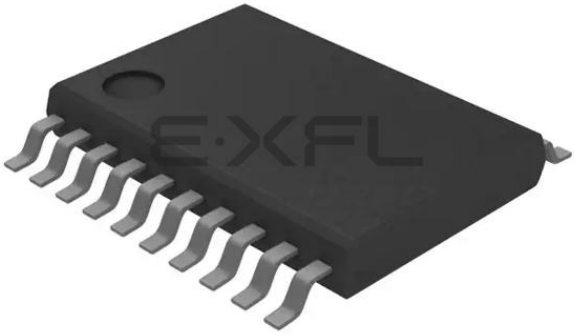


Welcome to E-XFL.COM

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	17
Program Memory Size	4KB (4K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	256 x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	20-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	20-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/mc9s08sh4ctj

MC9S08SH8 Features

8-Bit HCS08 Central Processor Unit (CPU)

- 40-MHz HCS08 CPU (central processor unit)
- HC08 instruction set with added BGND instruction
- Support for up to 32 interrupt/reset sources

On-Chip Memory

- FLASH read/program/erase over full operating voltage and temperature
- Random-access memory (RAM)

Power-Saving Modes

- Two very low power stop modes
- Reduced power wait mode
- Very low power real time interrupt for use in run, wait, and stop

Clock Source Options

- Oscillator (XOSC) — Loop-control Pierce oscillator; Crystal or ceramic resonator range of 31.25 kHz to 38.4 kHz or 1 MHz to 16 MHz
- Internal Clock Source (ICS) — Internal clock source module containing a frequency-locked loop (FLL) controlled by internal or external reference; precision trimming of internal reference allows 0.2% resolution and 2% deviation over temperature and voltage; supports bus frequencies from 2 MHz to 20 MHz.

System Protection

- Watchdog computer operating properly (COP) reset with opti/n to run from dedicated 1-kHz internal clock source or bus clock
- Low-voltage detection with reset or interrupt; selectable trip points
- Illegal opcode detection with reset
- Illegal address detection with reset
- FLASH block protect

Development Support

- Single-wire background debug interface
- Breakpoint capability to allow single breakpoint setting during in-circuit debugging (pluss two more breakpoints in on-chip debug module)
- On-chip, in-circuit emulation (ICE) debug module containing two comparators and nine trigger modes. Eight deep FIFO for storing change-of-flow address and event-only data. Debug module supports both tag and force breakpoints.

Peripherals

- **ADC** — 12-channel, 10-bit resolution, 2.5 µs conversion time, automatic compare function, temperature sensor, internal bandgap reference channel; runs in stop3
- **ACMP** — Analog comparator with selectable interrupt on rising, falling, or either edge of comparator output; compare option to fixed internal bandgap reference voltage; output can be optionally routed to TPM module; runs in stop3
- **SCI** — Full duplex non-return to zero (NRZ); LIN master extended break generation; LIN slave extended break detection; wake up on active edge
- **SPI** — Full-duplex or single-wire bidirectional; Double-buffered transmit and receive; Master or Slave mode; MSB-first or LSB-first shifting
- **IIC** — Up to 100 kbps with maximum bus loading; Multi-master operation; Programmable slave address; Interrupt driven byte-by-byte data transfer; supports broadcast mode and 10-bit addressing
- **MTIM** — 8-bit modulo counter with 8-bit prescaler and overflow interrupt
- **TPMx** — Two 2-channel timer pwm modules (TPM1, TPM2); Selectable input capture, output compare, or buffered edge- or center-aligned PWM on each channel
- **RTC** — (Real-time counter) 8-bit modulus counter with binary or decimal based prescaler; External clock source for precise time base, time-of-day, calendar or task scheduling functions; Free running on-chip low power oscillator (1 kHz) for cyclic wake-up without external components, runs in all MCU modes

Input/Output

- 17 general purpose I/O pins (GPIOs) and 1 output-only pin
- 8 interrupt pins with selectable polarity
- Ganged output option for PTB[5:2] and PTC[3:0]; allows single write to change state of multiple pins
- Hysteresis and configurable pull up device on all input pins; Configurable slew rate and drive strength on all output pins.

Package Options

- 24-QFN, 20-TSSOP, 20-SOIC, 20-PDIP, 16-TSSOP, 8-SOIC

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

Address	Register Name	Bit 7	6	5	4	3	2	1	Bit 0
0x1845	**PTAPS**	0	0	0	PTAPS4	PTAPS3	PTAPS2	PTAPS1	PTAPS0
0x1846	**PTAES**	0	0	0	PTAES4	PTAES3	PTAES2	PTAES1	PTAES0
0x1847	Reserved	—	—	—	—	—	—	—	—
0x1848	**PTBPE**	PTBPE7	PTBPE6	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1849	**PTBSE**	PTBSE7	PTBSE6	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x184A	**PTBDS**	PTBDS7	PTBDS6	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x184B	Reserved	—	—	—	—	—	—	—	—
0x184C	**PTBSC**	0	0	0	0	PTBIF	PTBACK	PTBIE	PTBMOD
0x184D	**PTBPS**	0	0	0	0	PTBPS3	PTBPS2	PTBPS1	PTBPS0
0x184E	**PTBES**	0	0	0	0	PTBES3	PTBES2	PTBES1	PTBES0
0x184F	Reserved	—	—	—	—	—	—	—	—
0x1850	**PTCPE**	0	0	0	0	PTCPE3	PTCPE2	PTCPE1	PTCPE0
0x1851	**PTCSE**	0	0	0	0	PTCSE3	PTCSE2	PTCSE1	PTCSE0
0x1852	**PTCDS**	0	0	0	0	PTCDS3	PTCDS2	PTCDS1	PTCDS0
0x1853	**GNGC**	GNGPS7	GNGPS6	GNGPS5	GNGPS4	GNGPS3	GNGPS2	GNGPS1	GNGEN
0x1854– 0x185F	Reserved	— —	— —	— —	— —	— —	— —	— —	— —

other than the most experienced programmers because it can lead to subtle program errors that are difficult to debug.

