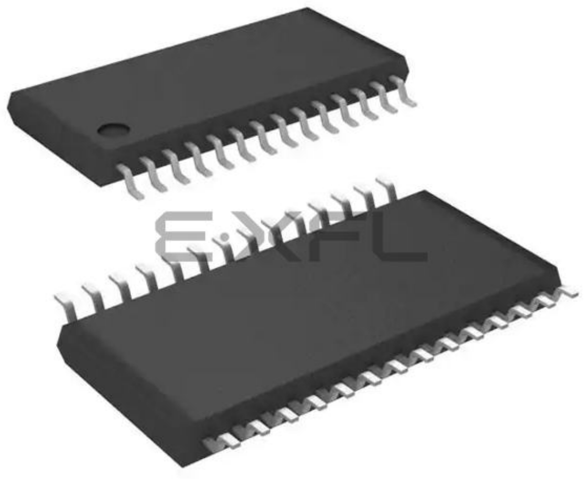
NXP USA Inc. - S9S08SL16F1MTL Datasheet



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, 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	256 x 8
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 125°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/s9s08sl16f1mtl

6.5.2 Port B Registers 87
6.5.3 Port C Registers 91

Chapter 7
Central Processor Unit (S08CPUV3)

7.1 Introduction 95
7.1.1 Features 95
7.2 Programmer's Model and CPU Registers 96
7.2.1 Accumulator (A) 96
7.2.2 Index Register (H:X) 96
7.2.3 Stack Pointer (SP) 97
7.2.4 Program Counter (PC) 97
7.2.5 Condition Code Register (CCR) 97
7.3 Addressing Modes 99
7.3.1 Inherent Addressing Mode (INH) 99
7.3.2 Relative Addressing Mode (REL) 99
7.3.3 Immediate Addressing Mode (IMM) 99
7.3.4 Direct Addressing Mode (DIR) 99
7.3.5 Extended Addressing Mode (EXT) 100
7.3.6 Indexed Addressing Mode 100
7.4 Special Operations 101
7.4.1 Reset Sequence 101
7.4.2 Interrupt Sequence 101
7.4.3 Wait Mode Operation 102
7.4.4 Stop Mode Operation 102
7.4.5 BGND Instruction 103
7.5 HCS08 Instruction Set Summary 103

Chapter 8
Internal Clock Source (S08ICSV2)

8.1 Introduction 115
8.1.1 Module Configuration 115
8.1.2 Features 117
8.1.3 Block Diagram 117
8.1.4 Modes of Operation 118
8.2 External Signal Description 119
8.3 Register Definition 119
8.3.1 ICS Control Register 1 (ICSC1) 120
8.3.2 ICS Control Register 2 (ICSC2) 121
8.3.3 ICS Trim Register (ICSTRM) 122
8.3.4 ICS Status and Control (ICSSC) 122
8.4 Functional Description 123

NOTE

In EMC-sensitive applications, use an external RC filter on $\overline{\text{RESET}}$. See Figure 2-3 for an example.

2.2.4 Background / Mode Select (BKGD/MS)

While in reset, the BKGD/MS pin functions as a mode select pin. Immediately after reset rises, the pin functions as the background pin and can be used for background debug communication. While functioning as a background or mode select pin, the pin includes an internal pull-up device, input hysteresis, a standard output driver, and no output slew rate control.

If nothing is connected to this pin, the MCU will enter normal operating mode at the rising edge of reset. If a debug system is connected to the 6-pin standard background debug header, it can hold BKGD low during the rising edge of reset which forces the MCU to active background mode.

The BKGD/MS pin is used primarily for background debug controller (BDC) communications using a custom protocol that uses 16 clock cycles of the target MCU's BDC clock per bit time. The target MCU's BDC clock could be as fast as the bus clock rate, so there should never be any significant capacitance connected to the BKGD/MS pin that could interfere with background serial communications.

Although the BKGD/MS pin is a pseudo open-drain pin, the background debug communication protocol provides brief, actively driven, high speedup pulses to ensure fast rise times. Small capacitances from cables and the absolute value of the internal pull-up device play almost no role in determining rise and fall times on the BKGD/MS pin.

2.2.5 General-Purpose I/O and Peripheral Ports

The MC9S08EL32 Series and MC9S08SL16 Series of MCUs support up to 22 general-purpose I/O pins which are shared with on-chip peripheral functions (timers, serial I/O, ADC, etc.).

When a port pin is configured as a general-purpose output or a peripheral uses the port pin as an output, software can select one of two drive strengths and enable or disable slew rate control. When a port pin is configured as a general-purpose input or a peripheral uses the port pin as an input, software can enable a pull-up device. Immediately after reset, all of these pins are configured as high-impedance general-purpose inputs with internal pull-up devices disabled.

When an on-chip peripheral system is controlling a pin, data direction control bits still determine what is read from port data registers even though the peripheral module controls the pin direction by controlling the enable for the pin's output buffer. For information about controlling these pins as general-purpose I/O pins, see Chapter 6, "Parallel Input/Output Control."

NOTE

To avoid extra current drain from floating input pins, the reset initialization routine in the application program should either enable on-chip pull-up devices or change the direction of unused or non-bonded pins to outputs so they do not float.

6.3 Pin Interrupts

Port A[3:0], port B[3:0] and port C pins can be configured as external interrupt inputs and as an external mean of waking the MCU from stop3 or wait low-power modes.

The block diagram for each port interrupt logic is shown Figure 6-2.

