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	HCS08
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, LINbus, SCI, SPI
Peripherals	LVD, POR, PWM, WDT
Number of I/O	22
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	512 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 16x10b SAR
Oscillator Type	External, Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	28-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08el16ctl

Table 4-2. Direct-Page Register Summary (Sheet 1 of 3)

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x0000	PTAD	PTAD7	PTAD6	0	0	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	PTADD7	PTADD6	0	0	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	PTBD	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004	PTCD	PTCD7	PTCD6	PTCD5	PTCD4	PTCD3	PTCD2	PTCD1	PTCD0
0x0005	PTCDD	PTCDD7	PTCDD6	PTCDD5	PTCDD4	PTCDD3	PTCDD2	PTCDD1	PTCDD0
0x0006–0x000D	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x000E	ACMP1SC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
0x000F	ACMP2SC	ACME	ACBGS	ACF	ACIE	ACO	ACOPE	ACMOD1	ACMOD0
0x0010	ADCSC1	COCO	AIEN	ADCO	ADCH				
0x0011	ADCSC2	ADACT	ADTRG	ACFE	ACFGT	—	—	—	—
0x0012	ADCRH	0	0	0	0	0	0	ADR9	ADR8
0x0013	ADCRL	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
0x0014	ADCCVH	0	0	0	0	0	0	ADCV9	ADCV8
0x0015	ADCCVL	ADCV7	ADCV6	ADCV5	ADCV4	ADCV3	ADCV2	ADCV1	ADCV0
0x0016	ADCCFG	ADLPC	ADIV		ADLSMP	MODE		ADICLK	
0x0017	APCTL1	ADPC7	ADPC6	ADPC5	ADPC4	ADPC3	ADPC2	ADPC1	ADPC0
0x0018	APCTL2	ADPC15	ADPC14	ADPC13	ADPC12	ADPC11	ADPC10	ADPC9	ADPC8
0x0019–0x001F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —
0x0020	TPM1SC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0021	TPM1CNTH	Bit 15	14	13	12	11	10	9	Bit 8
0x0022	TPM1CNTL	Bit 7	6	5	4	3	2	1	Bit 0
0x0023	TPM1MODH	Bit 15	14	13	12	11	10	9	Bit 8
0x0024	TPM1MODL	Bit 7	6	5	4	3	2	1	Bit 0
0x0025	TPM1C0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0026	TPM1C0VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0027	TPM1C0VL	Bit 7	6	5	4	3	2	1	Bit 0
0x0028	TPM1C1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0029	TPM1C1VH	Bit 15	14	13	12	11	10	9	Bit 8
0x002A	TPM1C1VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002B	TPM1C2SC	CH2F	CH2IE	MS2B	MS2A	ELS2B	ELS2A	0	0
0x002C	TPM1C2VH	Bit 15	14	13	12	11	10	9	Bit 8
0x002D	TPM1C2VL	Bit 7	6	5	4	3	2	1	Bit 0
0x002E	TPM1C3SC	CH3F	CH3IE	MS3B	MS3A	ELS3B	ELS3A	0	0
0x002F	TPM1C3VH	Bit 15	14	13	12	11	10	9	Bit 8
0x0030	TPM1C3VL	Bit 7	6	5	4	3	2	1	Bit 0