The interrupt service routine ends with a return-from-interrupt (RTI) instruction which restores the CCR, A, X, and PC registers to their pre-interrupt values by reading the previously saved information from the stack.

NOTE

For compatibility with M68HC08 devices, the H register is not automatically saved and restored. It is good programming practice to push H onto the stack at the start of the interrupt service routine (ISR) and restore it immediately before the RTI that is used to return from the ISR.

If more than one interrupt is pending when the I bit is cleared, the highest priority source is serviced first (see Table 5-2).

5.5.1 Interrupt Stack Frame

Figure 5-1 shows the contents and organization of a stack frame. Before the interrupt, the stack pointer (SP) points at the next available byte location on the stack. The current values of CPU registers are stored on the stack starting with the low-order byte of the program counter (PCL) and ending with the CCR. After stacking, the SP points at the next available location on the stack which is the address that is one less than the address where the CCR was saved. The PC value that is stacked is the address of the instruction in the main program that would have executed next if the interrupt had not occurred.

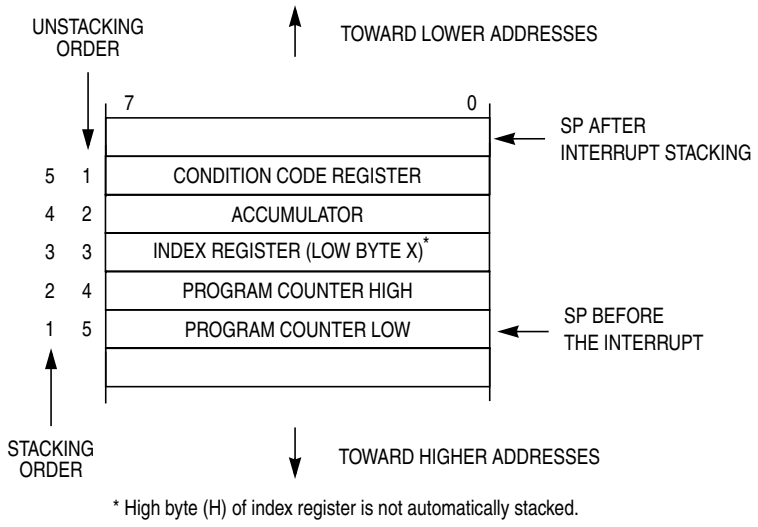


Figure 5-1. Interrupt Stack Frame

When an RTI instruction is executed, these values are recovered from the stack in reverse order. As part of the RTI sequence, the CPU fills the instruction pipeline by reading three bytes of program information, starting from the PC address recovered from the stack.

Chapter 6
Parallel Input/Output Control

This section explains software controls related to parallel input/output (I/O) and pin control. The MC9S08SH8 has three parallel I/O ports which include a total of 17 I/O pins and one output-only pin. 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 pin 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 disabled (PTxSEn = 0), 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.

6.6.1.6 Port A Interrupt Status and Control Register (PTASC)

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTAIF	0	PTAIE	PTAMOD
W						PTAACK		
Reset:	0	0	0	0	0	0	0	0

Figure 6-8. Port A Interrupt Status and Control Register (PTASC)

Table 6-7. PTASC Register Field Descriptions

Field	Description
3 PTAIF	**Port A Interrupt Flag** — PTAIF indicates when a port A interrupt is detected. Writes have no effect on PTAIF. 0 No port A interrupt detected. 1 Port A interrupt detected.
2 PTAACK	**Port A Interrupt Acknowledge** — Writing a 1 to PTAACK is part of the flag clearing mechanism. PTAACK always reads as 0.
1 PTAIE	**Port A Interrupt Enable** — PTAIE determines whether a port A interrupt is requested. 0 Port A interrupt request not enabled. 1 Port A interrupt request enabled.
0 PTAMOD	**Port A Detection Mode** — PTAMOD (along with the PTAES bits) controls the detection mode of the port A interrupt pins. 0 Port A pins detect edges only. 1 Port A pins detect both edges and levels.

6.6.1.7 Port A Interrupt Pin Select Register (PTAPS)

	7	6	5	4	3	2	1	0
R	0	0	0	0	PTAPS3	PTAPS2	PTAPS1	PTAPS0
W								
Reset:	0	0	0	0	0	0	0	0

Figure 6-9. Port A Interrupt Pin Select Register (PTAPS)

Table 6-8. PTAPS Register Field Descriptions

Field	Description
3:0 PTAPS[3:0]	**Port A Interrupt Pin Selects** — Each of the PTAPSn bits enable the corresponding port A interrupt pin. 0 Pin not enabled as interrupt. 1 Pin enabled as interrupt.

