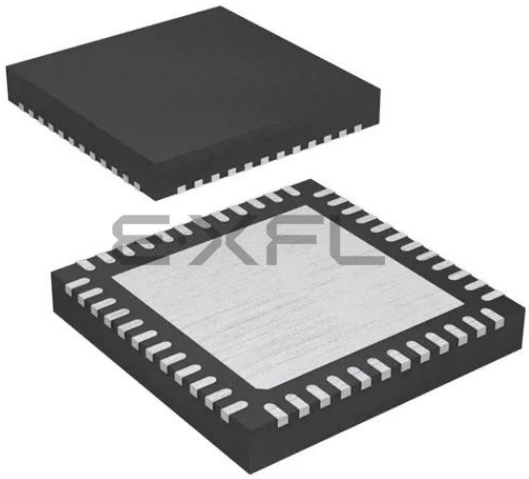


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	S08
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	38
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K 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	48-VFQFN Exposed Pad
Supplier Device Package	48-QFN-EP (7x7)
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08aw32cfde

Chapter 6
Parallel Input/Output

Chapter 7
Central Processor Unit (S08CPUV2)

Chapter 9
Keyboard Interrupt (S08KBIV1)

Chapter 10
Timer/PWM (S08TPMV2)

Chapter 6
Parallel Input/Output

6.1 Introduction

This chapter explains software controls related to parallel input/output (I/O). The MC9S08AW60 has seven I/O ports which include a total of 54 general-purpose I/O pins. See Chapter 2, “Pins and Connections” for more information about the logic and hardware aspects of these pins.

Many of these pins are shared with on-chip peripherals such as timer systems, communication systems, or keyboard interrupts. When these other modules are not controlling the port pins, they revert to general-purpose I/O control.

Pins that are not used in the application must be terminated. This prevents excess current caused by floating inputs and enhances immunity during noise or transient events. Termination methods include:

- Configuring unused pins as outputs driving high or low
- Configuring unused pins as inputs and using internal or external pullups

Never connect unused pins to V_{DD} or V_{SS}.

Table 6-1. KBI and Parallel I/O Interaction

PTxPEn (Pull Enable)	PTxDDn (Data Direction)	KBIPEn (KBI Pin Enable)	KBEDGn (KBI Edge Select)	Pullup	Pulldown
0	0	0	x[1]	disabled	disabled
1	0	0	x	enabled	disabled
x	1	0	x	disabled	disabled
1	x	1	0	enabled	disabled
1	x	1	1	disabled	enabled
0	x	1	x	disabled	disabled

1 x = Don’t care

6.2 Features

Parallel I/O and Pin Control features, depending on package choice, include:

- A total of 54 general-purpose I/O pins in seven ports
- Hysteresis input buffers
- Software-controlled pullups on each input pin

6.7.9 Port E I/O Registers (PTED and PTEDD)

Port E parallel I/O function is controlled by the registers listed below.

	7	6	5	4	3	2	1	0
R W	PTED7	PTED6	PTED5	PTED4	PTED3	PTED2	PTED1	PTED0
Reset	0	0	0	0	0	0	0	0

Figure 6-29. Port E Data Register (PTED)

Table 6-22. PTED Register Field Descriptions

Field	Description
7:0 PTED[7:0]	**Port E Data Register Bits** — For port E pins that are inputs, reads return the logic level on the pin. For port E pins that are configured as outputs, reads return the last value written to this register. Writes are latched into all bits of this register. For port E pins that are configured as outputs, the logic level is driven out the corresponding MCU pin. Reset forces PTED to all 0s, but these 0s are not driven out the corresponding pins because reset also configures all port pins as high-impedance inputs with pullups disabled.

	7	6	5	4	3	2	1	0
R W	PTEDD7	PTEDD6	PTEDD5	PTEDD4	PTEDD3	PTEDD2	PTEDD1	PTEDD0
Reset	0	0	0	0	0	0	0	0

Figure 6-30. Data Direction for Port E (PTEDD)

Table 6-23. PTEDD Register Field Descriptions

Field	Description
7:0 PTEDD[7:0]	**Data Direction for Port E Bits** — These read/write bits control the direction of port E pins and what is read for PTED reads. 0 Input (output driver disabled) and reads return the pin value. 1 Output driver enabled for port E bit n and PTED reads return the contents of PTEDn.

Chapter 7
Central Processor Unit (S08CPUV2)

7.1 Introduction

This section provides summary information about the registers, addressing modes, and instruction set of the CPU of the HCS08 family. For a more detailed discussion, refer to the *HCS08 Family Reference Manual, volume 1,* Freescale Semiconductor document order number HCS08RMV1/D.

The HCS08 CPU is fully source- and object-code-compatible with the M68HC08 CPU. Several instructions and enhanced addressing modes were added to improve C compiler efficiency and to support a new background debug system which replaces the monitor mode of earlier M68HC08 microcontrollers (MCU).