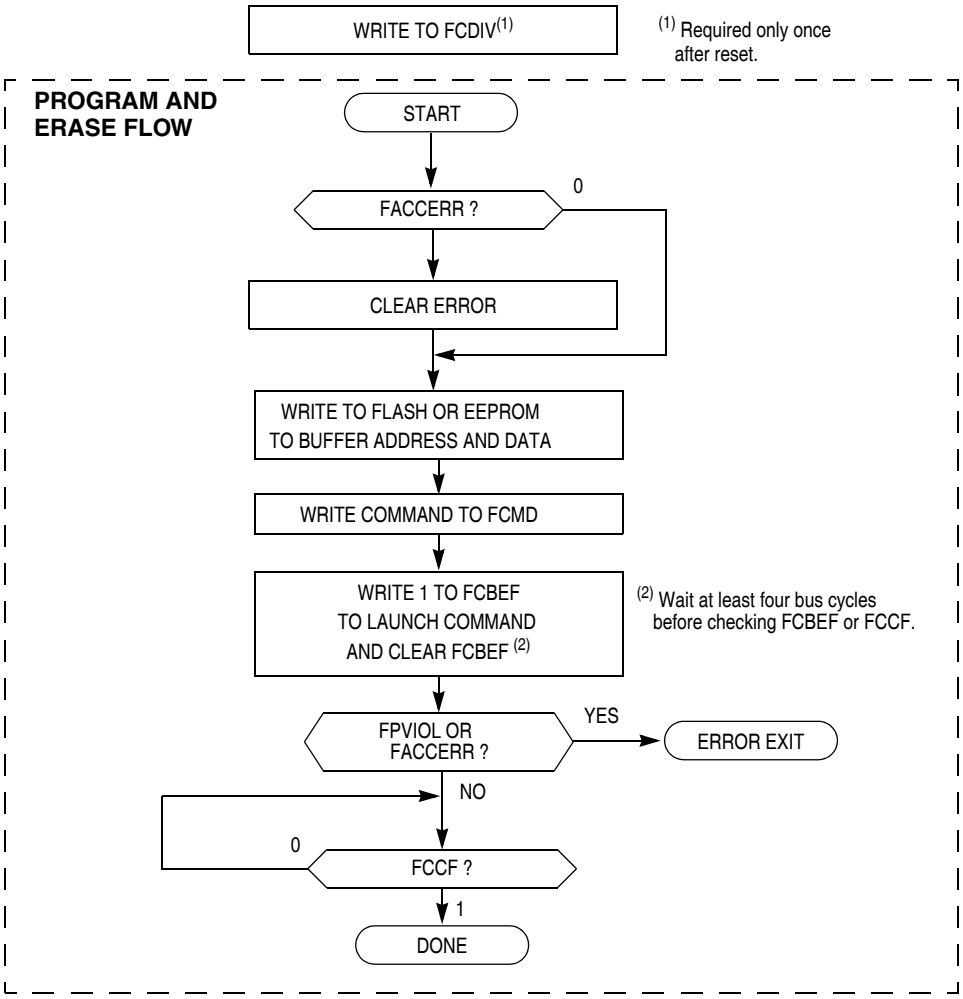


Figure 4-2. Program and Erase Flowchart

4.5.4 Burst Program Execution

The burst program command is used to program sequential bytes of data in less time than would be required using the standard program command. This is possible because the high voltage to the FLASH array does not need to be disabled between program operations. Ordinarily, when a program or erase command is issued, an internal charge pump associated with the FLASH memory must be enabled to supply high voltage to the array. Upon completion of the command, the charge pump is turned off. When a burst program command is issued, the charge pump is enabled and then remains enabled after completion of the burst program operation if these two conditions are met:

- The next burst program command has been queued before the current program operation has completed.
- The next sequential address selects a byte on the same burst block as the current byte being programmed. A burst block in this FLASH memory consists of 64 bytes. A new burst block begins at each 64-byte address boundary.

4.5.11.6 FLASH and EEPROM Command Register (FCMD)

Only six command codes are recognized in normal user modes as shown in Table 4-15. All other command codes are illegal and generate an access error. Refer to Section 4.5.3, "Program and Erase Command Execution," for a detailed discussion of FLASH and EEPROM programming and erase operations.

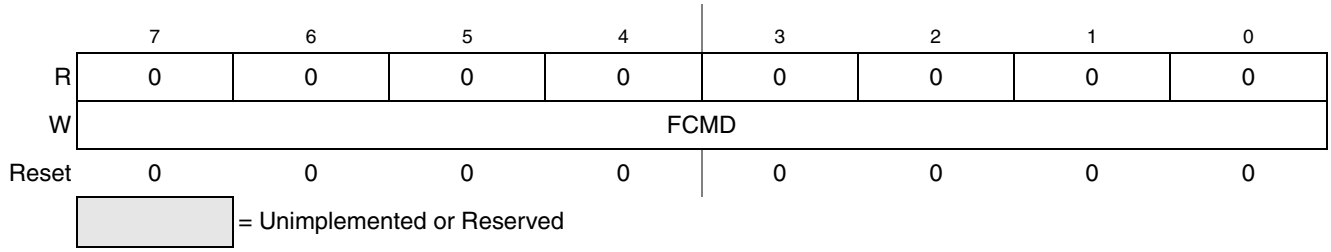


Figure 4-10. FLASH and EEPROM Command Register (FCMD)

Table 4-15. FLASH and EEPROM Commands

Command	FCMD	Equate File Label
Blank check	0x05	mBlank
Byte program	0x20	mByteProg
Burst program	0x25	mBurstProg
Sector erase	0x40	mSectorErase
Mass erase	0x41	mMassErase
Sector erase abort	0x47	mEraseAbort

It is not necessary to perform a blank check command after a mass erase operation. Only blank check is required as part of the security unlocking mechanism.

Chapter 5
Resets, Interrupts, and General System Control

5.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt in the MC9S08EL32 Series and MC9S08SL16 Series. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this data sheet. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog are not part of on-chip peripheral systems with their own chapters.

5.2 Features

Reset and interrupt features include:

- Multiple sources of reset for flexible system configuration and reliable operation
- Reset status register (SRS) to indicate source of most recent reset
- Separate interrupt vector for each module (reduces polling overhead) (see Table 5-2)

5.3 MCU Reset

Resetting the MCU provides a way to start processing from a known set of initial conditions. During reset, most control and status registers are forced to initial values and the program counter is loaded from the reset vector (0xFFFE:0xFFFF). On-chip peripheral modules are disabled and I/O pins are initially configured as general-purpose high-impedance inputs with pull-up devices disabled. The I bit in the condition code register (CCR) is set to block maskable interrupts so the user program has a chance to initialize the stack pointer (SP) and system control settings. SP is forced to 0x00FF at reset.

The MC9S08EL32 Series and MC9S08SL16 Series has eight sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD)
- Background debug forced reset

Each of these sources, with the exception of the background debug forced reset, has an associated bit in the system reset status register (SRS).

The COP counter is initialized by the first writes to the SOPT1 and SOPT2 registers after any system reset. Subsequent writes to SOPT1 and SOPT2 have no effect on COP operation. Even if the application will use the reset default settings of COPT, COPCLKS, and COPW bits, the user should write to the write-once SOPT1 and SOPT2 registers during reset initialization to lock in the settings. This will prevent accidental changes if the application program gets lost.

The write to SRS that services (clears) the COP counter should not be placed in an interrupt service routine (ISR) because the ISR could continue to be executed periodically even if the main application program fails.

If the bus clock source is selected, the COP counter does not increment while the MCU is in background debug mode or while the system is in stop mode. The COP counter resumes when the MCU exits background debug mode or stop mode.

If the 1-kHz clock source is selected, the COP counter is re-initialized to zero upon entry to either background debug mode or stop mode and begins from zero upon exit from background debug mode or stop mode.

