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	HC08
Core Size	8-Bit
Speed	8MHz
Connectivity	LINbus
Peripherals	LVD, POR, PWM
Number of I/O	13
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 5.5V
Data Converters	A/D 6x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	16-SOIC (0.295", 7.50mm Width)
Supplier Device Package	16-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc908ql4mdwe

The following revision history table summarizes changes contained in this document. For your convenience, the page number designators have been linked to the appropriate location.

Revision History (Sheet 1 of 2)

Date	Revision Level	Description	Page Number(s)
September, 2003	N/A	Initial release	N/A
November, 2003	1.0	17.3 Functional Operating Range — Corrected operating voltage range	226
		17.6 Control Timing — Corrected values for internal operating frequency and internal clock period	228
		17.14 5.0-Volt ADC Characteristics — Replaced ADC characteristic table found in initial release	236
		17.15 3.3-Volt ADC Characteristics — Added	237
March, 2004 Continued on next page	2.0	Figure 2-2. Control, Status, and Data Registers Corrected reset state for the FLASH Block Protect Register. Corrected reset value for the Internal Oscillator Time Value	33 34
		Table 7-1. Instruction Set Summary — Added WAIT instruction	83
		11.8.1 Oscillator Status and Control Register — Revised description of ECGON bit for clarity	117
March, 2004 Continued	2.0	Table 13-3. Interrupt Sources — Corrected address locations for SLIC, KBI, and ADC	140
		14.3.5 SLIC Wait (Core Specific) — Revised description for clarity	144
		14.3.7 SLIC Stop (Core Specific) — Revised description for clarity	144
		14.6.6.2 Byte Transfer Mode Operation — Revised definition of Receiver Buffer Overrun Error	156
		14.7.1 LIN Message Frame Header — Revised third paragraph of description	161
		14.14 Sleep and Wakeup Operation — Revised second paragraph of description	174
		15.8 Input/Output Signals — Corrected reference from PTA0/TCH) to PTB0/TCH0	196
		15.8.2 TIM Channel I/O Pins (PTB0/TCH0 and PTA1/TCH1) — Corrected reference to from PTA0/TCH) to PTB0/TCH0	196
		Figure 16-1. Block Diagram Highlighting BRK and MON Blocks — Added	206
		17.5 5-V DC Electrical Characteristics — Updated table notes	227
		17.8 5-V Oscillator Characteristics — Updated table notes	230
		17.9 3.3-V DC Electrical Characteristics — Updated table notes	231
June, 2004	3.0	Modular sections reworked for clarity.	Throughout

Table of Contents

Chapter 1
General Description

Chapter 2
Memory

Chapter 3
Analog-to-Digital Converter (ADC10) Module

Chapter 9
Keyboard Interrupt Module (KBI)

Chapter 10
Low-Voltage Inhibit (LVI)

Chapter 11
Oscillator Module (OSC)

2.5 Random-Access Memory (RAM)

This MCU includes static RAM. The locations in RAM below $0100 can be accessed using the more efficient direct addressing mode, and any single bit in this area can be accessed with the bit manipulation instructions (BCLR, BSET, BRCLR, and BRSET). Locating the most frequently accessed program variables in this area of RAM is preferred.

The RAM retains data when the MCU is in low-power wait or stop mode. At power-on, the contents of RAM are uninitialized. RAM data is unaffected by any reset provided that the supply voltage does not drop below the minimum value for RAM retention.

For compatibility with older M68HC05 MCUs, the HC08 resets the stack pointer to $00FF. In the devices that have RAM above $00FF, it is usually best to reinitialize the stack pointer to the top of the RAM so the direct page RAM can be used for frequently accessed RAM variables and bit-addressable program variables.

Include the following 2-instruction sequence in your reset initialization routine (where RamLast is equated to the highest address of the RAM).

```
LDHX    #RamLast+1    ;point one past RAM
TXS                   ;SP<-(H:X-1)
```

2.6 FLASH Memory (FLASH)

The FLASH memory is intended primarily for program storage. In-circuit programming allows the operating program to be loaded into the FLASH memory after final assembly of the application product. It is possible to program the entire array through the single-wire monitor mode interface. Because no special voltages are needed for FLASH erase and programming operations, in-application programming is also possible through other software-controlled communication paths.

This subsection describes the operation of the embedded FLASH memory. The FLASH memory can be read, programmed, and erased from the internal V_{DD} supply. The program and erase operations are enabled through the use of an internal charge pump.

The minimum size of FLASH memory that can be erased is 64 bytes; and the maximum size of FLASH memory that can be programmed in a program cycle is 32 bytes (a row). Program and erase operations are facilitated through control bits in the FLASH control register (FLCR). Details for these operations appear later in this section.