7.1.1 Features

Features of the HCS08 CPU include:

- Object code fully upward-compatible with M68HC05 and M68HC08 Families
- All registers and memory are mapped to a single 64-Kbyte address space
- 16-bit stack pointer (any size stack anywhere in 64-Kbyte address space)
- 16-bit index register (H:X) with powerful indexed addressing modes
- 8-bit accumulator (A)
- Many instructions treat X as a second general-purpose 8-bit register
- Seven addressing modes:
  - Inherent — Operands in internal registers
  - Relative — 8-bit signed offset to branch destination
  - Immediate — Operand in next object code byte(s)
  - Direct — Operand in memory at 0x0000–0x00FF
  - Extended — Operand anywhere in 64-Kbyte address space
  - Indexed relative to H:X — Five submodes including auto increment
  - Indexed relative to SP — Improves C efficiency dramatically
- Memory-to-memory data move instructions with four address mode combinations
- Overflow, half-carry, negative, zero, and carry condition codes support conditional branching on the results of signed, unsigned, and binary-coded decimal (BCD) operations
- Efficient bit manipulation instructions
- Fast 8-bit by 8-bit multiply and 16-bit by 8-bit divide instructions
- STOP and WAIT instructions to invoke low-power operating modes

8.5.4 Example #3: No External Crystal Connection, 5.4 MHz Bus Frequency

In this example, the FLL will be used (in FEI mode) to multiply the internal 243 kHz (approximate) reference clock up to 10.8 MHz to achieve 5.4 MHz bus frequency. This system will also use the trim function to fine tune the frequency based on an external reference signal.

After the MCU is released from reset, the ICG is in self-clocked mode (SCM) and supplies approximately 8 MHz on ICGOUT which corresponds to a 4 MHz bus frequency (f_{Bus}).

The clock scheme will be FLL engaged, internal (FEI). So

$$f_{ICGOUT} = (f_{IRG} / 7) * P * N / R ; P = 64, \ f_{IRG} = \ 243 \text{ kHz} \qquad \textit{Eqn. 8-5}$$

Solving for N / R gives:

$$N / R = 10.8 \text{ MHz} /(243/7 \text{ kHz} * 64) = 4.86 ; \ \text{We can choose N = 10 and R = 2.} \qquad \textit{Eqn. 8-6}$$

A trim procedure will be required to hone the frequency to exactly 5.4 MHz. An example of the trim procedure is shown in example #4.

The values needed in each register to set up the desired operation are:

ICGC1 = $28 (%00101000)

Bit 7	HGO	0	Configures oscillator for low power
Bit 6	RANGE	0	Configures oscillator for low-frequency range; FLL prescale factor is 64
Bit 5	REFS	1	Oscillator using crystal or resonator requested (bit is really a don't care)
Bits 4:3	CLKS	01	FLL engaged, internal reference clock mode
Bit 2	OSCSTEN	0	Disables the oscillator
Bit 1	LOCD	0	Loss-of-clock enabled
Bit 0		0	Unimplemented or reserved, always reads zero

ICGC2 = $31 (%00110001)

Bit 7	LOLRE	0	Generates an interrupt request on loss of lock
Bit 6:4	MFD	011	Sets the MFD multiplication factor to 10
Bit 3	LOCRE	0	Generates an interrupt request on loss of clock
Bit 2:0	RFD	001	Sets the RFD division factor to ÷2

ICGS1 = $xx

This is read only except for clearing interrupt flag

ICGS2 = $xx

This is read only; good idea to read this before performing time critical operations

ICGFLTLU/L = $xx

Not used in this example

10.2.1 Features

The TPM has the following features:

- Each TPM may be configured for buffered, center-aligned pulse-width modulation (CPWM) on all channels
- Clock sources independently selectable per TPM (multiple TPMs device)
- Selectable clock sources (device dependent): bus clock, fixed system clock, external pin
- Clock prescaler taps for divide by 1, 2, 4, 8, 16, 32, 64, or 128
- 16-bit free-running or up/down (CPWM) count operation
- 16-bit modulus register to control counter range
- Timer system enable
- One interrupt per channel plus a terminal count interrupt for each TPM module (multiple TPMs device)
- Channel features:
  - Each channel may be input capture, output compare, or buffered edge-aligned PWM
  - Rising-edge, falling-edge, or any-edge input capture trigger
  - Set, clear, or toggle output compare action
  - Selectable polarity on PWM outputs

10.2.2 Block Diagram

Figure 10-2 shows the structure of a TPM. Some MCUs include more than one TPM, with various numbers of channels.

	7	6	5	4	3	2	1	0
R	Bit 7	6	5	4	3	2	1	Bit 0
W	Any write to TPMxCNTL clears the 16-bit counter.							
Reset	0	0	0	0	0	0	0	0

Figure 10-5. Timer x Counter Register Low (TPMxCNTL)

When background mode is active, the timer counter and the coherency mechanism are frozen such that the buffer latches remain in the state they were in when the background mode became active even if one or both bytes of the counter are read while background mode is active.