5.5 Interrupts

Interrupts provide a way to save the current CPU status and registers, execute an interrupt service routine (ISR), and then restore the CPU status so processing resumes where it left off before the interrupt. Other than the software interrupt (SWI), which is a program instruction, interrupts are caused by hardware events such as an edge on an external interrupt pin or a timer-overflow event. The debug module can also generate an SWI under certain circumstances.

If an event occurs in an enabled interrupt source, an associated read-only status flag will become set. The CPU will not respond unless the local interrupt enable is a 1 to enable the interrupt and the I bit in the CCR is 0 to allow interrupts. The global interrupt mask (I bit) in the CCR is initially set after reset which prevents all maskable interrupt sources. The user program initializes the stack pointer and performs other system setup before clearing the I bit to allow the CPU to respond to interrupts.

When the CPU receives a qualified interrupt request, it completes the current instruction before responding to the interrupt. The interrupt sequence obeys the same cycle-by-cycle sequence as the SWI instruction and consists of:

- Saving the CPU registers on the stack
- Setting the I bit in the CCR to mask further interrupts
- Fetching the interrupt vector for the highest-priority interrupt that is currently pending
- Filling the instruction queue with the first three bytes of program information starting from the address fetched from the interrupt vector locations

While the CPU is responding to the interrupt, the I bit is automatically set to avoid the possibility of another interrupt interrupting the ISR itself (this is called nesting of interrupts). Normally, the I bit is restored to 0 when the CCR is restored from the value stacked on entry to the ISR. In rare cases, the I bit can be cleared inside an ISR (after clearing the status flag that generated the interrupt) so that other interrupts can be serviced without waiting for the first service routine to finish. This practice is not

Chapter 6
Parallel Input/Output Control

This section explains software controls related to parallel input/output (I/O) and pin control. The MC9S08EL32 has three parallel I/O ports which include a total of 22 I/O pins. See Chapter 2, “Pins and Connections,” for more information about pin assignments and external hardware considerations of these pins.

Many of these pins are shared with on-chip peripherals such as timer systems, communication systems, or keyboard interrupts as shown in Table 2-1. The peripheral modules have priority over the general-purpose I/O functions so that when a peripheral is enabled, the I/O functions associated with the shared pins are disabled.

After reset, the shared peripheral functions are disabled and the pins are configured as inputs (PTxDDn = 0). The pin control functions for each pin are configured as follows: slew rate control enabled (PTxSEn = 1), low drive strength selected (PTxDSn = 0), and internal pull-ups disabled (PTxPEn = 0).

NOTE

Not all general-purpose I/O pins are available on all packages. To avoid extra current drain from floating input pins, the user’s reset initialization routine in the application program must either enable on-chip pull-up devices or change the direction of unconnected pins to outputs so the pins do not float.

6.1 Port Data and Data Direction

Reading and writing of parallel I/Os are performed through the port data registers. The direction, either input or output, is controlled through the port data direction registers. The parallel I/O port function for an individual pin is illustrated in the block diagram shown in Figure 6-1.

The data direction control bit (PTxDDn) determines whether the output buffer for the associated pin is enabled, and also controls the source for port data register reads. The input buffer for the associated pin is always enabled unless the pin is enabled as an analog function or is an output-only pin.

When a shared digital function is enabled for a pin, the output buffer is controlled by the shared function. However, the data direction register bit will continue to control the source for reads of the port data register.

When a shared analog function is enabled for a pin, both the input and output buffers are disabled. A value of 0 is read for any port data bit where the bit is an input (PTxDDn = 0) and the input buffer is disabled. In general, whenever a pin is shared with both an alternate digital function and an analog function, the analog function has priority such that if both the digital and analog functions are enabled, the analog function controls the pin.

11.4.1.2 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/$\overline{W}$ bit. The R/$\overline{W}$ 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 responds by sending back an acknowledge bit. This is done by pulling the SDA low at the ninth clock (see Figure 11-9).

No two slaves in the system may have the same address. If the IIC module is the master, it must not transmit an address 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 reverts to slave mode and operates correctly even if it is being addressed by another master.

11.4.1.3 Data Transfer

Before successful slave addressing is achieved, the data transfer can proceed byte-by-byte in a direction specified by the R/$\overline{W}$ 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 11-9. 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 ninth 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.

11.4.1.4 Stop Signal

The master can terminate the communication by generating a stop signal to free the bus. However, the master may generate a start signal followed by a calling command without generating a stop signal first. This is called repeated start. A stop signal is defined as a low-to-high transition of SDA while SCL at logical 1 (see Figure 11-9).

The master can generate a stop even if the slave has generated an acknowledge at which point the slave must release the bus.

11.4.1.8 Handshaking

The clock synchronization mechanism can be used as a handshake in data transfer. Slave devices may hold the SCL low after completion of one byte transfer (9 bits). In such a case, it halts the bus clock and forces the master clock into wait states until the slave releases the SCL line.

11.4.1.9 Clock Stretching

The clock synchronization mechanism can be used by slaves to slow down the bit rate of a transfer. After the master has driven SCL low the slave can drive SCL low for the required period and then release it. If the slave SCL low period is greater than the master SCL low period then the resulting SCL bus signal low period is stretched.

11.4.2 10-bit Address

For 10-bit addressing, 0x11110 is used for the first 5 bits of the first address byte. Various combinations of read/write formats are possible within a transfer that includes 10-bit addressing.