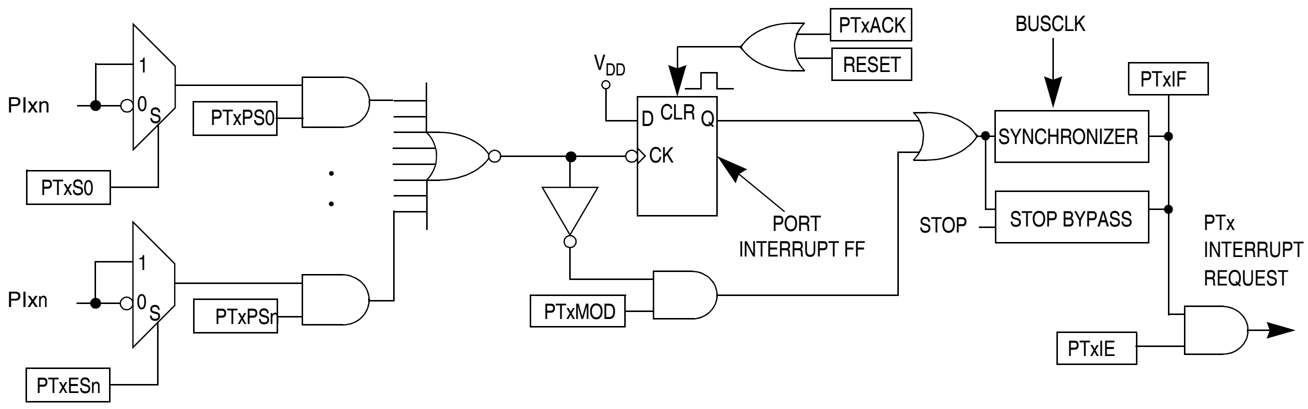


Figure 6-2. Port Interrupt Block Diagram

Writing to the PTxPSn bits in the port interrupt pin select register (PTxPS) independently enables or disables each port pin. Each port can be configured as edge sensitive or edge and level sensitive based on the PTxMOD bit in the port interrupt status and control register (PTxSC). Edge sensitivity can be software programmed to be either falling or rising; the level can be either low or high. The polarity of the edge or edge and level sensitivity is selected using the PTxESn bits in the port interrupt edge select register (PTxES).

Synchronous logic is used to detect edges. Prior to detecting an edge, enabled port inputs must be at the deasserted logic level. A falling edge is detected when an enabled port input signal is seen as a logic 1 (the deasserted level) during one bus cycle and then a logic 0 (the asserted level) during the next cycle. A rising edge is detected when the input signal is seen as a logic 0 during one bus cycle and then a logic 1 during the next cycle.

6.3.1 Edge Only Sensitivity

A valid edge on an enabled port pin will set PTxIF in PTxSC. If PTxIE in PTxSC is set, an interrupt request will be presented to the CPU. Clearing of PTxIF is accomplished by writing a 1 to PTxACK in PTxSC.

6.3.2 Edge and Level Sensitivity

A valid edge or level on an enabled port pin will set PTxIF in PTxSC. If PTxIE in PTxSC is set, an interrupt request will be presented to the CPU. Clearing of PTxIF is accomplished by writing a 1 to PTxACK in PTxSC provided all enabled port inputs are at their deasserted levels. PTxIF will remain set if any enabled port pin is asserted while attempting to clear by writing a 1 to PTxACK.

6.5.2.3 Port B Pull Enable Register (PTBPE)

	7	6	5	4	3	2	1	0
R W	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-13. Internal Pull Enable for Port B Register (PTBPE)

Table 6-11. PTBPE Register Field Descriptions

Field	Description
7:0 PTBPE[7:0]	**Internal Pull Enable for Port B Bits** — Each of these control bits determines if the internal pull-up or internal (pin interrupt only) pull-down device is enabled for the associated PTB pin. For port B pins that are configured as outputs, these bits have no effect and the internal pull devices are disabled. 0 Internal pull-up/pull-down device disabled for port B bit n. 1 Internal pull-up/pull-down device enabled for port B bit n.

6.5.2.4 Port B Slew Rate Enable Register (PTBSE)

	7	6	5	4	3	2	1	0
R W	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
Reset:	0	0	0	0	0	0	0	0

Figure 6-14. Slew Rate Enable for Port B Register (PTBSE)

Table 6-12. PTBSE Register Field Descriptions

Field	Description
7:0 PTBSE[7:0]	**Output Slew Rate Enable for Port B Bits** — Each of these control bits determines if the output slew rate control is enabled for the associated PTB pin. For port B pins that are configured as inputs, these bits have no effect. 0 Output slew rate control disabled for port B bit n. 1 Output slew rate control enabled for port B bit n.

Chapter 7
Central Processor Unit (S08CPUV3)

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

interrupt service routine, this would allow nesting of interrupts (which is not recommended because it leads to programs that are difficult to debug and maintain).

For compatibility with the earlier M68HC05 MCUs, the high-order half of the H:X index register pair (H) is not saved on the stack as part of the interrupt sequence. The user must use a PSHH instruction at the beginning of the service routine to save H and then use a PULH instruction just before the RTI that ends the interrupt service routine. It is not necessary to save H if you are certain that the interrupt service routine does not use any instructions or auto-increment addressing modes that might change the value of H.

The software interrupt (SWI) instruction is like a hardware interrupt except that it is not masked by the global I bit in the CCR and it is associated with an instruction opcode within the program so it is not asynchronous to program execution.

7.4.3 Wait Mode Operation

The WAIT instruction enables interrupts by clearing the I bit in the CCR. It then halts the clocks to the CPU to reduce overall power consumption while the CPU is waiting for the interrupt or reset event that will wake the CPU from wait mode. When an interrupt or reset event occurs, the CPU clocks will resume and the interrupt or reset event will be processed normally.

If a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in wait mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in wait mode.

7.4.4 Stop Mode Operation

Usually, all system clocks, including the crystal oscillator (when used), are halted during stop mode to minimize power consumption. In such systems, external circuitry is needed to control the time spent in stop mode and to issue a signal to wake up the target MCU when it is time to resume processing. Unlike the earlier M68HC05 and M68HC08 MCUs, the HCS08 can be configured to keep a minimum set of clocks running in stop mode. This optionally allows an internal periodic signal to wake the target MCU from stop mode.

When a host debug system is connected to the background debug pin (BKGD) and the ENBDM control bit has been set by a serial command through the background interface (or because the MCU was reset into active background mode), the oscillator is forced to remain active when the MCU enters stop mode. In this case, if a serial BACKGROUND command is issued to the MCU through the background debug interface while the CPU is in stop mode, CPU clocks will resume and the CPU will enter active background mode where other serial background commands can be processed. This ensures that a host development system can still gain access to a target MCU even if it is in stop mode.

Recovery from stop mode depends on the particular HCS08 and whether the oscillator was stopped in stop mode. Refer to the Modes of Operation chapter for more details.