NOTE

An erased bit reads as a 1 and a programmed bit reads as a 0. A security feature prevents viewing of the FLASH contents.[1]

1. No security feature is absolutely secure. However, Freescale's strategy is to make reading or copying the FLASH difficult for unauthorized users.

Whichever clock is selected, its frequency must fall within the acceptable frequency range for ADCK. If the available clocks are too slow, the ADC10 will not perform according to specifications. If the available clocks are too fast, then the clock must be divided to the appropriate frequency. This divider is specified by the ADIV[1:0] bits and can be divide-by 1, 2, 4, or 8.

3.3.2 Input Select and Pin Control

Only one analog input may be used for conversion at any given time. The channel select bits in ADCSC are used to select the input signal for conversion.

3.3.3 Conversion Control

Conversions can be performed in either 10-bit mode or 8-bit mode as determined by the MODE bits. Conversions can be initiated by either a software or hardware trigger. In addition, the ADC10 module can be configured for low power operation, long sample time, and continuous conversion.

3.3.3.1 Initiating Conversions

A conversion is initiated:

- Following a write to ADCSC (with ADCH bits not all 1s) if software triggered operation is selected.
- Following a hardware trigger event if hardware triggered operation is selected.
- Following the transfer of the result to the data registers when continuous conversion is enabled.

If continuous conversions are enabled a new conversion is automatically initiated after the completion of the current conversion. In software triggered operation, continuous conversions begin after ADCSC is written and continue until aborted. In hardware triggered operation, continuous conversions begin after a hardware trigger event and continue until aborted.

3.3.3.2 Completing Conversions

A conversion is completed when the result of the conversion is transferred into the data result registers, ADRH and ADRL. This is indicated by the setting of the COCO bit. An interrupt is generated if AIEN is high at the time that COCO is set.

A blocking mechanism prevents a new result from overwriting previous data in ADRH and ADRL if the previous data is in the process of being read while in 10-bit mode (ADRH has been read but ADRL has not). In this case the data transfer is blocked, COCO is not set, and the new result is lost. When a data transfer is blocked, another conversion is initiated regardless of the state of ADCO (single or continuous conversions enabled). If single conversions are enabled, this could result in several discarded conversions and excess power consumption. To avoid this issue, the data registers must not be read after initiating a single conversion until the conversion completes.

3.3.3.3 Aborting Conversions

Any conversion in progress will be aborted when:

- A write to ADCSC occurs (the current conversion will be aborted and a new conversion will be initiated, if ADCH are not all 1s).
- A write to ADCLK occurs.
- The MCU is reset.
- The MCU enters stop mode with ACLK not enabled.

13.4 Reset and System Initialization

The MCU has these reset sources:

- Power-on reset module (POR)
- External reset pin ($\overline{RST}$)
- Computer operating properly module (COP)
- Low-voltage inhibit module (LVI)
- Illegal opcode
- Illegal address

All of these resets produce the vector $FFFE–FFFF ($FEFE–FEFF in monitor mode) and assert the internal reset signal (IRST). IRST causes all registers to be returned to their default values and all modules to be returned to their reset states.

An internal reset clears the SIM counter (see 13.5 SIM Counter), but an external reset does not. Each of the resets sets a corresponding bit in the SIM reset status register (SRSR). See 13.8 SIM Registers.

13.4.1 External Pin Reset

The $\overline{RST}$ pin circuits include an internal pullup device. Pulling the asynchronous $\overline{RST}$ pin low halts all processing. The PIN bit of the SIM reset status register (SRSR) is set as long as $\overline{RST}$ is held low for at least the minimum t_{RL} time. Figure 13-3 shows the relative timing. The $\overline{RST}$ pin function is only available if the RSTEN bit is set in the CONFIG2 register.

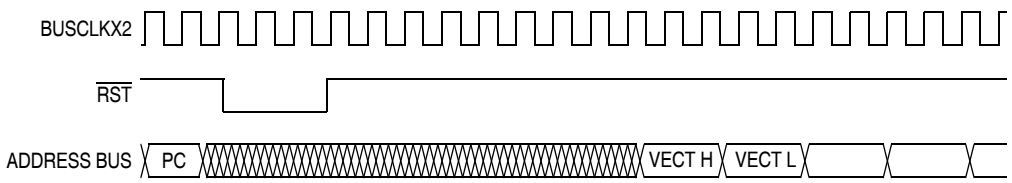


Figure 13-3. External Reset Timing

13.4.2 Active Resets from Internal Sources

The $\overline{RST}$ pin is initially setup as a general-purpose input after a POR. Setting the RSTEN bit in the CONFIG2 register enables the pin for the reset function. This section assumes the RSTEN bit is set when describing activity on the $\overline{RST}$ pin.