11.4.2.1 Master-Transmitter Addresses a Slave-Receiver

The transfer direction is not changed (see Table 11-10). When a 10-bit address follows a start condition, each slave compares the first seven bits of the first byte of the slave address (11110XX) with its own address and tests whether the eighth bit (R/$\overline{W}$ direction bit) is 0. More than one device can find a match and generate an acknowledge (A1). Then, each slave that finds a match compares the eight bits of the second byte of the slave address with its own address. Only one slave finds a match and generates an acknowledge (A2). The matching slave remains addressed by the master until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

S	Slave Address 1st 7 bits 11110 + AD10 + AD9	R/W 0	A1	Slave Address 2nd byte AD[8:1]	A2	Data	A	...	Data	A/A	P

Table 11-10. Master-Transmitter Addresses Slave-Receiver with a 10-bit Address

After the master-transmitter has sent the first byte of the 10-bit address, the slave-receiver sees an IIC interrupt. Software must ensure the contents of IICD are ignored and not treated as valid data for this interrupt.

11.4.2.2 Master-Receiver Addresses a Slave-Transmitter

The transfer direction is changed after the second R/$\overline{W}$ bit (see Table 11-11). Up to and including acknowledge bit A2, the procedure is the same as that described for a master-transmitter addressing a slave-receiver. After the repeated start condition (Sr), a matching slave remembers that it was addressed before. This slave then checks whether the first seven bits of the first byte of the slave address following Sr are the same as they were after the start condition (S) and tests whether the eighth (R/$\overline{W}$) bit is 1. If there is a match, the slave considers that it has been addressed as a transmitter and generates acknowledge A3. The slave-transmitter remains addressed until it receives a stop condition (P) or a repeated start condition (Sr) followed by a different slave address.

2. When INITACK = 0, write SLCC2 with desired values for:
   a) SLCWCM — Wait clock mode.
3. Write SLCC2 to set up:
   a) RXFP — Digital receive filter clock prescaler.
4. Enable the SLIC module by writing SLCC2:
   a) SLCE = 1 to place SLIC module into run mode.
   b) BTM = 1 to enable byte transfer mode.
5. Write SLCBT value.
6. Write SLCC1 to enable SLIC interrupts (if desired).

NOTE

The SLIC module is designed primarily for use in LIN systems and assumes the connection of a LIN transceiver, which provides a resistive path between the transmit and receive pins. BTM mode will not operate properly without a resistive feedback path between SLCTx and SLCRx.

12.6.7 Handling LIN Message Headers

Figure 12-14 shows how the SLIC module deals with incoming LIN message headers.

Table 12-13. Maximum LIN Bit Rates for High-Speed Operation Due to Digital Receive Filter

SLIC Clock (MHz)	Maximum LIN Bit Rate for ±1.5% SLIC Accuracy (for Master-Slave Communication (kbps) DIGITAL RX FILTER NOT CONSIDERED	RXFP Prescaler Values (See Table 12-11)							
		÷8 (Note 1)	÷7 (Note 1)	÷6 (Note 1)	÷÷5 (Note 1)	÷÷4 (Note 1)	÷÷3 (Note 1)	÷÷2	÷÷1
		Maximum LIN Bit Rate (kbps)[1]							
20	300	120.00	120.00	120.00	120.00	120.00	120.00	120.00	120.00
18	270	120.00	120.00	120.00	120.00	120.00	120.00	120.00	120.00
16	240	120.00	120.00	120.00	120.00	120.00	120.00	120.00	120.00
14	210	109.38	120.00	120.00	120.00	120.00	120.00	120.00	120.00
12	180	93.75	107.14	120.00	120.00	120.00	120.00	120.00	120.00
10	150	78.13	89.29	104.17	120.00	120.00	120.00	120.00	120.00
8	120	62.50	71.43	83.33	100.00	120.00	120.00	120.00	120.00
6	90	46.88	53.57	62.50	75.00	93.75	120.00	120.00	120.00
4	60	31.25	35.71	41.67	50.00	62.50	83.33	120.00	120.00
2	30	15.63	17.86	20.83	25.00	31.25	41.67	62.50	120.00

[1] Bit rates over 120,000 bits per second are not recommended for LIN communications, as physical layer delay between the TX and RX pins can cause the stop bit of a byte to be mis-sampled as the last data bit. This could result in a byte framing error.

Table 12-14. Digital Receive Filter Absolute Cutoff (Ideal Conditions)[1]

SLIC clock (MHz)	Max Bit Rate (kbps)	Min Pulse Width Allowed (µs)	Max Bit Rate (kbps)	Min Pulse Width Allowed (µs)	Max Bit Rate (kbps)	Min Pulse Width Allowed (µs)	Max Bit Rate (kbps)	Min Pulse Width Allowed (µs)
	RXFP = ÷8		RXFP = ÷7		RXFP = ÷6		RXFP = ÷5	
20	156,250	6.40	178,571	5.60	208,333	4.80	250,000	4.00
18	140,625	7.11	160,714	6.22	187,500	5.33	225,000	4.44
16	125,000	8.00	142,857	7.00	166,667	6.00	200,000	5.00
14	109,375	9.14	125,000	8.00	145,833	6.86	175,000	5.71
12	93,750	10.67	107,143	9.33	125,000	8.00	150,000	6.67
10	78,125	12.80	89,286	11.20	104,167	9.60	125,000	8.00
8	62,500	16.00	71,429	14.00	83,333	12.00	100,000	10.00
6	46,875	21.33	53,571	18.67	62,500	16.00	75,000	13.33
4	31,250	32.00	35,714	28.00	41,667	24.00	50,000	20.00