10.4.3 Timer x Counter Modulo Registers (TPMxMODH:TPMxMODL)

The read/write TPM modulo registers contain the modulo value for the TPM counter. After the TPM counter reaches the modulo value, the TPM counter resumes counting from 0x0000 at the next clock (CPWMS = 0) or starts counting down (CPWMS = 1), and the overflow flag (TOF) becomes set. Writing to TPMxMODH or TPMxMODL inhibits TOF and overflow interrupts until the other byte is written. Reset sets the TPM counter modulo registers to 0x0000, which results in a free-running timer counter (modulo disabled).

	7	6	5	4	3	2	1	0
R W	Bit 15	14	13	12	11	10	9	Bit 8
Reset	0	0	0	0	0	0	0	0

Figure 10-6. Timer x Counter Modulo Register High (TPMxMODH)

	7	6	5	4	3	2	1	0
R W	Bit 7	6	5	4	3	2	1	Bit 0
Reset	0	0	0	0	0	0	0	0

Figure 10-7. Timer x Counter Modulo Register Low (TPMxMODL)

It is good practice to wait for an overflow interrupt so both bytes of the modulo register can be written well before a new overflow. An alternative approach is to reset the TPM counter before writing to the TPM modulo registers to avoid confusion about when the first counter overflow will occur.

transferred to the corresponding timer channel registers only after both 8-bit bytes of a 16-bit register have been written and the timer counter overflows (reverses direction from up-counting to down-counting at the end of the terminal count in the modulus register). This TPMxCNT overflow requirement only applies to PWM channels, not output compares.

Optionally, when TPMxCNTH:TPMxCNTL = TPMxMODH:TPMxMODL, the TPM can generate a TOF interrupt at the end of this count. The user can choose to reload any number of the PWM buffers, and they will all update simultaneously at the start of a new period.

Writing to TPMxSC cancels any values written to TPMxMODH and/or TPMxMODL and resets the coherency mechanism for the modulo registers. Writing to TPMxCnSC cancels any values written to the channel value registers and resets the coherency mechanism for TPMxCnVH:TPMxCnVL.

10.6 TPM Interrupts

The TPM generates an optional interrupt for the main counter overflow and an interrupt for each channel. The meaning of channel interrupts depends on the mode of operation for each channel. If the channel is configured for input capture, the interrupt flag is set each time the selected input capture edge is recognized. If the channel is configured for output compare or PWM modes, the interrupt flag is set each time the main timer counter matches the value in the 16-bit channel value register. See the Resets, Interrupts, and System Configuration chapter for absolute interrupt vector addresses, priority, and local interrupt mask control bits.

For each interrupt source in the TPM, a flag bit is set on recognition of the interrupt condition such as timer overflow, channel input capture, or output compare events. This flag may be read (polled) by software to verify that the action has occurred, or an associated enable bit (TOIE or CHnIE) can be set to enable hardware interrupt generation. While the interrupt enable bit is set, a static interrupt will be generated whenever the associated interrupt flag equals 1. It is the responsibility of user software to perform a sequence of steps to clear the interrupt flag before returning from the interrupt service routine.

10.6.1 Clearing Timer Interrupt Flags

TPM interrupt flags are cleared by a 2-step process that includes a read of the flag bit while it is set (1) followed by a write of 0 to the bit. If a new event is detected between these two steps, the sequence is reset and the interrupt flag remains set after the second step to avoid the possibility of missing the new event.

10.6.2 Timer Overflow Interrupt Description

The conditions that cause TOF to become set depend on the counting mode (up or up/down). In up-counting mode, the 16-bit timer counter counts from 0x0000 through 0xFFFF and overflows to 0x0000 on the next counting clock. TOF becomes set at the transition from 0xFFFF to 0x0000. When a modulus limit is set, TOF becomes set at the transition from the value set in the modulus register to 0x0000. When the counter is operating in up-/down-counting mode, the TOF flag gets set as the counter changes direction at the transition from the value set in the modulus register and the next lower count value. This corresponds to the end of a PWM period. (The 0x0000 count value corresponds to the center of a period.)

Figure 11-3 shows the receiver portion of the SCI.