NOTE

For POR and LVI resets, the SIM cycles through 4096 BUSCLKX4 cycles. The internal reset signal then follows the sequence from the falling edge of $\overline{RST}$ shown in Figure 13-4.

The COP reset is asynchronous to the bus clock.

The active reset feature allows the part to issue a reset to peripherals and other chips within a system built around the MCU.

All internal reset sources actively pull the $\overline{RST}$ pin low for 32 BUSCLKX4 cycles to allow resetting of external peripherals. The internal reset signal IRST continues to be asserted for an additional 32 cycles (see Figure 13-4). An internal reset can be caused by an illegal address, illegal opcode, COP time out, LVI, or POR (see Figure 13-5).

13.6.2 Interrupt Status Registers

The flags in the interrupt status registers identify maskable interrupt sources. Table 13-3 summarizes the interrupt sources and the interrupt status register flags that they set. The interrupt status registers can be useful for debugging.

Table 13-3. Interrupt Sources

Priority	Source	Flag	Mask[1]	INT Register Flag	Vector Address
Highest	Reset	—	—	—	$FFFE–$FFFF
↑	SWI instruction	—	—	—	$FFFC–$FFFD
↑	$\overline{\text{IRQ}}$ pin	IRQF1	IMASK1	IF1	$FFFA–$FFFB
↑	Timer channel 0 interrupt	CH0F	CH0IE	IF3	$FFF6–$FFF7
↑	Timer channel 1 interrupt	CH1F	CH1IE	IF4	$FFF4–$FFF5
↑	Timer overflow interrupt	TOF	TOIE	IF5	$FFF2–$FFF3
↑	SLIC interrupt	SLCF	SLCIE	IF9	$FFEA–$FFEB
↑	Keyboard interrupt	KEYF	IMASKK	IF14	$FFE0–$FFE1
Lowest	ADC conversion complete interrupt	COCO	AIEN	IF15	$FFDE–$FFDF

1. The I bit in the condition code register is a global mask for all interrupt sources except the SWI instruction.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF6	IF5	IF4	IF3	0	IF1	0	0
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 13-11. Interrupt Status Register 1 (INT1)

IF1 and IF3–IF6 — Interrupt Flags

These flags indicate the presence of interrupt requests from the sources shown in Table 13-3.

1 = Interrupt request present
0 = No interrupt request present

Bit 0, 1, and 3— Always read 0

13.6.2.1 Interrupt Status Register 2

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	IF14	IF13	IF12	IF11	IF10	IF9	IF8	IF7
Write:	R	R	R	R	R	R	R	R
Reset:	0	0	0	0	0	0	0	0

R = Reserved

Figure 13-12. Interrupt Status Register 2 (INT2)

IF7–IF14 — Interrupt Flags

This flag indicates the presence of interrupt requests from the sources shown in Table 13-3.

1 = Interrupt request present
0 = No interrupt request present

14.4 Interrupts

The SLIC module contains one interrupt vector, which can be triggered by sources encoded in the SLIC state vector register. See 14.8.6 SLIC State Vector Register.

14.5 Modes of Operation

Figure 14-3 shows the modes in which the SLIC will operate.