Table 7-2. Instruction Set Summary (Sheet 4 of 9)

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affect on CCR	
						V 1 1 H	I N Z C
CLC	Clear Carry Bit (C ← 0)	INH	98	1	p	– 1 1 –	– – – 0
CLI	Clear Interrupt Mask Bit (I ← 0)	INH	9A	1	p	– 1 1 –	0 – – –
CLR *opr8a* CLRA CLRX CLRH CLR *oprx8*,X CLR ,X CLR *oprx8*,SP	Clear M ← $00 A ← $00 X ← $00 H ← $00 M ← $00 M ← $00 M ← $00	DIR INH INH INH IX1 IX SP1	3F dd 4F 5F 8C 6F ff 7F 9E 6F ff	5 1 1 1 5 4 6	rfwpp p p p rfwpp rfwp prfwpp	0 1 1 –	– 0 1 –
CMP #*opr8i* CMP *opr8a* CMP *opr16a* CMP *oprx16*,X CMP *oprx8*,X CMP ,X CMP *oprx16*,SP CMP *oprx8*,SP	Compare Accumulator with Memory A – M (CCR Updated But Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 ii B1 dd C1 hh ll D1 ee ff E1 ff F1 9E D1 ee ff 9E E1 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↕ 1 1 –	– ↕ ↕ ↕
COM *opr8a* COMA COMX COM *oprx8*,X COM ,X COM *oprx8*,SP	Complement M ← $(\overline{M})$= $FF – (M) (One's Complement) A ← $(\overline{A})$ = $FF – (A) X ← $(\overline{X})$ = $FF – (X) M ← $(\overline{M})$ = $FF – (M) M ← $(\overline{M})$ = $FF – (M) M ← $(\overline{M})$ = $FF – (M)	DIR INH INH IX1 IX SP1	33 dd 43 53 63 ff 73 9E 63 ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	0 1 1 –	– ↕ ↕ 1
CPHX *opr16a* CPHX #*opr16i* CPHX *opr8a* CPHX *oprx8*,SP	Compare Index Register (H:X) with Memory (H:X) – (M:M + $0001) (CCR Updated But Operands Not Changed)	EXT IMM DIR SP1	3E hh ll 65 jj kk 75 dd 9E F3 ff	6 3 5 6	prrfpp ppp rrfpp prrfpp	↕ 1 1 –	– ↕ ↕ ↕
CPX #*opr8i* CPX *opr8a* CPX *opr16a* CPX *oprx16*,X CPX *oprx8*,X CPX ,X CPX *oprx16*,SP CPX *oprx8*,SP	Compare X (Index Register Low) with Memory X – M (CCR Updated But Operands Not Changed)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A3 ii B3 dd C3 hh ll D3 ee ff E3 ff F3 9E D3 ee ff 9E E3 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	↕ 1 1 –	– ↕ ↕ ↕
DAA	Decimal Adjust Accumulator After ADD or ADC of BCD Values	INH	72	1	p	U 1 1 –	– ↕ ↕ ↕
DBNZ *opr8a*,*rel* DBNZA *rel* DBNZX *rel* DBNZ *oprx8*,X,*rel* DBNZ ,X,*rel* DBNZ *oprx8*,SP,*rel*	Decrement A, X, or M and Branch if Not Zero (if (result) ≠ 0) DBNZX Affects X Not H	DIR INH INH IX1 IX SP1	3B dd rr 4B rr 5B rr 6B ff rr 7B rr 9E 6B ff rr	7 4 4 7 6 8	rfwpppp fppp fppp rfwpppp rfwppp prfwpppp	– 1 1 –	– – – –
DEC *opr8a* DECA DECX DEC *oprx8*,X DEC ,X DEC *oprx8*,SP	Decrement M ← (M) – $01 A ← (A) – $01 X ← (X) – $01 M ← (M) – $01 M ← (M) – $01 M ← (M) – $01	DIR INH INH IX1 IX SP1	3A dd 4A 5A 6A ff 7A 9E 6A ff	5 1 1 5 4 6	rfwpp p p rfwpp rfwp prfwpp	↕ 1 1 –	– ↕ ↕ –

	7	6	5	4	3	2	1	0
R	COCO	AIEN	ADCO	ADCH				
W	▨							
Reset:	0	0	0	1	1	1	1	1

▨ = Unimplemented or Reserved

Figure 10-3. Status and Control Register (ADCSC1)

Table 10-3. ADCSC1 Register Field Descriptions

Field	Description
7 COCO	**Conversion Complete Flag** — The COCO flag is a read-only bit which is set each time a conversion is completed when the compare function is disabled (ACFE = 0). When the compare function is enabled (ACFE = 1) the COCO flag is set upon completion of a conversion only if the compare result is true. This bit is cleared whenever ADCSC1 is written or whenever ADCRL is read. 0 Conversion not completed 1 Conversion completed
6 AIEN	**Interrupt Enable** — AIEN is used to enable conversion complete interrupts. When COCO becomes set while AIEN is high, an interrupt is asserted. 0 Conversion complete interrupt disabled 1 Conversion complete interrupt enabled
5 ADCO	**Continuous Conversion Enable** — ADCO is used to enable continuous conversions. 0 One conversion following a write to the ADCSC1 when software triggered operation is selected, or one conversion following assertion of ADHWT when hardware triggered operation is selected. 1 Continuous conversions initiated following a write to ADCSC1 when software triggered operation is selected. Continuous conversions are initiated by an ADHWT event when hardware triggered operation is selected.
4:0 ADCH	**Input Channel Select** — The ADCH bits form a 5-bit field which is used to select one of the input channels. The input channels are detailed in Figure 10-4. The successive approximation converter subsystem is turned off when the channel select bits are all set to 1. This feature allows for explicit disabling of the ADC and isolation of the input channel from all sources. Terminating continuous conversions this way will prevent an additional, single conversion from being performed. It is not necessary to set the channel select bits to all 1s to place the ADC in a low-power state when continuous conversions are not enabled because the module automatically enters a low-power state when a conversion completes.

Figure 10-4. Input Channel Select

ADCH	Input Select	ADCH	Input Select
00000	AD0	10000	AD16
00001	AD1	10001	AD17
00010	AD2	10010	AD18
00011	AD3	10011	AD19
00100	AD4	10100	AD20
00101	AD5	10101	AD21
00110	AD6	10110	AD22
00111	AD7	10111	AD23

In cases where separate power supplies are used for analog and digital power, the ground connection between these supplies must be at the V_{SSAD} pin. This should be the only ground connection between these supplies if possible. The V_{SSAD} pin makes a good single point ground location.

10.6.1.2 Analog Reference Pins

In addition to the analog supplies, the ADC module has connections for two reference voltage inputs. The high reference is V_{REFH}, which may be shared on the same pin as V_{DDAD} on some devices. The low reference is V_{REFL}, which may be shared on the same pin as V_{SSAD} on some devices.