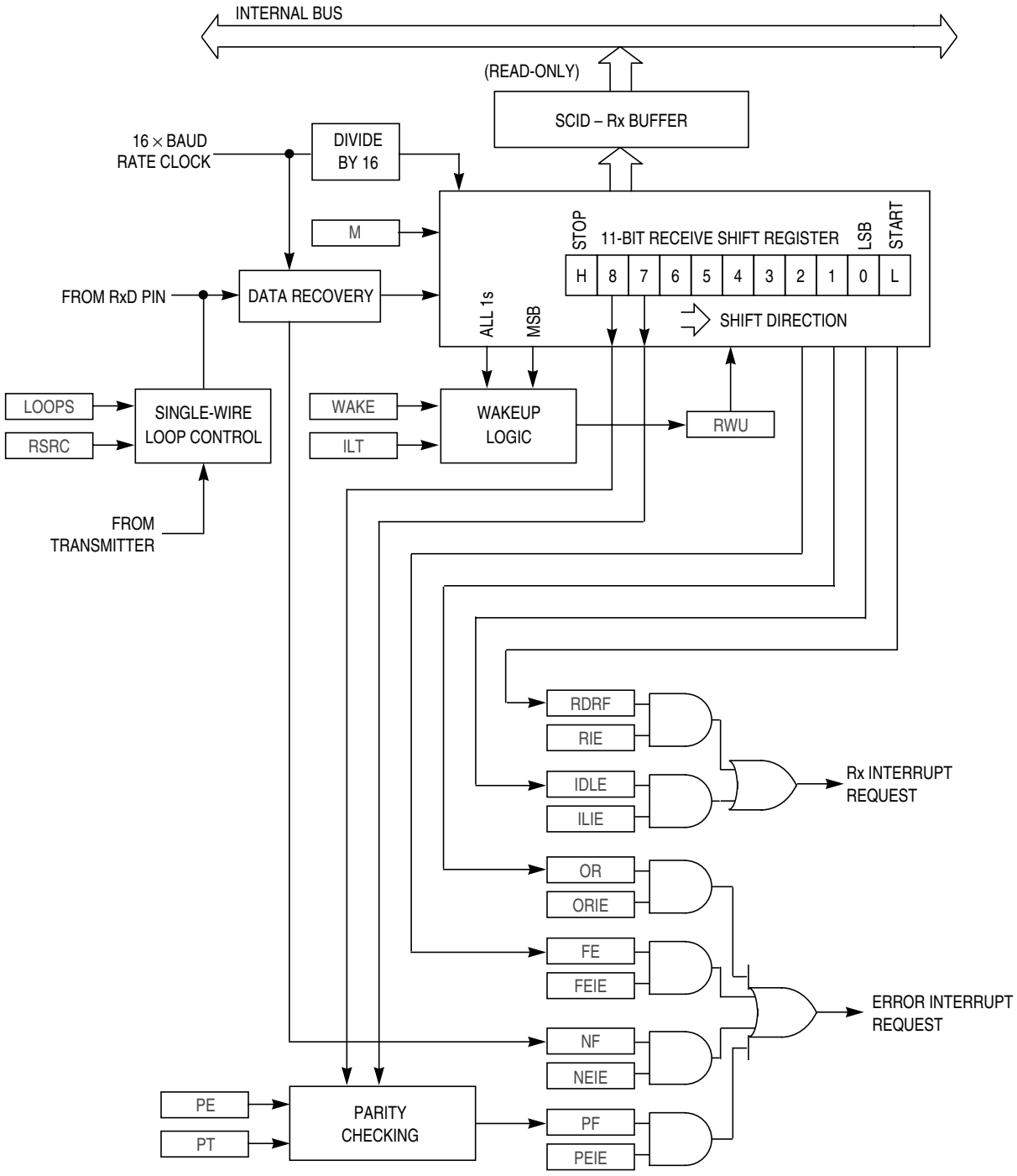
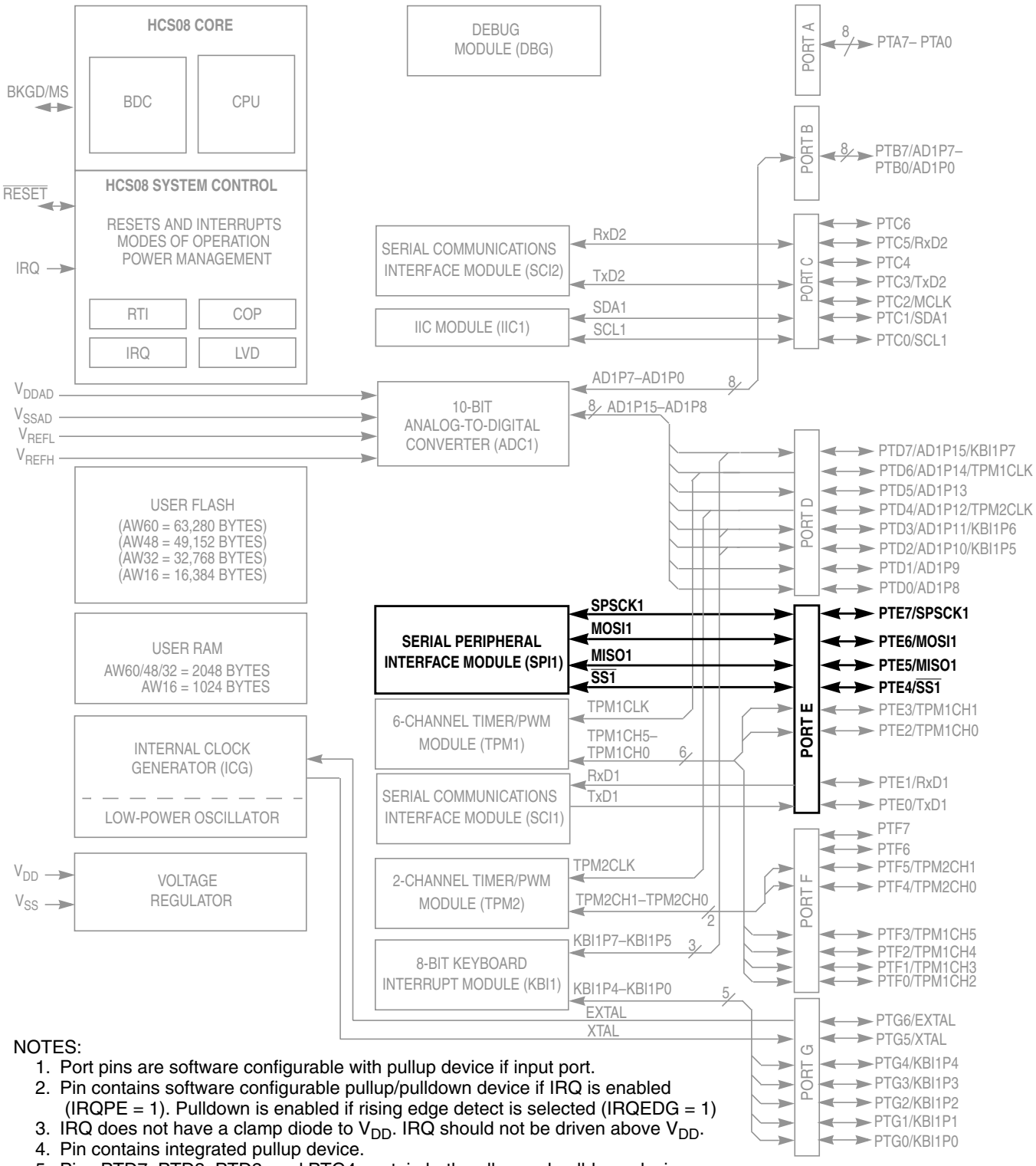