6.6.2 Port B Registers

Port B is controlled by the registers listed below.

6.6.2.1 Port B Data Register (PTBD)

	7	6	5	4	3	2	1	0
R W	PTBD7	PTBD6	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
Reset:	0	0	0	0	0	0	0	0

Figure 6-11. Port B Data Register (PTBD)

Table 6-10. PTBD Register Field Descriptions

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

6.6.2.2 Port B Data Direction Register (PTBDD)

	7	6	5	4	3	2	1	0
R W	PTBDD7	PTBDD6	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
Reset:	0	0	0	0	0	0	0	0

Figure 6-12. Port B Data Direction Register (PTBDD)

Table 6-11. PTBDD Register Field Descriptions

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

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

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 *oprx8*a COMA COMX COM *oprx8*,X COM ,X COM *oprx8*,SP	Complement (One's Complement) M ← (M̄)= $FF – (M) A ← (Ā) = $FF – (A) X ← (X̄) = $FF – (X) M ← (M̄) = $FF – (M) M ← (M̄) = $FF – (M) M ← (M̄) = $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 –	– ↕ ↕ –
DIV	Divide A ← (H:A)÷(X); H ← Remainder	INH	52	6	ffffp	– 1 1 –	– – ↕ ↕
EOR #*opr8i* EOR *opr8a* EOR *opr16a* EOR *oprx16*,X EOR *oprx8*,X EOR ,X EOR *oprx16*,SP EOR *oprx8*,SP	Exclusive OR Memory with Accumulator A ← (A ⊕ M)	IMM DIR EXT IX2 IX1 IX SP2 SP1	A8 ii B8 dd C8 hh ll D8 ee ff E8 ff F8 9E D8 ee ff 9E E8 ff	2 3 4 4 3 3 5 4	pp rpp prpp prpp rpp rfp pprpp prpp	0 1 1 –	– ↕ ↕ –

Table 7-2. Instruction Set Summary (Sheet 9 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
TXS	Transfer Index Reg. to SP SP ← (H:X) – $0001	INH	94	2	fp	– 1 1 –	– – – –
WAIT	Enable Interrupts; Wait for Interrupt I bit ← 0; Halt CPU	INH	8F	2+	fp...	– 1 1 –	0 – – –

Source Form: Everything in the source forms columns, *except expressions in italic characters*, is literal information which must appear in the assembly source file exactly as shown. The initial 3- to 5-letter mnemonic and the characters (#, ( ) and +) are always a literal characters.

n	Any label or expression that evaluates to a single integer in the range 0-7.
opr8i	Any label or expression that evaluates to an 8-bit immediate value.
opr16i	Any label or expression that evaluates to a 16-bit immediate value.
opr8a	Any label or expression that evaluates to an 8-bit direct-page address ($00xx).
opr16a	Any label or expression that evaluates to a 16-bit address.
oprx8	Any label or expression that evaluates to an unsigned 8-bit value, used for indexed addressing.
oprx16	Any label or expression that evaluates to a 16-bit value, used for indexed addressing.
rel	Any label or expression that refers to an address that is within –128 to +127 locations from the start of the next instruction.

Operation Symbols:

A	Accumulator
CCR	Condition code register
H	Index register high byte
M	Memory location
n	Any bit
opr	Operand (one or two bytes)
PC	Program counter
PCH	Program counter high byte
PCL	Program counter low byte
rel	Relative program counter offset byte
SP	Stack pointer
SPL	Stack pointer low byte
X	Index register low byte
&	Logical AND
|	Logical OR
⊕	Logical EXCLUSIVE OR
( )	Contents of
+	Add
–	Subtract, Negation (two's complement)
×	Multiply
÷	Divide
#	Immediate value
←	Loaded with
:	Concatenated with

Addressing Modes:

DIR	Direct addressing mode
EXT	Extended addressing mode
IMM	Immediate addressing mode
INH	Inherent addressing mode
IX	Indexed, no offset addressing mode
IX1	Indexed, 8-bit offset addressing mode
IX2	Indexed, 16-bit offset addressing mode
IX+	Indexed, no offset, post increment addressing mode
IX1+	Indexed, 8-bit offset, post increment addressing mode
REL	Relative addressing mode
SP1	Stack pointer, 8-bit offset addressing mode
SP2	Stack pointer 16-bit offset addressing mode

Cycle-by-Cycle Codes:

f	Free cycle. This indicates a cycle where the CPU does not require use of the system buses. An f cycle is always one cycle of the system bus clock and is always a read cycle.
p	Program fetch; read from next consecutive location in program memory
r	Read 8-bit operand
s	Push (write) one byte onto stack
u	Pop (read) one byte from stack
v	Read vector from $FFxx (high byte first)
w	Write 8-bit operand

CCR Bits:

V	Overflow bit
H	Half-carry bit
I	Interrupt mask
N	Negative bit
Z	Zero bit
C	Carry/borrow bit

CCR Effects:

↕	Set or cleared
–	Not affected
U	Undefined

Figure 9-4. Input Channel Select (continued)

ADCH	Input Select
01000	AD8
01001	AD9
01010	AD10
01011	AD11
01100	AD12
01101	AD13
01110	AD14
01111	AD15

ADCH	Input Select
11000	AD24
11001	AD25
11010	AD26
11011	AD27
11100	Reserved
11101	V_{REFH}
11110	V_{REFL}
11111	Module disabled

9.3.2 Status and Control Register 2 (ADCSC2)

The ADCSC2 register is used to control the compare function, conversion trigger and conversion active of the ADC module.