When available on a separate pin, V_{REFH} may be connected to the same potential as V_{DDAD}, or may be driven by an external source that is between the minimum V_{DDAD} spec and the V_{DDAD} potential (V_{REFH} must never exceed V_{DDAD}). When available on a separate pin, V_{REFL} must be connected to the same voltage potential as V_{SSAD}. Both V_{REFH} and V_{REFL} must be routed carefully for maximum noise immunity and bypass capacitors placed as near as possible to the package.

AC current in the form of current spikes required to supply charge to the capacitor array at each successive approximation step is drawn through the V_{REFH} and V_{REFL} loop. The best external component to meet this current demand is a 0.1 μF capacitor with good high frequency characteristics. This capacitor is connected between V_{REFH} and V_{REFL} and must be placed as near as possible to the package pins. Resistance in the path is not recommended because the current will cause a voltage drop which could result in conversion errors. Inductance in this path must be minimum (parasitic only).

10.6.1.3 Analog Input Pins

The external analog inputs are typically shared with digital I/O pins on MCU devices. The pin I/O control is disabled by setting the appropriate control bit in one of the pin control registers. Conversions can be performed on inputs without the associated pin control register bit set. It is recommended that the pin control register bit always be set when using a pin as an analog input. This avoids problems with contention because the output buffer will be in its high impedance state and the pullup is disabled. Also, the input buffer draws dc current when its input is not at either V_{DD} or V_{SS}. Setting the pin control register bits for all pins used as analog inputs should be done to achieve lowest operating current.

Empirical data shows that capacitors on the analog inputs improve performance in the presence of noise or when the source impedance is high. Use of 0.01 μF capacitors with good high-frequency characteristics is sufficient. These capacitors are not necessary in all cases, but when used they must be placed as near as possible to the package pins and be referenced to V_{SSA}.

For proper conversion, the input voltage must fall between V_{REFH} and V_{REFL}. If the input is equal to or exceeds V_{REFH}, the converter circuit converts the signal to \$3FF (full scale 10-bit representation) or \$FF (full scale 8-bit representation). If the input is equal to or less than V_{REFL}, the converter circuit converts it to \$000. Input voltages between V_{REFH} and V_{REFL} are straight-line linear conversions. There will be a brief current associated with V_{REFL} when the sampling capacitor is charging. The input is sampled for 3.5 cycles of the ADCK source when ADLSMP is low, or 23.5 cycles when ADLSMP is high.

For minimal loss of accuracy due to current injection, pins adjacent to the analog input pins should not be transitioning during conversions.

	7	6	5	4	3	2	1	0
R	0	0	INITREQ	BEDD	WAKETX	TXABRT	IMSG	SLCIE
W	▨	▨						
Reset	0	0	1	0	0	0	0	0

▨ = Unimplemented or Reserved

Figure 12-4. SLIC Control Register 1 (SLCC1)

Table 12-1. SLCC1 Field Descriptions

Field	Description
5 INITREQ	**Initialization Request** — Requesting initialization mode by setting this bit will place the SLIC module into its initialized state immediately. As a result of setting INITREQ, INITACK will be set in SLCS. INITACK = 1 causes all SLIC register bits (except SLCWCM: write once) to be held in their reset states and become not writable until INITACK has been cleared. If transmission or reception of data is in progress, the transaction will be terminated immediately upon entry into initialization mode (signified by INITACK being set to 1). To return to normal SLIC operation after the SLIC has been initialized (the INITACK is high), the INITREQ must be cleared by software. 0 Normal operation 1 Request for SLIC to be put into reset state immediately
4 BEDD	**BEDD Bit Error Detection Disable** — This bit allows the user to disable bit error detection circuitry. Bit error detection monitors the received bits to determine if they match the state of the corresponding transmitted bits. When bit error detection is enabled and a mismatch between transmitted bit and received bit is detected, a bit error is reported to the user through the SLCSV register and a SLIC interrupt is generated (if SLIC interrupts are enabled). The user must ensure that all physical delays which affect the timing of received bits are not significant enough to cause the bit error detection circuitry to incorrectly detect bit errors at higher LIN bus speeds. See Section 12.6.15, “Bit Error Detection and Physical Layer Delay,” for details. **NOTE** Bit Error detection is not recommended for use in BTM mode, as bit errors are reported on bit boundaries, not byte boundaries. This can result in misaligned data. Bit errors must not be disabled during normal LIN operations, as it allows the SLIC module to operate outside of the LIN specification. If you switch off bit error detection, there is no guaranteed way to detect bus collisions and automatically cease transmissions. Therefore pending SLIC transmissions may continue after a bit error should have been detected, potentially corrupting bus traffic. 0 Bit Error Detection Enabled 1 Bit Error Detection Disabled no bit errors will be detected or reported
3 WAKETX	**Transmit Wakeup Symbol**— This bit allows the user to transmit a wakeup symbol on the LIN bus. When set, this sends a wakeup symbol, as defined in the LIN specification a single time, then resets to 0. This bit will read 1 while the wakeup symbol is being transmitted on the bus. This bit will be automatically cleared when the wakeup symbol is complete. 0 Normal operation 1 Send wakeup symbol on LIN bus

Table 12-3. Digital Receive Filter Clock Prescaler

RXFP[2:0]	Digital RX Filter Clock Prescaler (Divide by)	Max Filter Delay (in µs)									
		Filter Input Clock (SLIC clock in MHz)									
		2	4	6	8	10	12	14	16	18	20
000	1	8.00	4.00	2.67	2.00	1.60	1.33	1.14	1.00	0.89	0.80
001	2	16.00	8.00	5.33	4.00	3.20	2.67	2.29	2.00	1.78	1.60
010	3	24.00	12.00	8.00	6.00	4.80	4.00	3.43	3.00	2.67	2.40
011	4	32.00	16.00	10.67	8.00	6.40	5.33	4.57	4.00	3.56	3.20
100	5	40.00	20.00	13.33	10.00	8.00	6.67	5.71	5.00	4.44	4.00
101	6	48.00	24.00	16.00	12.00	9.60	8.00	6.86	6.00	5.33	4.80
110	7	56.00	28.00	18.67	14.00	11.20	9.33	8.00	7.00	6.22	5.60
111	8	64.00	32.00	21.33	16.00	12.80	10.67	9.14	8.00	7.11	6.40

12.3.3 SLIC Bit Time Registers (SLCBTH, SLCBTL)

NOTE

In this subsection, the SLIC bit time registers are collectively referred to as SLCBT.