Figure 11-3. SCI Receiver Block Diagram

11.2 Register Definition

The SCI has eight 8-bit registers to control baud rate, select SCI options, report SCI status, and for transmit/receive data.



NOTES:
1. Port pins are software configurable with pullup device if input port.
2. Pin contains software configurable pullup/pulldown device if IRQ is enabled (IRQPE = 1). Pulldown is enabled if rising edge detect is selected (IRQEDG = 1)
3. IRQ does not have a clamp diode to V_{DD}. IRQ should not be driven above V_{DD}.
4. Pin contains integrated pullup device.
5. Pins PTD7, PTD3, PTD2, and PTG4 contain both pullup and pulldown devices. Pulldown enabled when KBI is enabled (KBIPEn = 1) and rising edge is selected (KBEDGn = 1).

Figure 12-1. Block Diagram Highlighting the SPI Module

13.4.1.1 START Signal

When the bus is free; i.e., no master device is engaging the bus (both SCL and SDA lines are at logical high), a master may initiate communication by sending a START signal. As shown in Figure 13-8, a START signal is defined as a high-to-low transition of SDA while SCL is high. This signal denotes the beginning of a new data transfer (each data transfer may contain several bytes of data) and brings all slaves out of their idle states.

13.4.1.2 Slave Address Transmission

The first byte of data transferred immediately after the START signal is the slave address transmitted by the master. This is a seven-bit calling address followed by a R/W bit. The R/W bit tells the slave the desired direction of data transfer.

1 = Read transfer, the slave transmits data to the master.
0 = Write transfer, the master transmits data to the slave.

Only the slave with a calling address that matches the one transmitted by the master will respond by sending back an acknowledge bit. This is done by pulling the SDA low at the 9th clock (see Figure 13-8).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address that is equal to its own slave address. The IIC cannot be master and slave at the same time. However, if arbitration is lost during an address cycle, the IIC will revert to slave mode and operate correctly even if it is being addressed by another master.

13.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/W bit sent by the calling master.

All transfers that come after an address cycle are referred to as data transfers, even if they carry sub-address information for the slave device

Each data byte is 8 bits long. Data may be changed only while SCL is low and must be held stable while SCL is high as shown in Figure 13-8. There is one clock pulse on SCL for each data bit, the MSB being transferred first. Each data byte is followed by a 9th (acknowledge) bit, which is signalled from the receiving device. An acknowledge is signalled by pulling the SDA low at the ninth clock. In summary, one complete data transfer needs nine clock pulses.

If the slave receiver does not acknowledge the master in the 9th bit time, the SDA line must be left high by the slave. The master interprets the failed acknowledge as an unsuccessful data transfer.

If the master receiver does not acknowledge the slave transmitter after a data byte transmission, the slave interprets this as an end of data transfer and releases the SDA line.