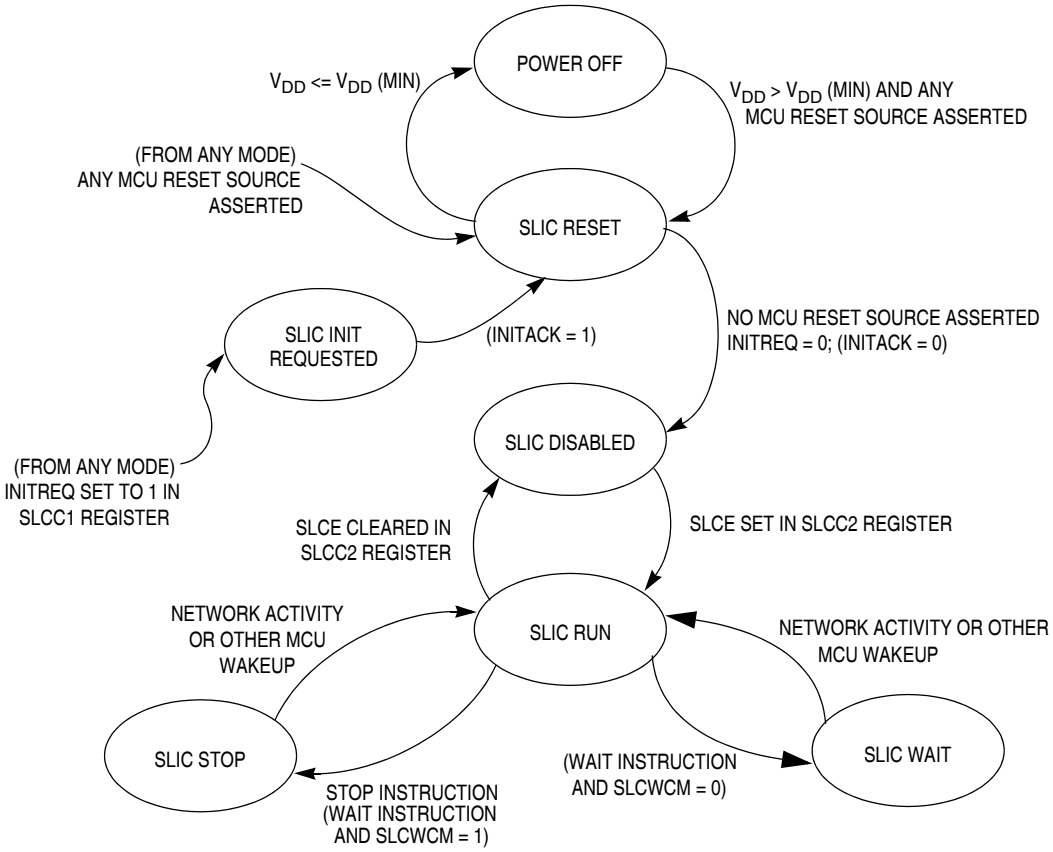


Figure 14-3. SLIC Operating Modes

14.5.1 Power Off

This mode is entered from the reset mode whenever the SLIC module supply voltage V_{DD} drops below its minimum specified value for the SLIC module to guarantee operation. The SLIC module will be placed in the reset mode by a system low-voltage reset (LVR) before being powered down. In this mode, the pin input and output specifications are not guaranteed.

14.5.2 Reset

This mode is entered from the power off mode whenever the SLIC module supply voltage V_{DD} rises above its minimum specified value ($V_{DD(MIN)}$) and some MCU reset source is asserted. To prevent the SLIC from entering an unknown state, the internal MCU reset is asserted while powering up the SLIC module. SLIC reset mode is also entered from any other mode as soon as one of the MCU's possible reset sources (e.g.,

instruction the CPU executes to cause the SLIC module to enter SLIC stop, the message which wakes up the SLIC module (and the CPU) may or may not be received.

There are two different possibilities:

1. Wakeup from SLIC Stop with CPU in STOP

   When the CPU executes the STOP instruction, all clocks in the MCU, including clocks to the SLIC module, are turned off. Therefore, the message which wakes up the SLIC module and the CPU from stop mode will not be received. This is due primarily to the amount of time required for the MCU's oscillator to stabilize before the clocks can be applied internally to the other MCU modules, including the SLIC module.

2. Wakeup from SLIC Stop with CPU in WAIT

   If the CPU executes the WAIT instruction and the SLIC module enters the stop mode (SLCWCM = 1), the clocks to the SLIC module are turned off, but the clocks in the MCU continue to run. Therefore, the message which wakes up the SLIC module from stop and the CPU from wait mode will be received correctly by the SLIC module. This is because very little time is required for the CPU to turn the clocks to the SLIC module back on after the wakeup interrupt occurs.

NOTE

While the SLIC module will correctly receive a message which arrives when the SLIC module is in stop or wait mode and the MCU is in wait mode, if the user enters this mode while a message is being received, the data in the message will become corrupted. This is due to the steps required for the SLIC module to resume operation upon exiting stop or wait mode, and its subsequent resynchronization with the LIN bus.

14.5.8 Normal and Emulation Mode Operation

The SLIC module operates in the same manner in all normal and emulation modes. All SLIC module registers can be read and written except those that are reserved, unimplemented, or write once. The user must be careful not to unintentionally write a register when using 16-bit writes to avoid unexpected SLIC module behavior.

14.5.9 Special Mode Operation

Some aspects of SLIC module operation can be modified in special test mode. This mode is reserved for internal use only.

14.5.10 Low-Power Options

The SLIC module can save power in disabled, wait, and stop modes. A complete description of what the SLIC module does while in a low-power mode can be found in 14.5 Modes of Operation.

14.6 SLIC During Break Interrupts

The BCFE bit in the BSCR register has no affect on the SLIC module. Therefore the SLIC modules status bits cannot be protected during break.