Desired Bit Rate: 57,600 bps
External Crystal Frequency: 4.9152 MHz

$$\frac{1\ \text{Second}}{57{,}600\ \text{Bits}} = \frac{17.36111\ \mu\text{s}}{1\ \text{Bit}}$$

$$\frac{1\ \text{Second}}{4{,}915{,}200\ \text{Clock Out Period}} \times \frac{2\ \text{Clock Out Period}}{1\ \text{SLIC Clock Period}} = \frac{406.901\ \text{ns}}{1\ \text{SLIC Clock Period}}$$

$$\frac{17.36111\ \mu\text{s}}{1\ \text{Bit}} \times \frac{1\ \text{SLIC Clock Period}}{406.901\ \text{ns}} = \frac{42.67\ \text{SLIC Clock Periods}}{1\ \text{Bit}}$$

Therefore, the closest SLCBT value would be 43 SLIC clocks (SLCBT = 0x002B).
Because you can only use even values in SLCBT, the closest acceptable value is 42 (0x002A).

Figure 12-17. SLCBT Value Calculation Example 1

Desired Bit Rate: 57,600 bps
External Crystal Frequency: 9.8304 MHz

$$\frac{1\ \text{Second}}{57{,}600\ \text{Bits}} = \frac{17.36111\ \mu\text{s}}{1\ \text{Bit}}$$

$$\frac{1\ \text{Second}}{9{,}830{,}400\ \text{Clock Out Periods}} \times \frac{2\ \text{Clock Out Period}}{1\ \text{SLIC Clock Period}} = \frac{203.45\ \text{ns}}{1\ \text{SLIC Clock Period}}$$

$$\frac{17.36111\ \mu\text{s}}{1\ \text{Bit}} \times \frac{1\ \text{SLIC Clock Period}}{203.45\ \text{ns}} = \frac{85.33\ \text{SLIC Clock Periods}}{1\ \text{Bit}}$$

Therefore, the closest SLCBT value would be 85 SLIC clocks (SLCBT = 0x0055).
Because you can only use even values in SLCBT, the closest acceptable value is 86 (0x0056)

Figure 12-18. SLCBT Value Calculation Example 2

Desired Bit Rate: 15,625 bps
External Crystal Frequency: 8.000 MHz

$$\frac{1\ \text{Second}}{15{,}625\ \text{Bits}} = \frac{64\ \mu\text{s}}{1\ \text{Bit}}$$

$$\frac{1\ \text{Second}}{8{,}000{,}000\ \text{Clock Out Periods}} \times \frac{2\ \text{Clock Out Period}}{1\ \text{SLIC Clock Period}} = \frac{250\ \text{ns}}{1\ \text{SLIC Clock Period}}$$

$$\frac{64\ \mu\text{s}}{1\ \text{Bit}} \times \frac{1\ \text{SLIC Clock Period}}{250\ \text{ns}} = \frac{256\ \text{SLIC Clock Periods}}{1\ \text{Bit}}$$

Therefore, the closest SLCBT value would be 256 SLIC clocks (SLCBT = 0x0100).

Figure 12-19. SLCBT Value Calculation Example 3

13.3 Modes of Operation

13.3.1 SPI in Stop Modes

The SPI is disabled in all stop modes, regardless of the settings before executing the STOP instruction. During either stop1 or stop2 mode, the SPI module will be fully powered down. Upon wake-up from stop1 or stop2 mode, the SPI module will be in the reset state. During stop3 mode, clocks to the SPI module are halted. No registers are affected. If stop3 is exited with a reset, the SPI will be put into its reset state. If stop3 is exited with an interrupt, the SPI continues from the state it was in when stop3 was entered.

13.4 Register Definition

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

Refer to the direct-page register summary in the Memory chapter of this data sheet for the absolute address assignments for all SPI registers. This section refers to registers and control bits only by their names, and a Freescale-provided equate or header file is used to translate these names into the appropriate absolute addresses.

13.4.1 SPI Control Register 1 (SPIC1)

This read/write register includes the SPI enable control, interrupt enables, and configuration options.

	7	6	5	4	3	2	1	0
R W	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
Reset	0	0	0	0	0	1	0	0

Figure 13-5. SPI Control Register 1 (SPIC1)

Table 13-1. SPIC1 Field Descriptions

Field	Description
7 SPIE	**SPI Interrupt Enable (for SPRF and MODF)** — This is the interrupt enable for SPI receive buffer full (SPRF) and mode fault (MODF) events. 0 Interrupts from SPRF and MODF inhibited (use polling) 1 When SPRF or MODF is 1, request a hardware interrupt
6 SPE	**SPI System Enable** — Disabling the SPI halts any transfer that is in progress, clears data buffers, and initializes internal state machines. SPRF is cleared and SPTEF is set to indicate the SPI transmit data buffer is empty. 0 SPI system inactive 1 SPI system enabled
5 SPTIE	**SPI Transmit Interrupt Enable** — This is the interrupt enable bit for SPI transmit buffer empty (SPTEF). 0 Interrupts from SPTEF inhibited (use polling) 1 When SPTEF is 1, hardware interrupt requested

status flag is set. If RDRF was already set indicating the receive data register (buffer) was already full, the overrun (OR) status flag is set and the new data is lost. Because the SCI receiver is double-buffered, the program has one full character time after RDRF is set before the data in the receive data buffer must be read to avoid a receiver overrun.

When a program detects that the receive data register is full (RDRF = 1), it gets the data from the receive data register by reading SCIxD. The RDRF flag is cleared automatically by a 2-step sequence which is normally satisfied in the course of the user's program that handles receive data. Refer to Section 14.3.4, "Interrupts and Status Flags" for more details about flag clearing.