In either case, the data transfer is aborted and the master does one of two things:

- Relinquishes the bus by generating a STOP signal.
- Commences a new calling by generating a repeated START signal.

13.6.1 Byte Transfer Interrupt

The TCF (transfer complete flag) bit is set at the falling edge of the 9th clock to indicate the completion of byte transfer.

13.6.2 Address Detect Interrupt

When the calling address matches the programmed slave address (IIC address register), the IAAS bit in the status register is set. The CPU is interrupted provided the IICIE is set. The CPU must check the SRW bit and set its Tx mode accordingly.

13.6.3 Arbitration Lost Interrupt

The IIC is a true multi-master bus that allows more than one master to be connected on it. If two or more masters try to control the bus at the same time, the relative priority of the contending masters is determined by a data arbitration procedure. The IIC module asserts this interrupt when it loses the data arbitration process and the ARBL bit in the status register is set.

Arbitration is lost in the following circumstances:

- SDA sampled as a low when the master drives a high during an address or data transmit cycle.
- SDA sampled as a low when the master drives a high during the acknowledge bit of a data receive cycle.
- A START cycle is attempted when the bus is busy.
- A repeated START cycle is requested in slave mode.
- A STOP condition is detected when the master did not request it.

This bit must be cleared by software by writing a one to it.

14.2.2.1 Analog Pin Enables

The ADC on MC9S08AW60 Series contains only two analog pin enable registers, APCTL1 and APCTL2.

14.2.2.2 Low-Power Mode Operation

The ADC is capable of running in stop3 mode but requires LVDSE and LVDE in SPMSC1 to be set.

14.2.3 Temperature Sensor

The ADC1 module includes a temperature sensor whose output is connected to one of the ADC analog channel inputs. Equation 14-1 provides an approximate transfer function of the temperature sensor.

$$Temp = 25 - ((V_{TEMP} - V_{TEMP25}) \div m) \qquad \textit{Eqn. 14-1}$$

where:

- V_{TEMP} is the voltage of the temperature sensor channel at the ambient temperature.
- V_{TEMP25} is the voltage of the temperature sensor channel at 25°C.
- m is the hot or cold voltage versus temperature slope in V/°C.

For temperature calculations, use the V_{TEMP25} and m values from the ADC Electricals table.

In application code, the user reads the temperature sensor channel, calculates V_{TEMP}, and compares to V_{TEMP25} . If V_{TEMP} is greater than V_{TEMP25} the cold slope value is applied in Equation 14-1. If V_{TEMP} is less than V_{TEMP25} the hot slope value is applied in Equation 14-1.

14.5.7.2 Stop3 Mode With ADACK Enabled

If ADACK is selected as the conversion clock, the ADC continues operation during stop3 mode. For guaranteed ADC operation, the MCU's voltage regulator must remain active during stop3 mode. Consult the module introduction for configuration information for this MCU.

If a conversion is in progress when the MCU enters stop3 mode, it continues until completion. Conversions can be initiated while the MCU is in stop3 mode by means of the hardware trigger or if continuous conversions are enabled.

A conversion complete event sets the COCO and generates an ADC interrupt to wake the MCU from stop3 mode if the ADC interrupt is enabled (AIEN = 1).

NOTE

It is possible for the ADC module to wake the system from low power stop and cause the MCU to begin consuming run-level currents without generating a system level interrupt. To prevent this scenario, software should ensure that the data transfer blocking mechanism (discussed in Section 14.5.4.2, "Completing Conversions) is cleared when entering stop3 and continuing ADC conversions.

14.5.8 MCU Stop1 and Stop2 Mode Operation

The ADC module is automatically disabled when the MCU enters either stop1 or stop2 mode. All module registers contain their reset values following exit from stop1 or stop2. Therefore the module must be re-enabled and re-configured following exit from stop1 or stop2.

14.6 Initialization Information

This section gives an example which provides some basic direction on how a user would initialize and configure the ADC module. The user has the flexibility of choosing between configuring the module for 8-bit or 10-bit resolution, single or continuous conversion, and a polled or interrupt approach, among many other options. Refer to Table 14-6, Table 14-7, and Table 14-8 for information used in this example.

NOTE

Hexadecimal values designated by a preceding 0x, binary values designated by a preceding %, and decimal values have no preceding character.

14.6.1 ADC Module Initialization Example

14.6.1.1 Initialization Sequence

Before the ADC module can be used to complete conversions, an initialization procedure must be performed. A typical sequence is as follows:

1. Update the configuration register (ADCCFG) to select the input clock source and the divide ratio used to generate the internal clock, ADCK. This register is also used for selecting sample time and low-power configuration.

- Average the result by converting the analog input many times in succession and dividing the sum of the results. Four samples are required to eliminate the effect of a 1LSB, one-time error.
- Reduce the effect of synchronous noise by operating off the asynchronous clock (ADACK) and averaging. Noise that is synchronous to ADCK cannot be averaged out.