Table 14-2. Interrupt Sources Summary (BTM = 0) (Continued)

SLCSV	I3	I2	I1	I0	Interrupt Source	Priority
$1C	0	1	1	1	Receiver Buffer Overrun	7
$20	1	0	0	0	Reserved	8
$24	1	0	0	1	Checksum Error	9
$28	1	0	1	0	Byte Framing Error	10
$2C	1	0	1	1	Identifier Received Successfully	11
$30	1	1	0	0	Identifier Parity Error	12
$34	1	1	0	1	Inconsistent-Synch-Field-Error	13
$38	1	1	1	0	Reserved	14
$3C	1	1	1	1	Wakeup	15 (Highest)

- No Interrupts Pending
  This value indicates that all pending interrupt sources have been serviced. In polling mode, the SLCSV is read and interrupts serviced until this value reads back 0. This source will not generate an interrupt of the CPU, regardless of state of SLCIE.
- No Bus Activity (LIN specified error)
  The No-Bus-Activity condition occurs if no valid SYNCH BREAK FIELD or BYTE FIELD was received for more than 2^{23} SLIC clock counts since the reception of the last valid message. For example, with 6.4 MHz SLIC clock frequency, a No-Bus-Activity interrupt will occur about 1.31 seconds after the bus begins to idle.
- TX Message Buffer Empty — Checksum Transmitted
  When the entire LIN message frame has been transmitted successfully, complete with the appropriately selected checksum byte, this interrupt source is asserted. This source is used for all standard LIN message frames and the final set of bytes with extended LIN message frames.
- TX Message Buffer Empty
  This interrupt source indicates that all 8 bytes in the LIN message buffer have been transmitted with no checksum appended. This source is used for intermediate sets of 8 bytes in extended LIN message frames.
- RX Message Buffer Full — Checksum OK
  When the entire LIN message frame has been received successfully, complete with the appropriately selected checksum byte, and the checksum calculates correctly, this interrupt source is asserted. This source is used for all standard LIN message frames and the final set of bytes with extended LIN message frames. To clear this source, SLCD0 must be read first.
- RX Data Buffer Full — No Errors
  This interrupt source indicates that 8 bytes have been received with no checksum byte and are waiting in the LIN message buffer. This source is used for intermediate sets of 8 bytes in extended LIN message frames. To clear this source, SLCD0 must be read first.
- Bit Error
  A unit that is sending a bit on the bus also monitors the bus. A BIT_ERROR must be detected at that bit time, when the bit value that is monitored is different from the bit value that is sent. The SLIC will terminate the data transmission upon detection of a bit error, according to the LIN

14.8.8 SLIC Identifier and Data Registers

The SLIC identifier (SLCID) and eight data registers (SLCD7–SLCD0) comprise the transmit and receive buffer and are used to read/write the identifier and message buffer 8 data bytes. In BTM mode (BTM = 1), only SLCID is used to send and receive bytes, as only one byte is handled at any one time. The number of bytes to be read from or written to these registers is determined by the user software and written to SLCDLC. To obtain proper data, reads and writes to these registers must be made based on the proper length corresponding to a particular message. It is the responsibility of the user software to keep track of this value to prevent data corruption. For example, it is possible to read data from locations in the message buffer which contain erroneous or old data if the user software reads more data registers than were updated by the incoming message, as indicated in SLCDLC.

NOTE

An incorrect length value written to SLCDLC can result in the user software misreading or miswriting data in the message buffer. An incorrect length value might also result in SLIC error messages. For example, if a 4-byte message is to be received, but the user software incorrectly reports a 3-byte length to the DLC, the SLIC will assume the 4th data byte is actually a checksum value and attempt to validate it as such. If this value doesn't match the calculated value, an incorrect checksum error will occur. If it does happen to match the expected value, then the message would be received as a 3-byte message with valid checksum. Either case is incorrect behavior for the application and can be avoided by ensuring that the correct length code is used for each identifier.

The first data byte received after the LIN identifier in a LIN message frame will be loaded into SLCD0. The next byte (if applicable) will be loaded into SLCD1, and so forth.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
Reset:	0	0	0	0	0	0	0	0

Figure 14-12. SLIC Identifier Register (SLCID)

The SLIC identifier register is used to capture the incoming LIN identifier and when the SLCSV value indicates that the identifier has been received successfully, this register contains the received identifier value. If the incoming identifier contained a parity error, this register value will not contain valid data.

In byte transfer mode (BTM = 1), this register is used for sending and receiving each byte of data. When transmitting bytes, the data is loaded into this register, then TXGO in SLCDLC is set to initiate the transmission. When receiving bytes, they are read from this register only.