	7	6	5	4	3	2	1	0
R	ADACT	ADTRG	ACFE	ACFGT	0	0	R[1]	R[1]
W								
Reset:	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

[1] Bits 1 and 0 are reserved bits that must always be written to 0.

Figure 9-5. Status and Control Register 2 (ADCSC2)

Table 9-4. ADCSC2 Register Field Descriptions

Field	Description
7 ADACT	**Conversion Active** — ADACT indicates that a conversion is in progress. ADACT is set when a conversion is initiated and cleared when a conversion is completed or aborted. 0 Conversion not in progress 1 Conversion in progress
6 ADTRG	**Conversion Trigger Select** — ADTRG is used to select the type of trigger to be used for initiating a conversion. Two types of trigger are selectable: software trigger and hardware trigger. When software trigger is selected, a conversion is initiated following a write to ADCSC1. When hardware trigger is selected, a conversion is initiated following the assertion of the ADHWT input. 0 Software trigger selected 1 Hardware trigger selected

9.4.5 Automatic Compare Function

The compare function can be configured to check for either an upper limit or lower limit. After the input is sampled and converted, the result is added to the two's complement of the compare value (ADCCVH and ADCCVL). When comparing to an upper limit (ACFGT = 1), if the result is greater-than or equal-to the compare value, COCO is set. When comparing to a lower limit (ACFGT = 0), if the result is less than the compare value, COCO is set. The value generated by the addition of the conversion result and the two's complement of the compare value is transferred to ADCRH and ADCRL.

Upon completion of a conversion while the compare function is enabled, if the compare condition is not true, COCO is not set and no data is transferred to the result registers. An ADC interrupt is generated upon the setting of COCO if the ADC interrupt is enabled (AIEN = 1).

NOTE

The compare function can be used to monitor the voltage on a channel while the MCU is in either wait or stop3 mode. The ADC interrupt will wake the MCU when the compare condition is met.

9.4.6 MCU Wait Mode Operation

The WAIT instruction puts the MCU in a lower power-consumption standby mode from which recovery is very fast because the clock sources remain active. If a conversion is in progress when the MCU enters wait mode, it continues until completion. Conversions can be initiated while the MCU is in wait mode by means of the hardware trigger or if continuous conversions are enabled.

The bus clock, bus clock divided by two, and ADACK are available as conversion clock sources while in wait mode. The use of ALTCLK as the conversion clock source in wait is dependent on the definition of ALTCLK for this MCU. Consult the module introduction for information on ALTCLK specific to this MCU.

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

9.4.7 MCU Stop3 Mode Operation

The STOP instruction is used to put the MCU in a low power-consumption standby mode during which most or all clock sources on the MCU are disabled.

9.4.7.1 Stop3 Mode With ADACK Disabled

If the asynchronous clock, ADACK, is not selected as the conversion clock, executing a STOP instruction aborts the current conversion and places the ADC in its idle state. The contents of ADCRH and ADCRL are unaffected by stop3 mode.After exiting from stop3 mode, a software or hardware trigger is required to resume conversions.

9.6.2 Sources of Error

Several sources of error exist for A/D conversions. These are discussed in the following sections.

9.6.2.1 Sampling Error

For proper conversions, the input must be sampled long enough to achieve the proper accuracy. Given the maximum input resistance of approximately 7kΩ and input capacitance of approximately 5.5 pF, sampling to within 1/4LSB (at 10-bit resolution) can be achieved within the minimum sample window (3.5 cycles @ 8 MHz maximum ADCK frequency) provided the resistance of the external analog source (R_{AS}) is kept below 5 kΩ.

Higher source resistances or higher-accuracy sampling is possible by setting ADLSMP (to increase the sample window to 23.5 cycles) or decreasing ADCK frequency to increase sample time.

9.6.2.2 Pin Leakage Error

Leakage on the I/O pins can cause conversion error if the external analog source resistance (R_{AS}) is high. If this error cannot be tolerated by the application, keep R_{AS} lower than $V_{DDAD} / (2^N * I_{LEAK})$ for less than 1/4LSB leakage error (N = 8 in 8-bit mode or 10 in 10-bit mode).

9.6.2.3 Noise-Induced Errors

System noise which occurs during the sample or conversion process can affect the accuracy of the conversion. The ADC accuracy numbers are guaranteed as specified only if the following conditions are met:

- There is a 0.1 µF low-ESR capacitor from V_{REFH} to V_{REFL}.
- There is a 0.1 µF low-ESR capacitor from V_{DDAD} to V_{SSAD}.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from V_{DDAD} to V_{SSAD}.
- V_{SSAD} (and V_{REFL}, if connected) is connected to V_{SS} at a quiet point in the ground plane.
- Operate the MCU in wait or stop3 mode before initiating (hardware triggered conversions) or immediately after initiating (hardware or software triggered conversions) the ADC conversion.
  - For software triggered conversions, immediately follow the write to the ADCSC1 with a WAIT instruction or STOP instruction.
  - For stop3 mode operation, select ADACK as the clock source. Operation in stop3 reduces V_{DD} noise but increases effective conversion time due to stop recovery.
- There is no I/O switching, input or output, on the MCU during the conversion.