14.7.2.4 Code Width and Quantization Error

The ADC quantizes the ideal straight-line transfer function into 1024 steps (in 10-bit mode). Each step ideally has the same height (1 code) and width. The width is defined as the delta between the transition points to one code and the next. The ideal code width for an N bit converter (in this case N can be 8 or 10), defined as 1LSB, is:

$$1_{LSB} = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 14-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code will transition when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2LSB in 8- or 10-bit mode. As a consequence, however, the code width of the first ($000) conversion is only 1/2LSB and the code width of the last ($FF or $3FF) is 1.5LSB.

14.7.2.5 Linearity Errors

The ADC may also exhibit non-linearity of several forms. Every effort has been made to reduce these errors but the system should be aware of them because they affect overall accuracy. These errors are:

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2LSB). Note, if the first conversion is $001, then the difference between the actual $001 code width and its ideal (1LSB) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5LSB). Note, if the last conversion is $3FE, then the difference between the actual $3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function, and therefore includes all forms of error.

14.7.2.6 Code Jitter, Non-Monotonicity and Missing Codes

Analog-to-digital converters are susceptible to three special forms of error. These are code jitter, non-monotonicity, and missing codes.

Code jitter is when, at certain points, a given input voltage converts to one of two values when sampled repeatedly. Ideally, when the input voltage is infinitesimally smaller than the transition voltage, the

converter yields the lower code (and vice-versa). However, even very small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around 1/2LSB and will increase with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 14.7.2.3 will reduce this error.

Non-monotonicity is defined as when, except for code jitter, the converter converts to a lower code for a higher input voltage. Missing codes are those values which are never converted for any input value.

In 8-bit or 10-bit mode, the ADC is guaranteed to be monotonic and to have no missing codes.

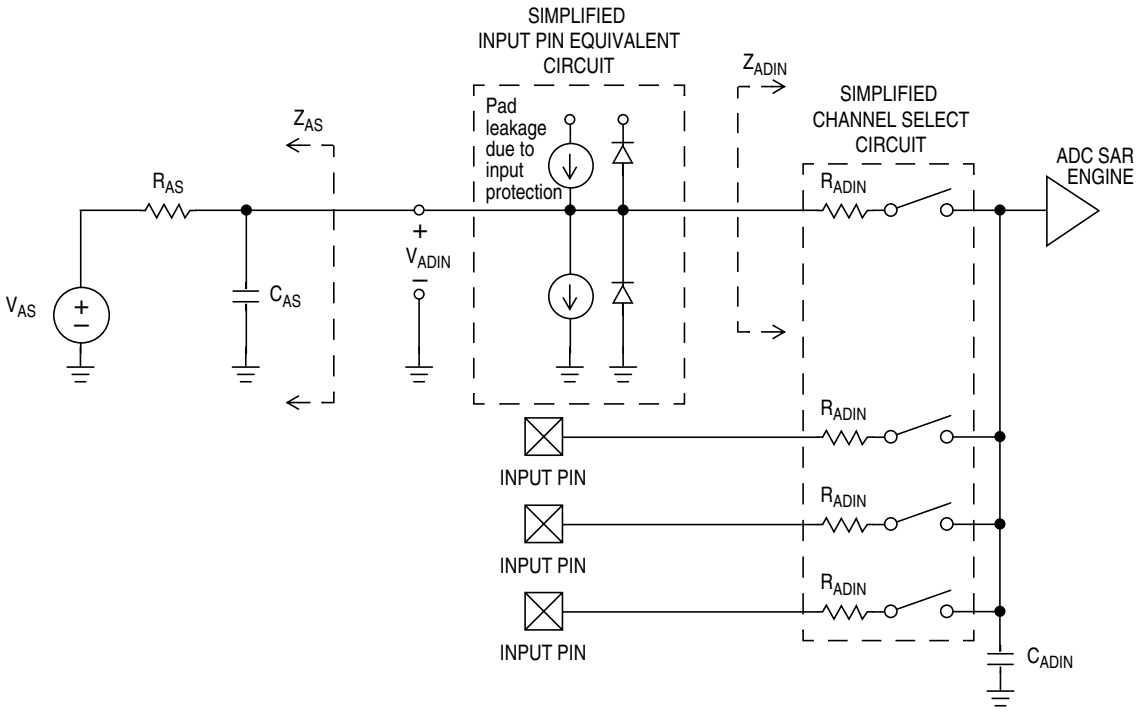
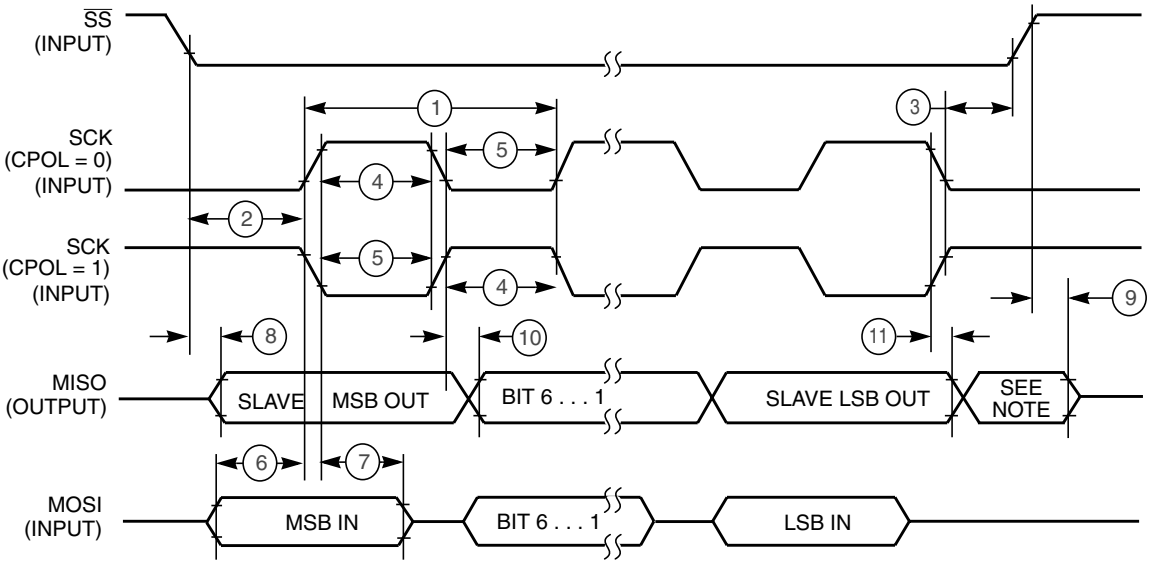


