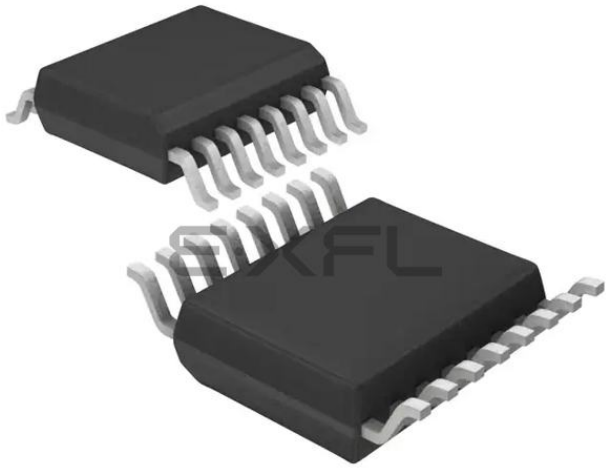


Welcome to E-XFL.COM

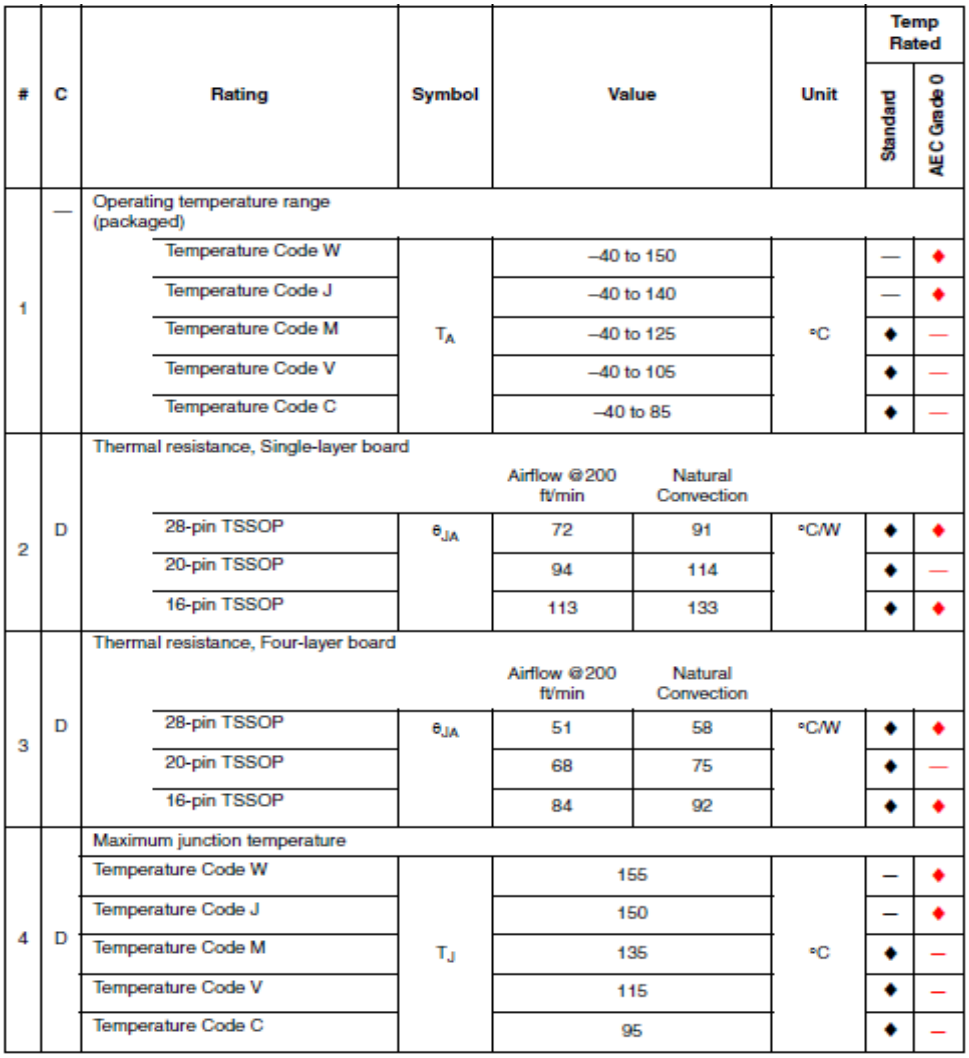
What is "Embedded - Microcontrollers"?

"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	12
Program Memory Size	32KB (32K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 150°C (TA)
Mounting Type	Surface Mount
Package / Case	16-TSSOP (0.173", 4.40mm Width)
Supplier Device Package	16-TSSOP
Purchase URL	https://www.e-xfl.com/product-detail/nxp-semiconductors/s9s08sg32e1wtg

Table 1. MC9S08SG32 Rev. 1 Addendum

<table>
<tr><th>Location</th><th>Description</th></tr>
<tr><td>Chapter “Electrical Characteristics”/Section “Thermal Characteristics”/Table A-3. Thermal Characteristics/Page 293</td><td>Update Table A-3. Thermal Characteristics as follows:
• —Change the value for row “Thermal resistance,Single-layer board/28-pin TSSOP/Airflow @200ft/min.” from 71 to 72 C/W
—Change the value for 16-pin TSSOP/Thermalresistance
1.Single layer board / Airflow @ 200ft/min. from 108 to 113 C/W.
2.Four layer board / Airflow @ 200ft/min. from 78 to 84 C/W.
• Update parameter 4 of Table “A-3.Thermal Characteristics” .
 