14.3.3.1 Data Sampling Technique

The SCI receiver uses a 16× baud rate clock for sampling. The receiver starts by taking logic level samples at 16 times the baud rate to search for a falling edge on the RxD serial data input pin. A falling edge is defined as a logic 0 sample after three consecutive logic 1 samples. The 16× baud rate clock is used to divide the bit time into 16 segments labeled RT1 through RT16. When a falling edge is located, three more samples are taken at RT3, RT5, and RT7 to make sure this was a real start bit and not merely noise. If at least two of these three samples are 0, the receiver assumes it is synchronized to a receive character.

The receiver then samples each bit time, including the start and stop bits, at RT8, RT9, and RT10 to determine the logic level for that bit. The logic level is interpreted to be that of the majority of the samples taken during the bit time. In the case of the start bit, the bit is assumed to be 0 if at least two of the samples at RT3, RT5, and RT7 are 0 even if one or all of the samples taken at RT8, RT9, and RT10 are 1s. If any sample in any bit time (including the start and stop bits) in a character frame fails to agree with the logic level for that bit, the noise flag (NF) will be set when the received character is transferred to the receive data buffer.

The falling edge detection logic continuously looks for falling edges, and if an edge is detected, the sample clock is resynchronized to bit times. This improves the reliability of the receiver in the presence of noise or mismatched baud rates. It does not improve worst case analysis because some characters do not have any extra falling edges anywhere in the character frame.

In the case of a framing error, provided the received character was not a break character, the sampling logic that searches for a falling edge is filled with three logic 1 samples so that a new start bit can be detected almost immediately.

In the case of a framing error, the receiver is inhibited from receiving any new characters until the framing error flag is cleared. The receive shift register continues to function, but a complete character cannot transfer to the receive data buffer if FE is still set.

14.3.3.2 Receiver Wakeup Operation

Receiver wakeup is a hardware mechanism that allows an SCI receiver to ignore the characters in a message that is intended for a different SCI receiver. In such a system, all receivers evaluate the first character(s) of each message, and as soon as they determine the message is intended for a different receiver, they write logic 1 to the receiver wake up (RWU) control bit in SCIxC2. When RWU bit is set, the status flags associated with the receiver (with the exception of the idle bit, IDLE, when RWUID bit is set) are inhibited from setting, thus eliminating the software overhead for handling the unimportant

- Edge-aligned PWM mode

  The value of a 16-bit modulo register plus 1 sets the period of the PWM output signal. The channel value register sets the duty cycle of the PWM output signal. The user may also choose the polarity of the PWM output signal. Interrupts are available at the end of the period and at the duty-cycle transition point. This type of PWM signal is called edge-aligned because the leading edges of all PWM signals are aligned with the beginning of the period, which is the same for all channels within a TPM.
- Center-aligned PWM mode

  Twice the value of a 16-bit modulo register sets the period of the PWM output, and the channel-value register sets the half-duty-cycle duration. The timer counter counts up until it reaches the modulo value and then counts down until it reaches zero. As the count matches the channel value register while counting down, the PWM output becomes active. When the count matches the channel value register while counting up, the PWM output becomes inactive. This type of PWM signal is called center-aligned because the centers of the active duty cycle periods for all channels are aligned with a count value of zero. This type of PWM is required for types of motors used in small appliances.

This is a high-level description only. Detailed descriptions of operating modes are in later sections.

16.1.3 Block Diagram

The TPM uses one input/output (I/O) pin per channel, TPMxCHn (timer channel n) where n is the channel number (1-8). The TPM shares its I/O pins with general purpose I/O port pins (refer to I/O pin descriptions in full-chip specification for the specific chip implementation).

Figure 16-2 shows the TPM structure. The central component of the TPM is the 16-bit counter that can operate as a free-running counter or a modulo up/down counter. The TPM counter (when operating in normal up-counting mode) provides the timing reference for the input capture, output compare, and edge-aligned PWM functions. The timer counter modulo registers, TPMxMODH:TPMxMODL, control the modulo value of the counter (the values 0x0000 or 0xFFFF effectively make the counter free running). Software can read the counter value at any time without affecting the counting sequence. Any write to either half of the TPMxCNT counter resets the counter, regardless of the data value written.

16.4.1.3 Counting Modes

The main timer counter has two counting modes. When center-aligned PWM is selected (CPWMS=1), the counter operates in up/down counting mode. Otherwise, the counter operates as a simple up counter. As an up counter, the timer counter counts from 0x0000 through its terminal count and then continues with 0x0000. The terminal count is 0xFFFF or a modulus value in TPMxMODH:TPMxMODL.

When center-aligned PWM operation is specified, the counter counts up from 0x0000 through its terminal count and then down to 0x0000 where it changes back to up counting. Both 0x0000 and the terminal count value are normal length counts (one timer clock period long). In this mode, the timer overflow flag (TOF) becomes set at the end of the terminal-count period (as the count changes to the next lower count value).

16.4.1.4 Manual Counter Reset

The main timer counter can be manually reset at any time by writing any value to either half of TPMxCNTH or TPMxCNTL. Resetting the counter in this manner also resets the coherency mechanism in case only half of the counter was read before resetting the count.

16.4.2 Channel Mode Selection

Provided CPWMS=0, the MSnB and MSnA control bits in the channel n status and control registers determine the basic mode of operation for the corresponding channel. Choices include input capture, output compare, and edge-aligned PWM.