Figure A-8. ADC Input Impedance Equivalency Diagram

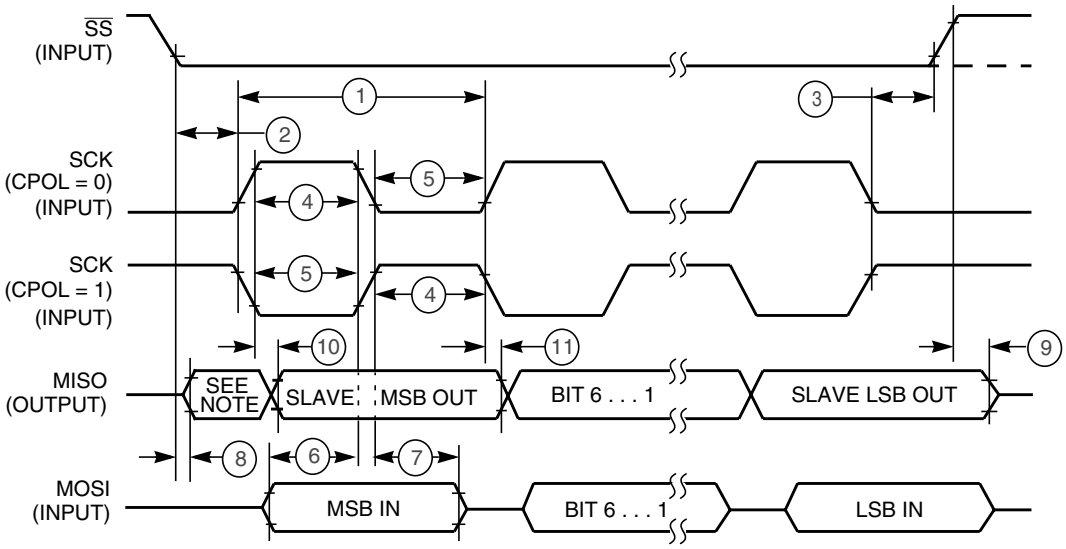
Table A-10. 10-bit ADC Characteristics (V_{REFH} = V_{DDAD}, V_{REFL} = V_{SSAD})

Characteristic	Conditions	C	Symb	Min	Typ[1]	Max	Unit
Supply current ADLPC = 1 ADLSMP = 1 ADCO = 1		T	I_{DDAD}	—	133	—	µA
Supply current ADLPC = 1 ADLSMP = 0 ADCO = 1		T	I_{DDAD}	—	218	—	µA
Supply current ADLPC = 0 ADLSMP = 1 ADCO = 1		T	I_{DDAD}	—	327	—	µA
Supply current ADLPC = 0 ADLSMP = 0 ADCO = 1	$V_{DDAD} \leq 5.5$ V	P	I_{DDAD}	—	582	990	µA
Supply current	Stop, reset, module off		I_{DDAD}	—	0.011	1	µA



NOTE:
1. Not defined but normally MSB of character just received

Figure A-18. SPI Slave Timing (CPHA = 0)



NOTE:
1. Not defined but normally LSB of character just received

Figure A-19. SPI Slave Timing (CPHA = 1)