In LIN operating mode (BTM = 0), the SLCBT is updated by the SLIC upon reception of a LIN break-sync combination and provides the number of SLIC clock counts that equal one LIN bit time to the user software. This value can be used by the software to calculate the clock drift in the oscillator as an offset to a known count value (based on nominal oscillator frequency and LIN bus speed). The user software can then trim the oscillator to compensate for clock drift. Refer to Section 12.6.17, "Oscillator Trimming with SLIC," for more information on this procedure. The user should only read the bit time value from SLCBTH and SLCBTL in the interrupt service routine code for reception of the identifier byte. Reads at any other time during LIN activity may not provide reliable results.

When in byte transfer mode (BTM = 1), the SLCBT must be written by the user to set the length of one bit at the desired bit rate in SLIC clock counts. The user software must initialize this number prior to sending or receiving data, based on the input clock selection, prescaler stage choice, and desired bit rate. This setting is similar to choosing an input capture or output compare value for a timer. A write to both registers is required to update the bit time value.

NOTE

The SLIC bit time will not be updated until a write of the SLCBTL has occurred.

Table 12-9. Interrupt Sources Summary (BTM = 1)

SLCSV	I3	I2	I1	I0	Interrupt Source	Priority
0x28	1	0	1	0	Byte Framing Error	10
0x38	1	1	1	0	Reserved	14
0x3C	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.
- **TX Message Buffer Empty**
  In byte transfer mode, this interrupt source indicates that the byte in the SLCID has been transmitted.
- **RX Data Buffer Full — No Errors**
  This interrupt source indicates that a byte has been received and is waiting in SLCID. To clear this source, SLCID 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 specification. Bit errors are not checked when the LIN bus is running at high speed due to the effects of physical layer round trip delay. Bit errors are checked only when BEDD = 0.
- **Receiver Buffer Overrun Error**
  This error is an indication that the receive buffer has not been emptied and additional byte(s) have been received, resulting in lost data. Because this interrupt is higher priority than the receive buffer full interrupts, it will appear first when an overflow condition occurs. There will, however, be a pending receive interrupt which must also be cleared after the buffer overrun flag is cleared. Buffer overrun errors can be avoided if on reception of data (SLCSV=$14) SLCD0 is read or received data causes a framing error to occur.
- **Byte Framing Error**
  This error comes from the standard UART definition for byte encoding and occurs when STOP is sampled and reads back as a 0. STOP should always read as 1. A byte framing error could be encountered if the bit timing value programmed in BTH:L does not match the bit rate of the incoming data.
-
- **Wakeup**
  The wakeup interrupt source indicates that the SLIC module has entered SLIC run mode from SLIC wait mode.

For clarification, in this document, “command” messages will refer to any message frame where the SLIC module is receiving data bytes and “request” messages refer to message frames where the SLIC module will be expected to transmit data bytes. This is a generic description and should not be confused with the terminology in the LIN specification. The LIN use of the terms “command” and “request” have the same basic meaning, but are limited in scope to specific identifier values of 0x3C and 0x3D. In the SLIC module documentation, these terms have been used to describe these functional types of messages, regardless of the specific identifier value used.

12.6.7.2 Possible Errors on Message Headers

Possible errors on message headers are:

- Identifier-Parity-Error
- Byte Framing Error

12.6.8 Handling Command Message Frames

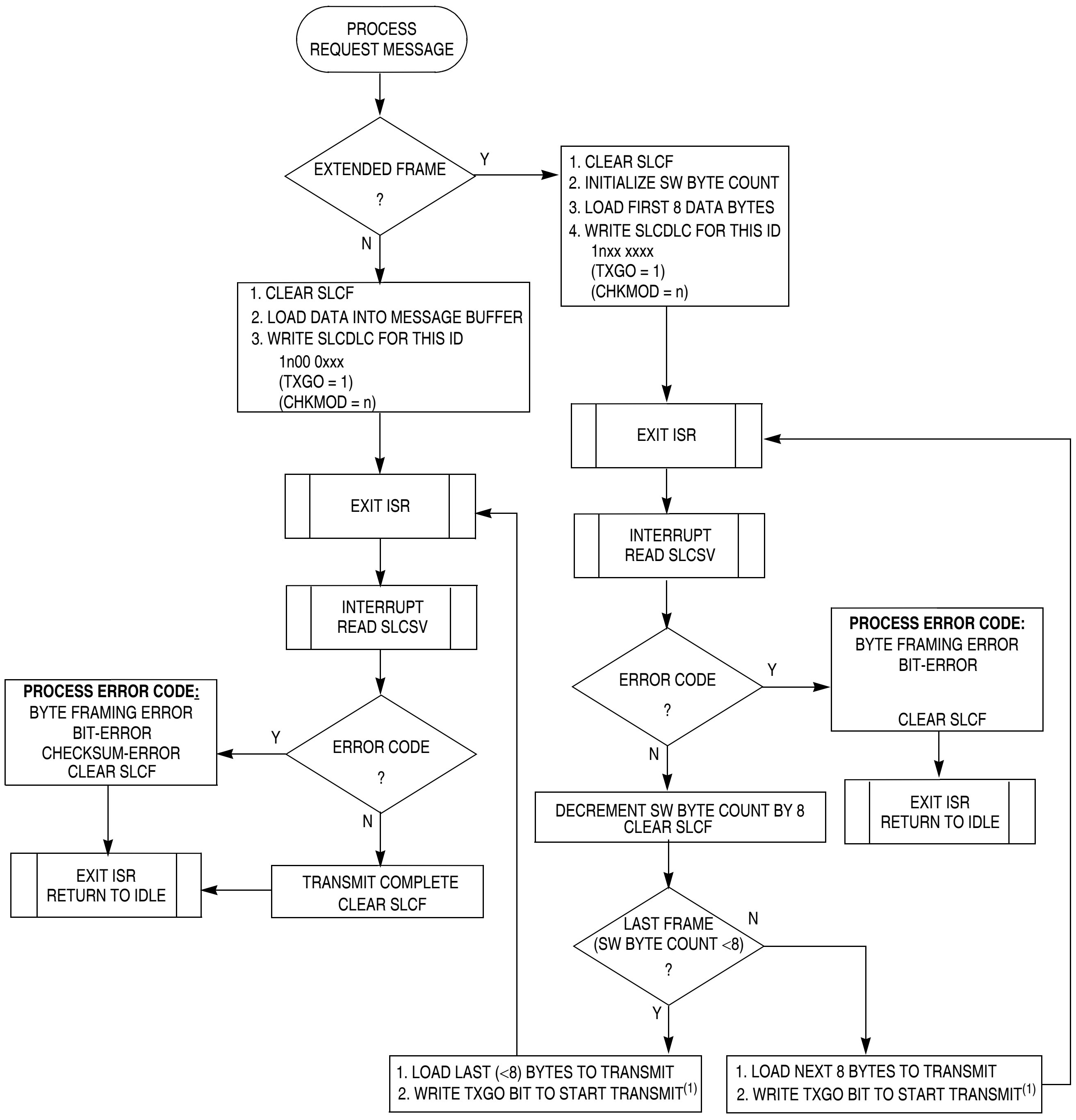
Figure 12-15 shows how to handle command message frames, where the SLIC module is receiving data from the master node.