16.4.2.1 Input Capture Mode

With the input-capture function, the TPM can capture the time at which an external event occurs. When an active edge occurs on the pin of an input-capture channel, the TPM latches the contents of the TPM counter into the channel-value registers (TPMxCnVH:TPMxCnVL). Rising edges, falling edges, or any edge may be chosen as the active edge that triggers an input capture.

In input capture mode, the TPMxCnVH and TPMxCnVL registers are read only.

When either half of the 16-bit capture register is read, the other half is latched into a buffer to support coherent 16-bit accesses in big-endian or little-endian order. The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An input capture event sets a flag bit (CHnF) which may optionally generate a CPU interrupt request.

While in BDM, the input capture function works as configured by the user. When an external event occurs, the TPM latches the contents of the TPM counter (which is frozen because of the BDM mode) into the channel value registers and sets the flag bit.

16.4.2.2 Output Compare Mode

With the output-compare function, the TPM can generate timed pulses with programmable position, polarity, duration, and frequency. When the counter reaches the value in the channel-value registers of an output-compare channel, the TPM can set, clear, or toggle the channel pin.

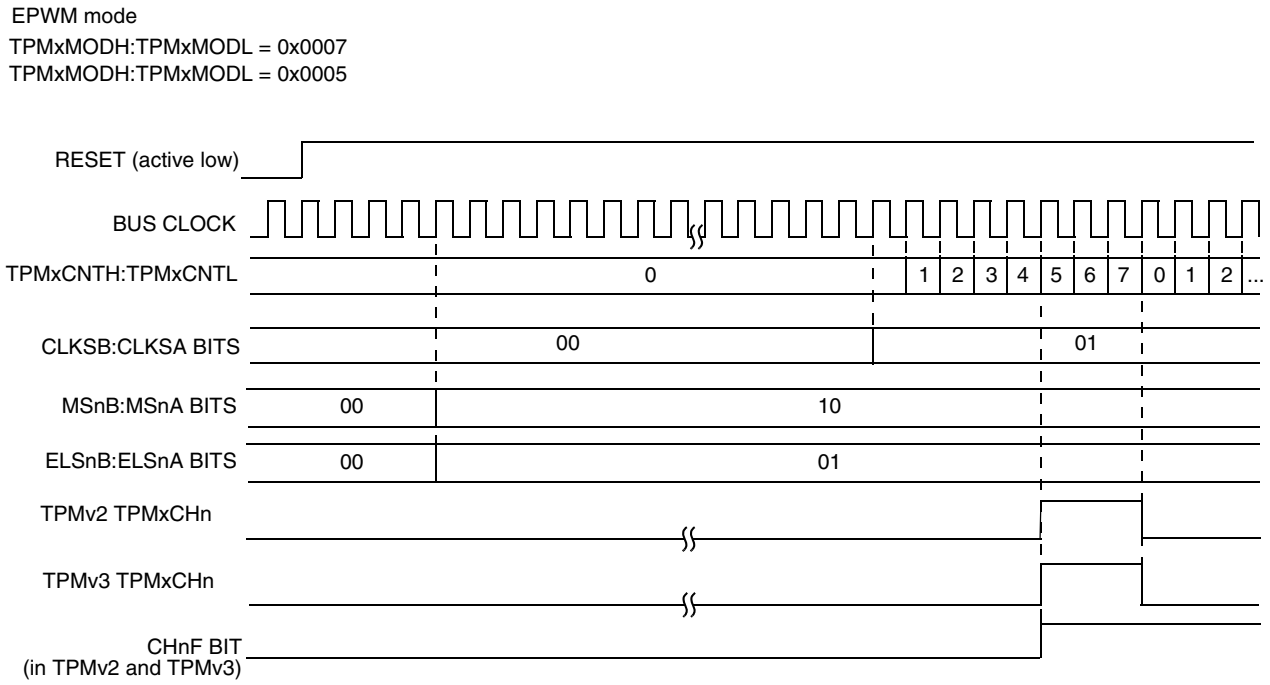


Figure 0-2. Generation of low-true EPWM signal by TPM v2 and v3 after the reset

The following procedure can be used in TPM v3 (when the channel pin is also a port pin) to emulate the high-true EPWM generated by TPM v2 after the reset.

...

configure the channel pin as output port pin and set the output pin;

configure the channel to generate the EPWM signal but keep ELSnB:ELSnA as 00;

configure the other registers (TPMxMODH, TPMxMODL, TPMxCnVH, TPMxCnVL, ...);

configure CLKSB:CLKSA bits (TPM v3 starts to generate the high-true EPWM signal, however TPM does not control the channel pin, so the EPWM signal is not available);

wait until the TOF is set (or use the TOF interrupt);

enable the channel output by configuring ELSnB:ELSnA bits (now EPWM signal is available);

...

1 Junction temperature is a function of die size, on-chip power dissipation, package thermal resistance, mounting site (board) temperature, ambient temperature, air flow, power dissipation of other components on the board, and board thermal resistance.
2 Junction to Ambient Natural Convection

The average chip-junction temperature (T_J) in °C can be obtained from:

$$T_J = T_A + (P_D \times \theta_{JA}) \quad \textit{Eqn. A-1}$$

where:

T_A = Ambient temperature, °C
θ_{JA} = Package thermal resistance, junction-to-ambient, °C/W
$P_D = P_{int} + P_{I/O}$
$P_{int} = I_{DD} \times V_{DD}$, Watts — chip internal power
$P_{I/O}$ = Power dissipation on input and output pins — user determined

For most applications, $P_{I/O} << P_{int}$ and can be neglected. An approximate relationship between P_D and T_J (if $P_{I/O}$ is neglected) is:

$$P_D = K \div (T_J + 273°C) \quad \textit{Eqn. A-2}$$

Solving Equation A-1 and Equation A-2 for K gives:

$$K = P_D \times (T_A + 273°C) + \theta_{JA} \times (P_D)^2 \quad \textit{Eqn. A-3}$$

where K is a constant pertaining to the particular part. K can be determined from equation 3 by measuring P_D (at equilibrium) for a known T_A. Using this value of K, the values of P_D and T_J can be obtained by solving Equation A-1 and Equation A-2 iteratively for any value of T_A.