<table>
<tr><th rowspan="2">#</th><th rowspan="2">C</th><th rowspan="2">Rating</th><th rowspan="2">Symbol</th><th rowspan="2">Value</th><th rowspan="2">Unit</th><th colspan="2">Temp Rated</th></tr>
<tr><th>Standard</th><th>AEC Grade 0</th></tr>
<tr><td rowspan="6">1</td><td rowspan="6">—</td><td colspan="6">Operating temperature range (packaged)</td></tr>
<tr><td>Temperature Code W</td><td rowspan="5">T_A</td><td>–40 to 150</td><td rowspan="5">°C</td><td>—</td><td>♦</td></tr>
<tr><td>Temperature Code J</td><td>–40 to 140</td><td>—</td><td>♦</td></tr>
<tr><td>Temperature Code M</td><td>–40 to 125</td><td>♦</td><td>—</td></tr>
<tr><td>Temperature Code V</td><td>–40 to 105</td><td>♦</td><td>—</td></tr>
<tr><td>Temperature Code C</td><td>–40 to 85</td><td>♦</td><td>—</td></tr>
<tr><td rowspan="4">2</td><td rowspan="4">D</td><td colspan="6">Thermal resistance, Single-layer board</td></tr>
<tr><td></td><td></td><td>Airflow @200 ft/min</td><td>Natural Convection</td><td></td><td></td><td></td></tr>
<tr><td>28-pin TSSOP</td><td rowspan="3">θ_{JA}</td><td>72</td><td>91</td><td rowspan="3">°C/W</td><td>♦</td><td>♦</td></tr>
<tr><td>20-pin TSSOP</td><td>94</td><td>114</td><td>♦</td><td>—</td></tr>
<tr><td>16-pin TSSOP</td><td>113</td><td>133</td><td>♦</td><td>♦</td></tr>
<tr><td rowspan="4">3</td><td rowspan="4">D</td><td colspan="6">Thermal resistance, Four-layer board</td></tr>
<tr><td></td><td></td><td>Airflow @200 ft/min</td><td>Natural Convection</td><td></td><td></td><td></td></tr>
<tr><td>28-pin TSSOP</td><td rowspan="3">θ_{JA}</td><td>51</td><td>58</td><td rowspan="3">°C/W</td><td>♦</td><td>♦</td></tr>
<tr><td>20-pin TSSOP</td><td>68</td><td>75</td><td>♦</td><td>—</td></tr>
<tr><td>16-pin TSSOP</td><td>84</td><td>92</td><td>♦</td><td>♦</td></tr>
<tr><td rowspan="6">4</td><td rowspan="6">D</td><td colspan="6">Maximum junction temperature</td></tr>
<tr><td>Temperature Code W</td><td rowspan="5">T_J</td><td>155</td><td rowspan="5">°C</td><td>—</td><td>♦</td></tr>
<tr><td>Temperature Code J</td><td>150</td><td>—</td><td>♦</td></tr>
<tr><td>Temperature Code M</td><td>135</td><td>♦</td><td>—</td></tr>
<tr><td>Temperature Code V</td><td>115</td><td>♦</td><td>—</td></tr>
<tr><td>Temperature Code C</td><td>95</td><td>♦</td><td>—</td></tr>
</table>
</td></tr>
<tr><td>Chapter “Electrical Characteristics”/Section “DC Characteristics”/Table A-6. DC Characteristics/Page 298</td><td>In the Table “DC Characteristics” add note 11 and 12 for parameter #18.
Note 11: Device functionality is guaranteed between the LVD threshold VLVD0 and VDD Min. When VDD is below the minimum operating voltage (VDD Min), the analog parameters for the IO pins, ACMP and ADC, are not guaranteed to meet data sheet performance parameters.
Note 12: In addition to LVD, it is recommended to also use the LVW feature. LVW can trigger an interrupt and be used as an indicator to warn that the VDD is dropping,so that the software can take actions accordingly before the VDD drops below VDD Min.</td></tr>
</table>

Contents

Section Number **Title** **Page**

Chapter 1
Device Overview

Chapter 2
Pins and Connections

Chapter 3
Modes of Operation

Chapter 4
Memory

Chapter 7
Central Processor Unit (S08CPUV3)

Chapter 8
Analog Comparator 5-V (S08ACMPV3)

Chapter 2
Pins and Connections

This section describes signals that connect to package pins. It includes pinout diagrams, recommended system connections, and detailed discussions of signals.