Command message frames refer to LIN messages frames where the master node is “commanding” the slave node to do something. The implication is that the slave will then be receiving data from 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 0x3C identifier), or an extended command message frame utilizing the reserved 0x3E user defined identifier or perhaps the 0x3F LIN reserved extended identifier. The SLIC module is capable of handling message frames containing up to 64 bytes of data, while still automatically calculating and/or verifying the checksum.

12.6.8.1 Standard Command Message Frames

After the application software has read the incoming identifier and determined that it is a valid identifier which cannot be ignored using IMSG, it must determine if this message frame is a command message frame or a request message frame. (i.e., should the application receive data from the master or send data back to the master?)

The first case, shown in Figure 12-15 deals with command messages, where the SLIC will be receiving data from the master node. If the received identifier corresponds to a standard LIN command frame (i.e., 1–8 data bytes), the user must then write the number of bytes (determined by the system designer and directly linked with this particular identifier) corresponding to the length of the message frame into SLCDLC. The two most significant bits of this register are used for special control bits describing the nature of this message frame.



Note 1. When writing TXGO bit only, ensure that CHKMOD and data length values are not accidentally modified.

Figure 12-16. Handling Request LIN Message Frames

The next SLIC interrupt which occurs, if unmasked, will indicate the end of the request message frame and will either indicate that the frame was properly transmitted or that an error was encountered during transmission. Refer to Section 12.6.9.4, “Possible Errors on Request Message Data,” for more detailed explanation of these possible errors. This interrupt also signals to the application that the message frame is complete and all data bytes and the checksum value have been properly transmitted onto the bus.

Desired Bit Rate:	9,615 bps
External Crystal Frequency:	8.000 MHz

$$\frac{\text{1 Second}}{\text{9,615 Bits}} = \frac{104.004\ \mu\text{s}}{\text{1 Bit}}$$

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

$$\frac{104.004\ \mu\text{s}}{\text{1 Bit}} \times \frac{\text{1 SLIC Clock Period}}{\text{250 ns}} = \frac{\text{416.017 SLIC Clock Periods}}{\text{1 Bit}}$$

Therefore, the closest SLCBT value would be 416 SLIC clocks (SLCBT = 0x01A0).

Figure 12-20. SLCBT Value Calculation Example 4

This resolution affects the sampling accuracy of the SLIC module on receiving bytes, but only as far as locating the sample point of each bit within a given byte. The best sample point of the bit may be off by as much as one SLIC clock period from the exact center of the bit, if the proper SLCBT value for the desired bit rate is an odd number of SLIC clock periods.

Figure 12-21 shows an example of this error. In this example, the user has additionally chosen an incorrect value of 30 SLIC clocks for the length of one bit time, and a filter prescaler of 1. This makes little difference in the receive sampling of this particular bit, as the sample point is still within the bit and the digital filter will catch any noise pulses shorter than 16 filter clocks long.The ideal value of SLCBT would be 35 SLIC clocks, but the closest available value is 34, placing the sample point at 17 SLIC clocks into the bit.

The error in the bit time value chosen by the user in the above example will grow throughout the byte, as the sample point for the next bit will be only 30 SLIC clock cycles later (1 full bit time at this bit rate setting). The SLIC resynchronizes upon every falling edge received. In a 0x00 data byte, however, there are no falling edges after the beginning of the start bit. This means that the accumulated error of the sampling point over the data byte with these settings could be as high as 30 SLIC clock cycles (10 bits x {2 SLIC clocks due to user error + 1 SLIC clock resolution error}) placing it at the boundary between the last bit and the stop bit. This could result in missampling and missing a byte framing error on the last bit on high speed communications when the SLCBT count is relatively low. A properly chosen SLCBT value would result in a maximum error of 10 SLIC clock counts over a given byte. This is less than one filter delay time, and will not cause missampling of any of the bits in that byte. At the falling edge of the next start bit, the SLIC will resynchronize and any accumulated sampling error returns to 0. The sampling error becomes even less significant at lower speeds, when higher values of SLCBT are used to define a bit time, as the worst case bit time resolution error is still only one SLIC clock per bit (or maximum of 10 SLIC clocks per byte).

12.6.18 Digital Receive Filter

The receiver section of the SLIC module includes a digital low-pass filter to remove narrow noise pulses from the incoming message. A block diagram of the digital filter is shown in Figure 12-22.

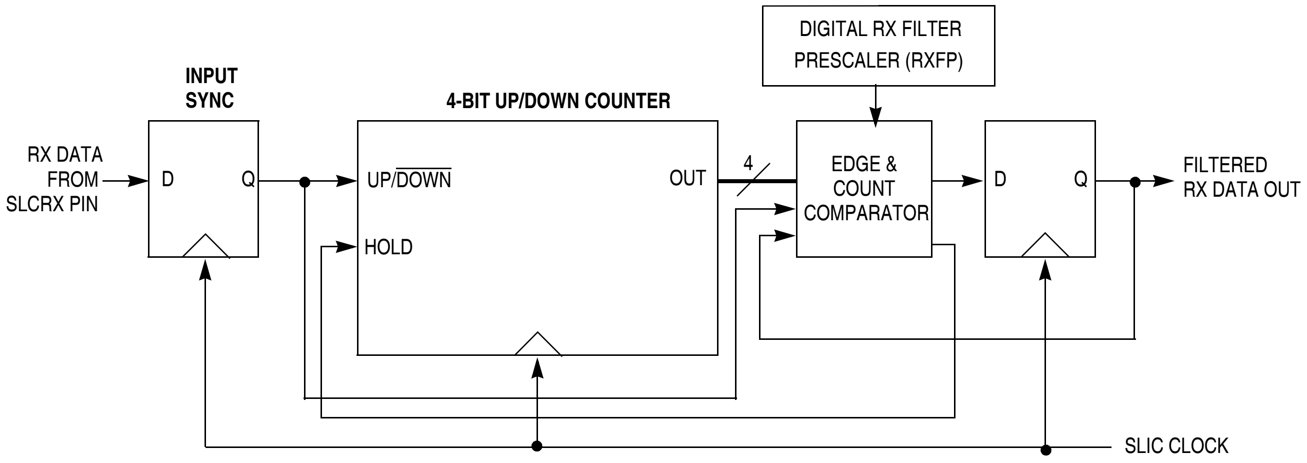


Figure 12-22. SLIC Module Rx Digital Filter Block Diagram

12.6.18.1 Digital Filter Operation

The clock for the digital filter is provided by the SLIC Interface clock. At each positive edge of the clock signal, the current state of the receiver input signal from the SLCRX pad is sampled. The SLCRX signal state is used to determine whether the counter should increment or decrement at the next positive edge of the clock signal.