There are some situations where external system activity causes radiated or conducted noise emissions or excessive V_{DD} noise is coupled into the ADC. In these situations, or when the MCU cannot be placed in wait or stop3 or I/O activity cannot be halted, these recommended actions may reduce the effect of noise on the accuracy:

- Place a 0.01 µF capacitor (C_{AS}) on the selected input channel to V_{REFL} or V_{SSAD} (this will improve noise issues but will affect sample rate based on the external analog source resistance).

10.3.1 ICS Control Register 1 (ICSC1)

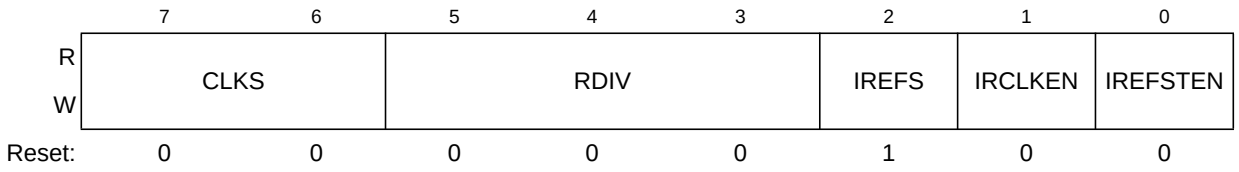


Figure 10-3. ICS Control Register 1 (ICSC1)

Table 10-2. ICS Control Register 1 Field Descriptions

Field	Description
7:6 CLKS	**Clock Source Select** — Selects the clock source that controls the bus frequency. The actual bus frequency depends on the value of the BDIV bits. 00 Output of FLL is selected. 01 Internal reference clock is selected. 10 External reference clock is selected. 11 Reserved, defaults to 00.
5:3 RDIV	**Reference Divider** — Selects the amount to divide down the FLL reference clock selected by the IREFS bits. Resulting frequency must be in the range 31.25 kHz to 39.0625 kHz. 000 Encoding 0 — Divides reference clock by 1 (reset default) 001 Encoding 1 — Divides reference clock by 2 010 Encoding 2 — Divides reference clock by 4 011 Encoding 3 — Divides reference clock by 8 100 Encoding 4 — Divides reference clock by 16 101 Encoding 5 — Divides reference clock by 32 110 Encoding 6 — Divides reference clock by 64 111 Encoding 7 — Divides reference clock by 128
2 IREFS	**Internal Reference Select** — The IREFS bit selects the reference clock source for the FLL. 1 Internal reference clock selected 0 External reference clock selected
1 IRCLKEN	**Internal Reference Clock Enable** — The IRCLKEN bit enables the internal reference clock for use as ICSIRCLK. 1 ICSIRCLK active 0 ICSIRCLK inactive
0 IREFSTEN	**Internal Reference Stop Enable** — The IREFSTEN bit controls whether or not the internal reference clock remains enabled when the ICS enters stop mode. 1 Internal reference clock stays enabled in stop if IRCLKEN is set or if ICS is in FEI, FBI, or FBILP mode before entering stop 0 Internal reference clock is disabled in stop

the transition from master to slave mode does not generate a stop condition. Meanwhile, a status bit is set by hardware to indicate loss of arbitration.

11.4.1.7 Clock Synchronization

Because wire-AND logic is performed on the SCL line, a high-to-low transition on the SCL line affects all the devices connected on the bus. The devices start counting their low period and after a device's clock has gone low, it holds the SCL line low until the clock high state is reached. However, the change of low to high in this device clock may not change the state of the SCL line if another device clock is still within its low period. Therefore, synchronized clock SCL is held low by the device with the longest low period. Devices with shorter low periods enter a high wait state during this time (see Figure 11-10). When all devices concerned have counted off their low period, the synchronized clock SCL line is released and pulled high. There is then no difference between the device clocks and the state of the SCL line and all the devices start counting their high periods. The first device to complete its high period pulls the SCL line low again.

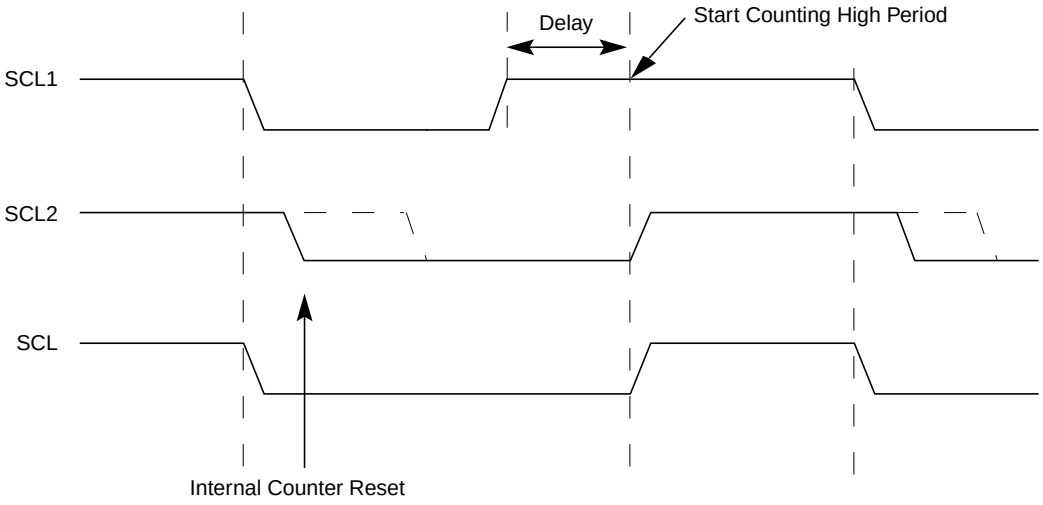


Figure 11-10. IIC Clock Synchronization

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.