	Bit 7	6	5	4	3	2	1	Bit 0
Read:	R7	R6	R5	R4	R3	R2	R1	R0
Write:	T7	T6	T5	T4	T3	T2	T1	T0
Reset:	0	0	0	0	0	0	0	0

Figure 14-13. SLIC Data Register x (SLCD7–SLCD0)

R — Read SLC Receive Data

T — Write SLC Transmit Data

data bytes will fit in the buffer and only one interrupt should occur. At this time, the final interrupt may be handled normally, continuing to use the software counter to read the proper number of bytes from the appropriate SLCD registers.

NOTE

Do not write SLCDLC more than one time per LIN message frame. The SLIC tracks the number of sent or received bytes based on the value written to this register at the beginning of the data field and rewriting this register will corrupt the checksum calculation and cause unpredictable behavior in the SLIC module. The application software must track the number of sent or received bytes to know what the current byte count in the SLIC is. If programming in C, make sure to use the VOLATILE modifier on this variable (or make it a global variable) to ensure that it keeps its value between interrupts.

14.9.8.3 Possible Errors on Command Message Data

Possible errors on command message data are:

- Byte Framing Error
- Checksum-Error (LIN specified error)
- No-Bus-Activity (LIN specified error)
- Receiver Buffer Overrun Error

14.9.9 Handling Request LIN Message Frames

Figure 14-17 shows how to handle request message frames, where the SLIC module is sending data to the master node.

Request message frames refer to LIN messages frames where the master node is “requesting” the slave node to supply information. The implication is that the slave will then be transmitting data to the master for this message frame. This can be a standard LIN message frame of 1–8 data bytes, a reserved LIN system message (using 0x3D identifier), or an extended request message frame utilizing the reserved 0x3E identifier or perhaps the 0x3F LIN reserved extended identifier. The SLIC module is capable of handling request message frames containing up to 64 bytes of data, while still automatically calculating and/or verifying the checksum.

14.9.9.1 Standard Request Message Frames

Dealing with request messages with the SLIC is very similar to dealing with command messages, with one important difference. Because the SLIC is now to be transmitting data in the LIN message frame, the user software must load the data to be transmitted into the message buffer prior to initiating the transmission. This means an extra step is taken inside the interrupt service routine after the identifier has been decoded and is determined to be an ID for a request message frame.

Figure 14-17 deals with request messages, where the SLIC will be transmitting data to the master node. If the received identifier corresponds to a standard LIN command frame (i.e., 1-8 data bytes), the message processing is very simple. The user must load the data to be transmitted into the transmit buffer by writing it to the SLCD registers. The first byte to be transmitted on the LIN bus must be loaded into SLCD0, then SLCD1 for the second byte, etc. After all of the bytes to be transmitted are loaded in this way, a single write to SLCDLC will allow the user to encode the number of data bytes to be transmitted (1–8 bytes for standard request frames), set the proper checksum calculation method for the data

will resume. If the SLIC was in SLIC stop mode, SLCSV will indicate wakeup as the interrupt source so that the user knows that the SLIC module brought the MCU out of stop or wait.

In a LIN system, a system message is generally sent to all nodes to indicate that they are to enter low-power network sleep mode. After a node enters sleep mode, it waits for outside events, such as switch or sensor inputs or network traffic to bring it out of network sleep mode. If the node using the SLIC module is awakened by a source other than network traffic, such as a switch input, the LIN specification requires this node to issue a wake-up signal to the rest of the network. The SLIC module supports this feature using WAKETX in SLCC2. The user software may set this bit and one LIN wake-up signal is immediately transmitted on the bus, then the bit is automatically cleared by the SLIC module. If another wake-up signal is required to be sent, the user must set WAKETX again.

In a LIN system, the LIN physical interface can often also provide an output to the $\overline{\text{IRQ}}$ pin to provide a wake-up mechanism on network activity. The physical layer might also control voltage regulation supply to the MCU, cutting power to the MCU when the physical layer is placed in its low-power mode. The user must take care to ensure that the interaction between the physical layer, $\overline{\text{IRQ}}$ pin, SLIC transmit and receive pins, and power supply regulator is fully understood and designed to ensure proper operation.

14.9.12 Polling Operation

It is possible to operate the SLIC module in polling mode, if desired. The primary difference is that the SLIC interrupt request should not be enabled (SLCIE = 0). The SLCSV will update and operate properly and interrupt requests will be indicated with the SLCF flag, which can be polled to determine status changes in the SLIC module. It is required that the polling rate be fast enough to ensure that SLIC status changes be recognized and processed in time to ensure that all application timings can be met.