The counter will increment if the input data sample is high but decrement if the input sample is low. The counter will thus progress up towards the highest count value (determined by RXFP bit settings), on average, the SLCRX signal remains high or progress down towards ‘0’ if, on average, the SLCRX signal remains low. The final counter value which determines when the filter will change state is generated by shifting the RXFP value right three positions and bitwise OR-ing the result with the value 0x0F. For example, a prescale setting of divide by 3 would give a count value of 0x2F.

When the counter eventually reaches this value, the digital filter decides that the condition of the SLCRX signal is at a stable logic level 1 and the data latch is set, causing the filtered Rx data signal to become a logic level 1. Furthermore, the counter is prevented from overflowing and can only be decremented from this state.

Alternatively, when the counter eventually reaches the value ‘0’, the digital filter decides that the condition of the SLCRX signal is at a stable logic level 0 and the data latch is reset, causing the filtered Rx data signal to become a logic level 0. Furthermore, the counter is prevented from underflowing and can only be incremented from this state.

The data latch will retain its value until the counter next reaches the opposite end point, signifying a definite transition of the SLCRX signal.

Chapter 15
Real-Time Counter (S08RTCV1)

15.1 Introduction

The RTC module consists of one 8-bit counter, one 8-bit comparator, several binary-based and decimal-based prescaler dividers, two clock sources, and one programmable periodic interrupt. This module can be used for time-of-day, calendar or any task scheduling functions. It can also serve as a cyclic wake up from low power modes without the need of external components.

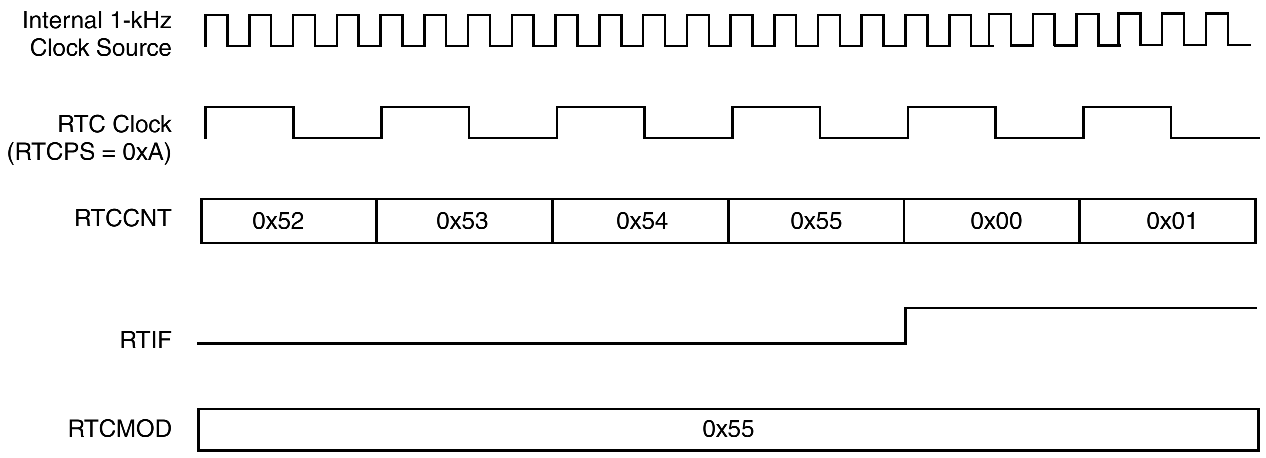


Figure 15-6. RTC Counter Overflow Example

In the example of Figure 15-6, the selected clock source is the 1-kHz internal oscillator clock source. The prescaler (RTCPS) is set to 0xA or divide-by-4. The modulo value in the RTCMOD register is set to 0x55. When the counter, RTCCNT, reaches the modulo value of 0x55, the counter overflows to 0x00 and continues counting. The real-time interrupt flag, RTIF, sets when the counter value changes from 0x55 to 0x00. A real-time interrupt is generated when RTIF is set, if RTIE is set.

15.5 Initialization/Application Information

This section provides example code to give some basic direction to a user on how to initialize and configure the RTC module. The example software is implemented in C language.

The example below shows how to implement time of day with the RTC using the 1-kHz clock source to achieve the lowest possible power consumption. Because the 1-kHz clock source is not as accurate as a crystal, software can be added for any adjustments. For accuracy without adjustments at the expense of additional power consumption, the external clock (ERCLK) or the internal clock (IRCLK) can be selected with appropriate prescaler and modulo values.

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;

/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;