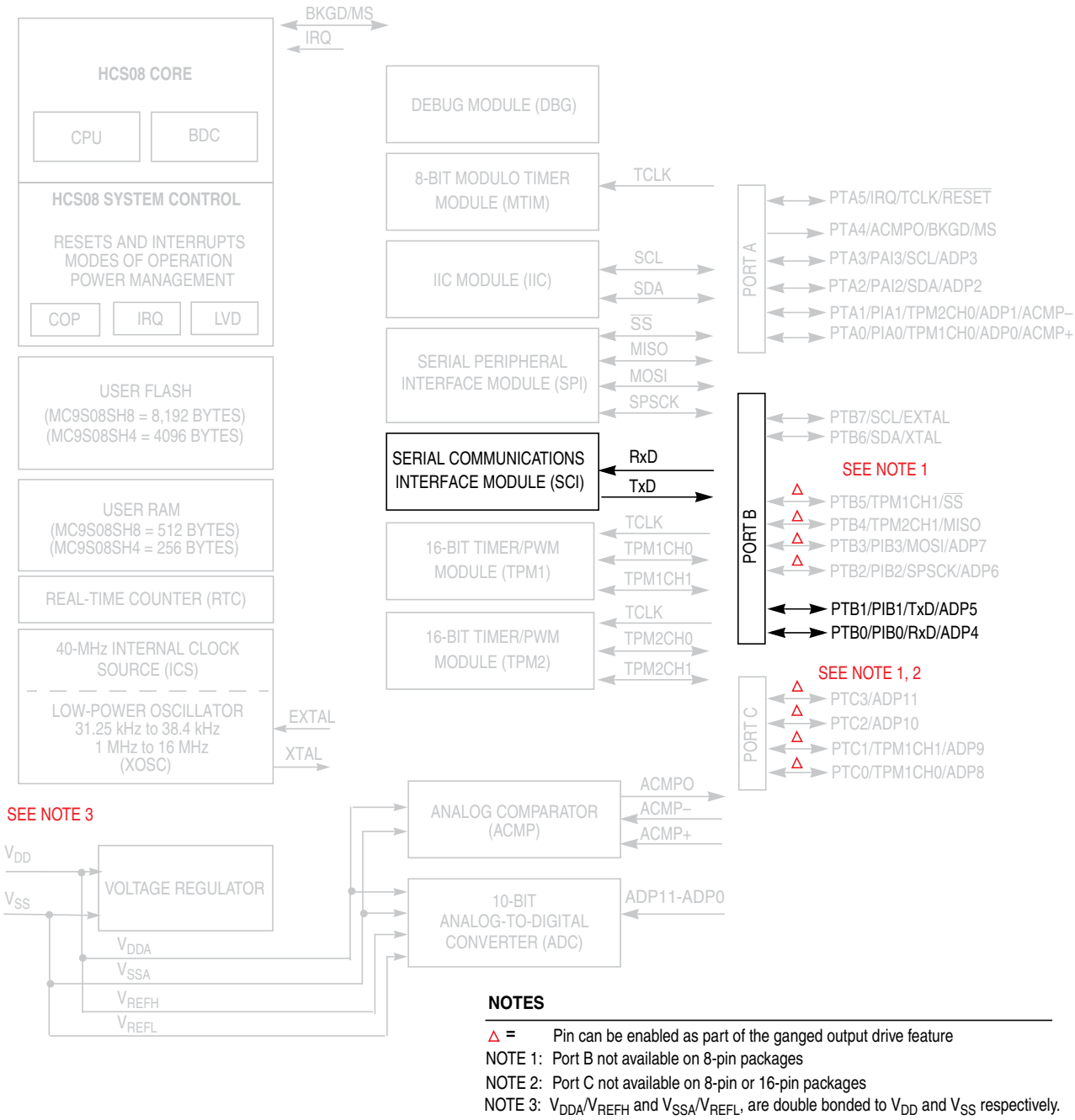


Figure 14-1. MC9S08SH8 Block Diagram Highlighting the SCI Module

The most common uses of the SPI system include connecting simple shift registers for adding input or output ports or connecting small peripheral devices such as serial A/D or D/A converters. Although Figure 15-2 shows a system where data is exchanged between two MCUs, many practical systems involve simpler connections where data is unidirectionally transferred from the master MCU to a slave or from a slave to the master MCU.

15.1.2.2 SPI Module Block Diagram

Figure 15-3 is a block diagram of the SPI module. The central element of the SPI is the SPI shift register. Data is written to the double-buffered transmitter (write to SPID) and gets transferred to the SPI shift register at the start of a data transfer. After shifting in a byte of data, the data is transferred into the double-buffered receiver where it can be read (read from SPID). Pin multiplexing logic controls connections between MCU pins and the SPI module.