A.5 ESD Protection and Latch-Up Immunity

Although damage from electrostatic discharge (ESD) is much less common on these devices than on early CMOS circuits, normal handling precautions should be used to avoid exposure to static discharge. Qualification tests are performed to ensure that these devices can withstand exposure to reasonable levels of static without suffering any permanent damage.

All ESD testing is in conformity with AEC-Q100 Stress Test Qualification for Automotive Grade Integrated Circuits. During the device qualification ESD stresses were performed for the human body model (HBM) and the charge device model (CDM).

A device is defined as a failure if after exposure to ESD pulses the device no longer meets the device specification. Complete DC parametric and functional testing is performed per the applicable device specification at room temperature followed by hot temperature, unless specified otherwise in the device specification.

Table A-6. DC Characteristics (continued)

Num	C	Characteristic	Symbol	Condition	Min	Typ[1]	Max	Unit
23	T	Low-voltage inhibit reset/recover hysteresis	V_{hys}	5 V	—	100	—	mV
				3 V	—	60	—	
24	P	Bandgap Voltage Reference[10]	V_{BG}		1.18	1.202	1.21	V

1 Typical values are measured at 25°C. Characterized, not tested

2 When a pin interrupt is configured to detect rising edges, pulldown resistors are used in place of pullup resistors.

3 The specified resistor value is the actual value internal to the device. The pullup value may measure higher when measured externally on the pin.

4 Power supply must maintain regulation within operating V_{DD} range during instantaneous and operating maximum current conditions. If positive injection current ($V_{In} > V_{DD}$) is greater than I_{DD}, the injection current may flow out of V_{DD} and could result in external power supply going out of regulation. Ensure external V_{DD} load shunts current greater than maximum injection current. This is the greatest risk when the MCU is not consuming power. For example, if no system clock is present, or if clock rate is very low (which would reduce overall power consumption).

5 All functional non-supply pins except $\overline{\text{RESET}}$ are internally clamped to V_{SS} and V_{DD}.

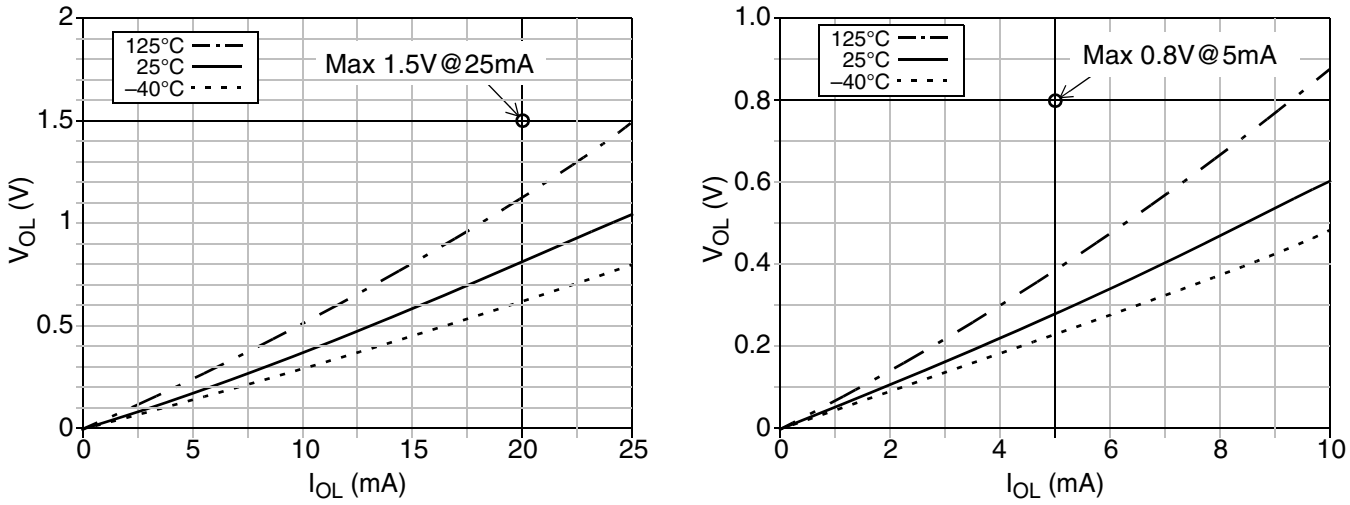
6 Input must be current limited to the value specified. To determine the value of the required current-limiting resistor, calculate resistance values for positive and negative clamp voltages, then use the larger of the two values.

7 The $\overline{\text{RESET}}$ pin does not have a clamp diode to V_{DD}. Do not drive this pin above V_{DD}.

8 Maximum is highest voltage that POR is guaranteed.

9 Simulated, not tested.

10 Factory trimmed at V_{DD} = 5.0 V, Temp = 25°C.



a) V_{DD} = 5V, High Drive **b) V_{DD} = 3V, High Drive**

Figure A-1. Typical V_{OL} vs I_{OL}, High Drive Strength