/**********************************************************************
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
**********************************************************************/
```

16.2.1.1 EXTCLK — External Clock Source

Control bits in the timer status and control register allow the user to select nothing (timer disable), the bus-rate clock (the normal default source), a crystal-related clock, or an external clock as the clock which drives the TPM prescaler and subsequently the 16-bit TPM counter. The external clock source is synchronized in the TPM. The bus clock clocks the synchronizer; the frequency of the external source must be no more than one-fourth the frequency of the bus-rate clock, to meet Nyquist criteria and allowing for jitter.

The external clock signal shares the same pin as a channel I/O pin, so the channel pin will not be usable for channel I/O function when selected as the external clock source. It is the user's responsibility to avoid such settings. If this pin is used as an external clock source (CLKSB:CLKSA = 1:1), the channel can still be used in output compare mode as a software timer (ELSnB:ELSnA = 0:0).

16.2.1.2 TPMxCHn — TPM Channel n I/O Pin(s)

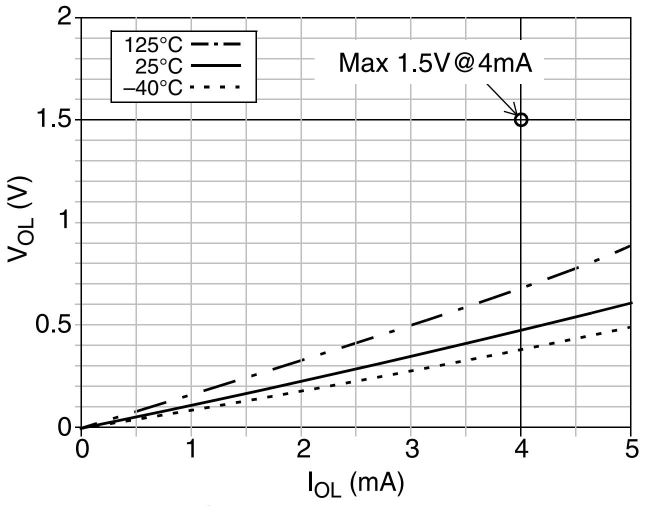
Each TPM channel is associated with an I/O pin on the MCU. The function of this pin depends on the channel configuration. The TPM pins share with general purpose I/O pins, where each pin has a port data register bit, and a data direction control bit, and the port has optional passive pullups which may be enabled whenever a port pin is acting as an input.

The TPM channel does not control the I/O pin when (ELSnB:ELSnA = 0:0) or when (CLKSB:CLKSA = 0:0) so it normally reverts to general purpose I/O control. When CPWMS = 1 (and ELSnB:ELSnA not = 0:0), all channels within the TPM are configured for center-aligned PWM and the TPMxCHn pins are all controlled by the TPM system. When CPWMS=0, the MSnB:MSnA control bits determine whether the channel is configured for input capture, output compare, or edge-aligned PWM.

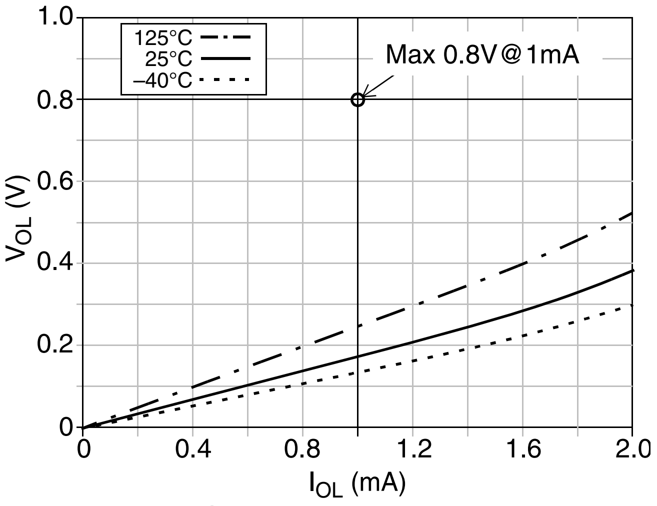
When a channel is configured for input capture (CPWMS=0, MSnB:MSnA = 0:0 and ELSnB:ELSnA not = 0:0), the TPMxCHn pin is forced to act as an edge-sensitive input to the TPM. ELSnB:ELSnA control bits determine what polarity edge or edges will trigger input-capture events. A synchronizer based on the bus clock is used to synchronize input edges to the bus clock. This implies the minimum pulse width—that can be reliably detected—on an input capture pin is four bus clock periods (with ideal clock pulses as near as two bus clocks can be detected). TPM uses this pin as an input capture input to override the port data and data direction controls for the same pin.

When a channel is configured for output compare (CPWMS=0, MSnB:MSnA = 0:1 and ELSnB:ELSnA not = 0:0), the associated data direction control is overridden, the TPMxCHn pin is considered an output controlled by the TPM, and the ELSnB:ELSnA control bits determine how the pin is controlled. The remaining three combinations of ELSnB:ELSnA determine whether the TPMxCHn pin is toggled, cleared, or set each time the 16-bit channel value register matches the timer counter.

When the output compare toggle mode is initially selected, the previous value on the pin is driven out until the next output compare event—then the pin is toggled.

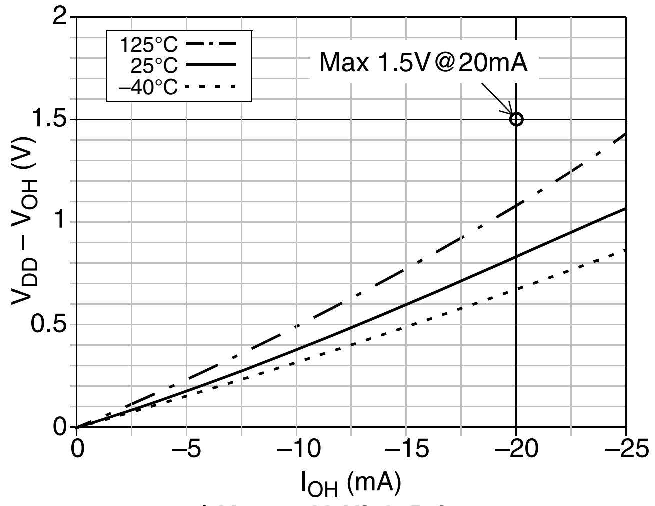


a) V_{DD} = 5V, Low Drive

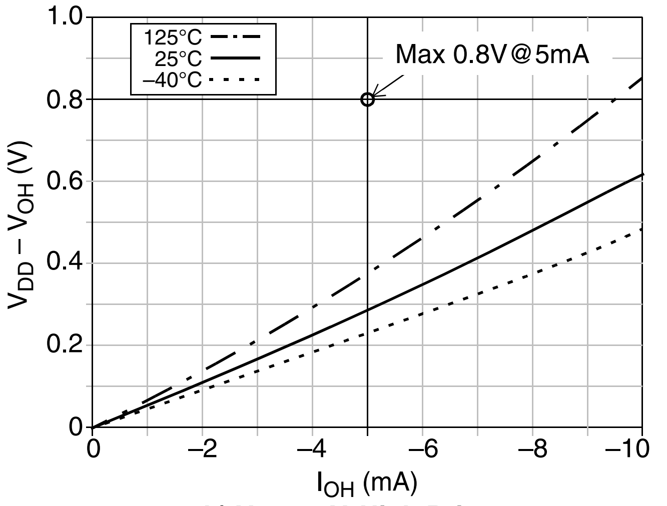


b) V_{DD} = 3V, Low Drive

Figure A-2. Typical V_{OL} vs I_{OL}, Low Drive Strength



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



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

Figure A-3. Typical $V_{DD} - V_{OH}$ vs I_{OH}, High Drive Strength