When the SPI is configured as a master, the clock output is routed to the SPSCK pin, the shifter output is routed to MOSI, and the shifter input is routed from the MISO pin.

When the SPI is configured as a slave, the SPSCK pin is routed to the clock input of the SPI, the shifter output is routed to MISO, and the shifter input is routed from the MOSI pin.

In the external SPI system, simply connect all SPSCK pins to each other, all MISO pins together, and all MOSI pins together. Peripheral devices often use slightly different names for these pins.

Table 15-7. SPIS Register Field Descriptions

Field	Description
7 SPRF	**SPI Read Buffer Full Flag** — SPRF is set at the completion of an SPI transfer to indicate that received data may be read from the SPI data register (SPID). SPRF is cleared by reading SPRF while it is set, then reading the SPI data register. 0 No data available in the receive data buffer 1 Data available in the receive data buffer
5 SPTEF	**SPI Transmit Buffer Empty Flag** — This bit is set when there is room in the transmit data buffer. It is cleared by reading SPIS with SPTEF set, followed by writing a data value to the transmit buffer at SPID. SPIS must be read with SPTEF = 1 before writing data to SPID or the SPID write will be ignored. SPTEF generates an SPTEF CPU interrupt request if the SPTIE bit in the SPIC1 is also set. SPTEF is automatically set when a data byte transfers from the transmit buffer into the transmit shift register. For an idle SPI (no data in the transmit buffer or the shift register and no transfer in progress), data written to SPID is transferred to the shifter almost immediately so SPTEF is set within two bus cycles allowing a second 8-bit data value to be queued into the transmit buffer. After completion of the transfer of the value in the shift register, the queued value from the transmit buffer will automatically move to the shifter and SPTEF will be set to indicate there is room for new data in the transmit buffer. If no new data is waiting in the transmit buffer, SPTEF simply remains set and no data moves from the buffer to the shifter. 0 SPI transmit buffer not empty 1 SPI transmit buffer empty
4 MODF	**Master Mode Fault Flag** — MODF is set if the SPI is configured as a master and the slave select input goes low, indicating some other SPI device is also configured as a master. The $\overline{SS}$ pin acts as a mode fault error input only when MSTR = 1, MODFEN = 1, and SSOE = 0; otherwise, MODF will never be set. MODF is cleared by reading MODF while it is 1, then writing to SPI control register 1 (SPIC1). 0 No mode fault error 1 Mode fault error detected

15.4.5 SPI Data Register (SPID)

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

Figure 15-9. SPI Data Register (SPID)

Reads of this register return the data read from the receive data buffer. Writes to this register write data to the transmit data buffer. When the SPI is configured as a master, writing data to the transmit data buffer initiates an SPI transfer.

Data should not be written to the transmit data buffer unless the SPI transmit buffer empty flag (SPTEF) is set, indicating there is room in the transmit buffer to queue a new transmit byte.

Data may be read from SPID any time after SPRF is set and before another transfer is finished. Failure to read the data out of the receive data buffer before a new transfer ends causes a receive overrun condition and the data from the new transfer is lost.

pin from a master and the MISO waveform applies to the MISO output from a slave. The $\overline{SS}$ OUT waveform applies to the slave select output from a master (provided MODFEN and SSOE = 1). The master $\overline{SS}$ output goes to active low one-half SPSCK cycle before the start of the transfer and goes back high at the end of the eighth bit time of the transfer. The $\overline{SS}$ IN waveform applies to the slave select input of a slave.