14.9.13 LIN Data Integrity Checking Methods

The SLIC module supports two different LIN-based data integrity options:

- The first option supports LIN 1.3 and older methods of checksum calculations.
- The second option supports an optional additional enhanced checksum calculation which has greater data integrity coverage.

The LIN 1.3 and earlier specifications transmit a checksum byte in the “CHECKSUM FIELD” of the LIN message frame. This CHECKSUM FIELD contains the inverted modulo-256 sum over all data bytes. The sum is calculated by an “ADD with Carry” where the carry bit of each addition is added to the least significant bit (LSB) of its resulting sum. This guarantees security also for the MSBs of the data bytes. The sum of modulo-256 sum over all data bytes and the checksum byte must be ’0xFF’.

An optional checksum calculation can also be performed on a LIN data frame which is very similar to the LIN 1.3 calculation, but with one important distinction. This enhanced calculation simply includes the identifier field as the first value in the calculation, whereas the LIN 1.3 calculation begins with the least significant byte of the data field (which is the first byte to be transmitted on the bus). This enhanced calculation further ensures that the identifier field is correct and ties the identifier and data together under a common calculation, ensuring greater reliability.

In the SLIC module, either checksum calculation can be performed on any given message frame by simply writing or clearing CHKMOD in SLCDLC, as desired, when the identifier for the message frame is decoded. The appropriate calculation for each message frame should be decided at system design time and documented in the LIN description file, indicating to the user which calculation to use for a particular identifier.

Table 16-4. WRITE (Write Memory) Command

Description	Write byte to memory
Operand	2-byte address in high-byte:low-byte order; low byte followed by data byte
Data Returned	None
Opcode	$49

Command Sequence

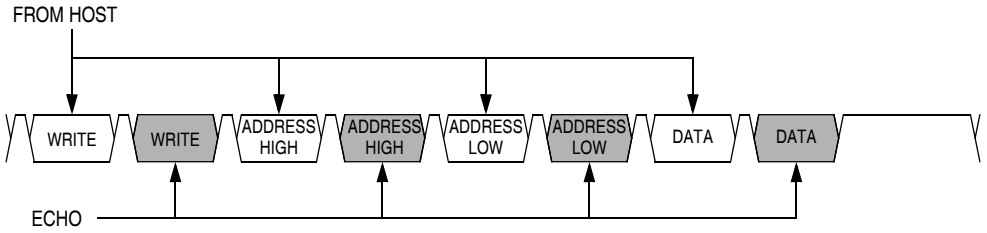


Table 16-5. IREAD (Indexed Read) Command

Description	Read next 2 bytes in memory from last address accessed
Operand	None
Data Returned	Returns contents of next two addresses
Opcode	$1A

Command Sequence

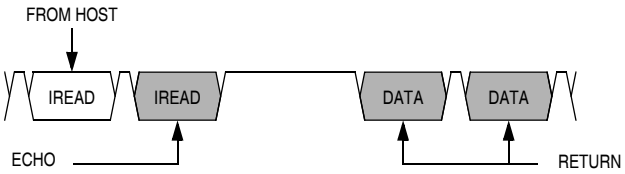
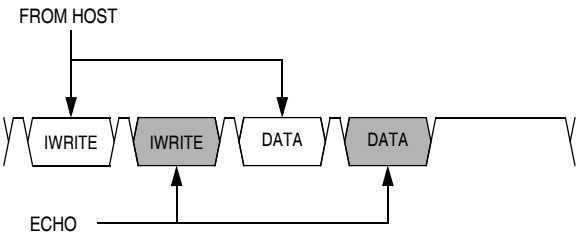


Table 16-6. IWRITE (Indexed Write) Command

Description	Write to last address accessed + 1
Operand	Single data byte
Data Returned	None
Opcode	$19

Command Sequence



A sequence of IREAD or IWRITE commands can access a block of memory sequentially over the full 64-Kbyte memory map.

17.3 Functional Operating Range

Characteristic	Symbol	Value	Unit	Temperature Code
Operating temperature range	T_A (T_L to T_H)	−40 to +125 −40 to +105 −40 to +85	°C	M V C
Operating voltage range	V_{DD}	3.0 to 5.5	V	—

17.4 Thermal Characteristics

Characteristic	Symbol	Value	Unit
Thermal resistance 16-pin SOIC 16-pin TSSOP	θ_{JA}	 90 133	°C/W
I/O pin power dissipation	$P_{I/O}$	User determined	W
Power dissipation(1)	P_D	$P_D = (I_{DD} \times V_{DD})$ + $P_{I/O} = K/(T_J + 273°C)$	W
Constant(2)	K	$P_D \times (T_A + 273°C)$ + $P_D^2 \times \theta_{JA}$	W/°C
Average junction temperature	T_J	$T_A + (P_D \times \theta_{JA})$	°C
Maximum junction temperature	T_{JM}	150	°C

1. Power dissipation is a function of temperature.
2. K constant unique to the device. K can be determined for a known T_A and measured P_D. With this value of K, P_D and T_J can be determined for any value of T_A.