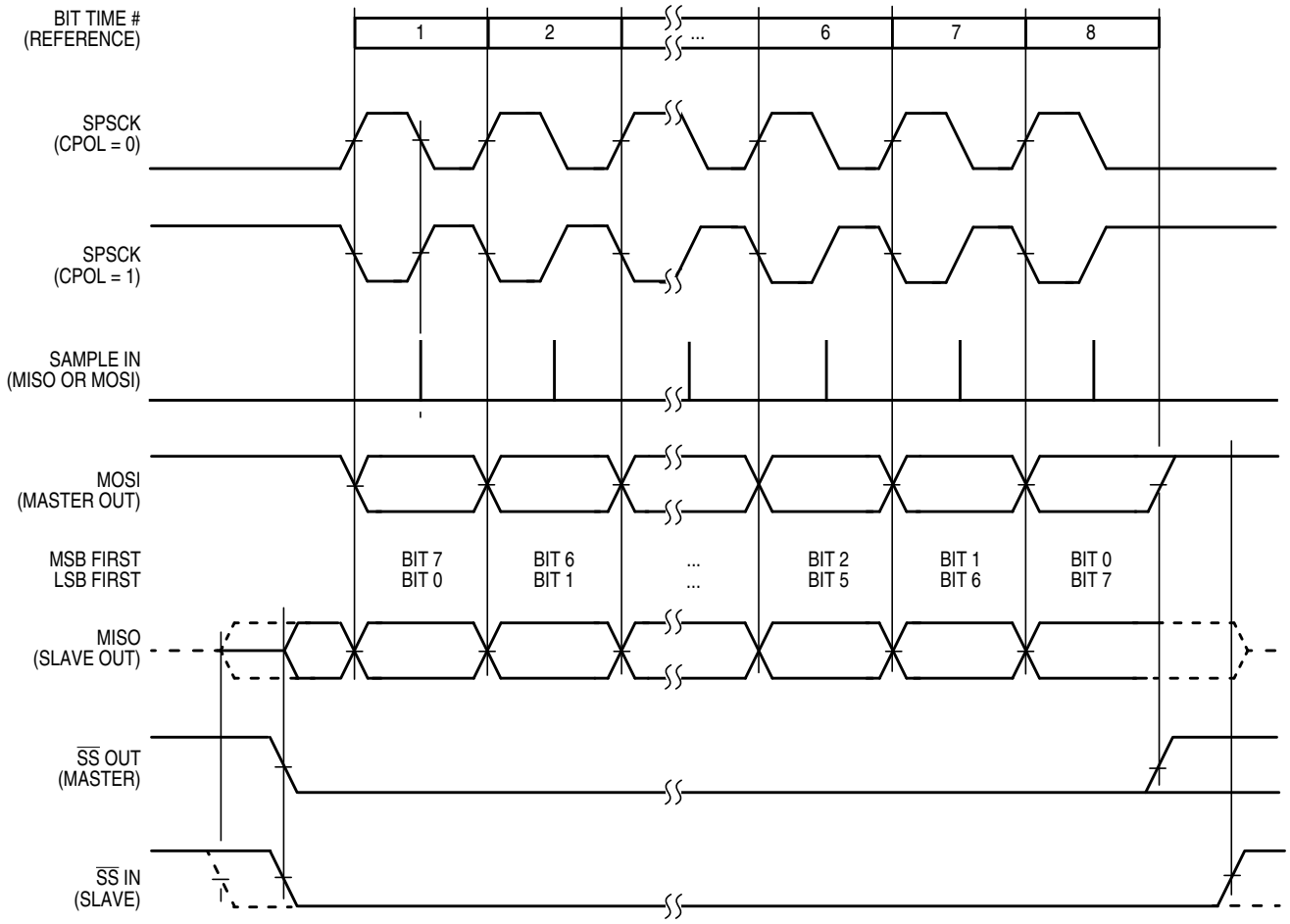


Figure 15-10. SPI Clock Formats (CPHA = 1)

When CPHA = 1, the slave begins to drive its MISO output when $\overline{SS}$ goes to active low, but the data is not defined until the first SPSCK edge. The first SPSCK edge shifts the first bit of data from the shifter onto the MOSI output of the master and the MISO output of the slave. The next SPSCK edge causes both the master and the slave to sample the data bit values on their MISO and MOSI inputs, respectively. At the third SPSCK edge, the SPI shifter shifts one bit position which shifts in the bit value that was just sampled, and shifts the second data bit value out the other end of the shifter to the MOSI and MISO outputs of the master and slave, respectively. When CHPA = 1, the slave's $\overline{SS}$ input is not required to go to its inactive high level between transfers.

Figure 15-11 shows the clock formats when CPHA = 0. At the top of the figure, the eight bit times are shown for reference with bit 1 starting as the slave is selected ($\overline{SS}$ IN goes low), and bit 8 ends at the last SPSCK edge. The MSB first and LSB first lines show the order of SPI data bits depending on the setting

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.

In output compare mode, values are transferred to the corresponding timer channel registers only after both 8-bit halves of a 16-bit register have been written and according to the value of CLKSB:CLKSA bits, so:

- If (CLKSB:CLKSA = 0:0), the registers are updated when the second byte is written
- If (CLKSB:CLKSA not = 0:0), the registers are updated at the next change of the TPM counter (end of the prescaler counting) after the second byte is written.

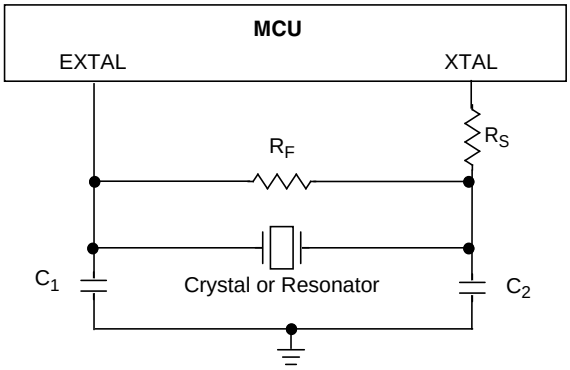
The coherency sequence can be manually reset by writing to the channel status/control register (TPMxCnSC).

An output compare event sets a flag bit (CHnF) which may optionally generate a CPU-interrupt request.

16.4.2.3 Edge-Aligned PWM Mode

This type of PWM output uses the normal up-counting mode of the timer counter (CPWMS=0) and can be used when other channels in the same TPM are configured for input capture or output compare functions. The period of this PWM signal is determined by the value of the modulus register (TPMxMODH:TPMxMODL) plus 1. The duty cycle is determined by the setting in the timer channel register (TPMxCnVH:TPMxCnVL). The polarity of this PWM signal is determined by the setting in the ELSnA control bit. 0% and 100% duty cycle cases are possible.

The output compare value in the TPM channel registers determines the pulse width (duty cycle) of the PWM signal (Figure 16-15). The time between the modulus overflow and the output compare is the pulse width. If ELSnA=0, the counter overflow forces the PWM signal high, and the output compare forces the PWM signal low. If ELSnA=1, the counter overflow forces the PWM signal low, and the output compare forces the PWM signal high.


MCU
EXTAL
XTAL
RS
RF
C1
Crystal or Resonator
C2