17.5 5-V DC Electrical Characteristics

Characteristic(1)	Symbol	Min	Typ(2)	Max	Unit
Output high voltage I_{Load} = –2.0 mA, all I/O pins I_{Load} = –10.0 mA, all I/O pins I_{Load} = –15.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OH}	 V_{DD}–0.4 V_{DD}–1.5 V_{DD}–0.8	 — — —	 — — —	V
Maximum combined I_{OH} (all I/O pins)	I_{OHT}	—	—	50	mA
Output low voltage I_{Load} = 1.6 mA, all I/O pins I_{Load} = 10.0 mA, all I/O pins I_{Load} = 15.0 mA, PTA0, PTA1, PTA3–PTA5 only	V_{OL}	 — — —	 — — —	 0.4 1.5 0.8	V
Maximum combined I_{OL} (all I/O pins)	I_{OHL}	—	—	50	mA
Input high voltage PTA0–PTA5, PTB0–PTB7	V_{IH}	0.7 x V_{DD}	—	V_{DD}	V

Table continued on next page.

17.14 Timer Interface Module Characteristics

Characteristic	Symbol	Min	Max	Unit
Timer input capture pulse width(1)	t_{TH}, t_{TL}	2	—	t_{cyc}
Timer input capture period	t_{TLTL}	Note(2)	—	t_{cyc}
Timer input clock pulse width(1)	t_{TCL}, t_{TCH}	t_{cyc} + 5	—	ns

1. Values are based on characterization results, not tested in production.
2. The minimum period is the number of cycles it takes to execute the interrupt service routine plus 1 t_{cyc}.

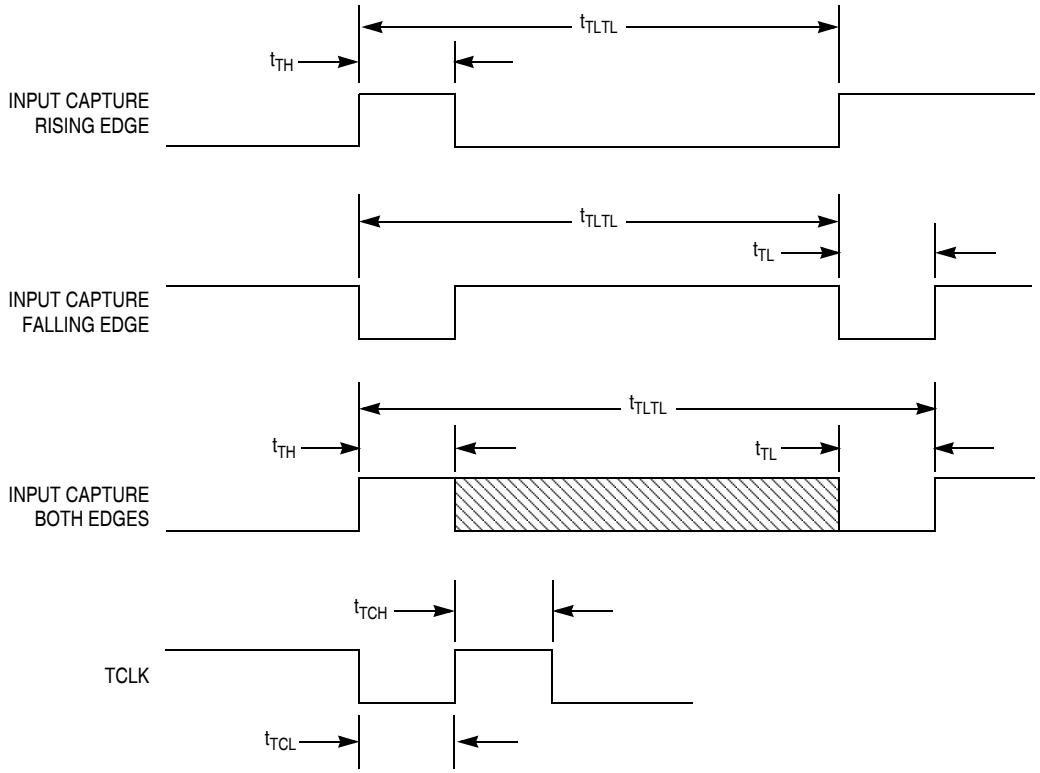


Figure 17-11. Input Capture Timing