2.1 Device Pin Assignment

The following figures show the pin assignments for the MC9S08SG32 Series devices.

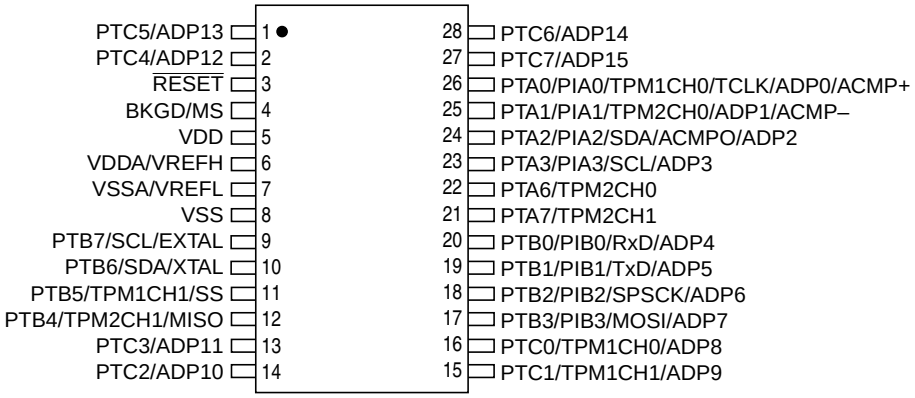


Figure 2-1. 28-Pin TSSOP

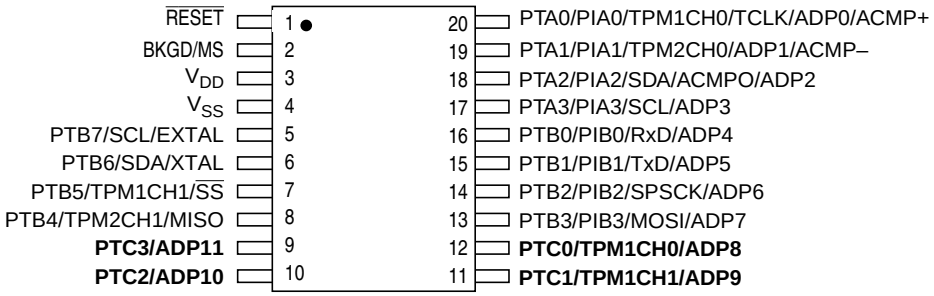


Figure 2-2. 20-Pin TSSOP[1]

1. 20-Pin TSSOP package not available for the high-temperature rated devices.

Background commands are of two types:

- Non-intrusive commands, defined as commands that can be issued while the user program is running. Non-intrusive commands can be issued through the BKGD/MS pin while the MCU is in run mode; non-intrusive commands can also be executed when the MCU is in the active background mode. Non-intrusive commands include:
  - Memory access commands
  - Memory-access-with-status commands
  - BDC register access commands
  - The BACKGROUND command
- Active background commands, which can only be executed while the MCU is in active background mode. Active background commands include commands to:
  - Read or write CPU registers
  - Trace one user program instruction at a time
  - Leave active background mode to return to the user application program (GO)

The active background mode is used to program a bootloader or user application program into the FLASH program memory before the MCU is operated in run mode for the first time. When the MC9S08SG32 Series is shipped from the Freescale Semiconductor factory, the FLASH program memory is erased by default unless specifically noted so there is no program that could be executed in run mode until the FLASH memory is initially programmed. The active background mode can also be used to erase and reprogram the FLASH memory after it has been previously programmed.

For additional information about the active background mode, refer to the Development Support chapter.

3.5 Wait Mode

Wait mode is entered by executing a WAIT instruction. Upon execution of the WAIT instruction, the CPU enters a low-power state in which it is not clocked. The I bit in CCR is cleared when the CPU enters the wait mode, enabling interrupts. When an interrupt request occurs, the CPU exits the wait mode and resumes processing, beginning with the stacking operations leading to the interrupt service routine.

While the MCU is in wait mode, there are some restrictions on which background debug commands can be used. Only the BACKGROUND command and memory-access-with-status commands are available when the MCU is in wait mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from wait mode and enter active background mode.

3.6 Stop Modes

One of two stop modes is entered upon execution of a STOP instruction when STOPE in SOPT1. In any stop mode, the bus and CPU clocks are halted. The ICS module can be configured to leave the reference clocks running. See Chapter 11, “Internal Clock Source (S08ICSV2),” for more information.

Most background commands are not available in stop mode. The memory-access-with-status commands do not allow memory access, but they report an error indicating that the MCU is in either stop or wait mode. The BACKGROUND command can be used to wake the MCU from stop and enter active background mode if the ENBDM bit is set. After entering background debug mode, all background commands are available.

3.6.2 Stop2 Mode

Stop2 mode is entered by executing a STOP instruction under the conditions as shown in Table 3-1. Most of the internal circuitry of the MCU is powered off in stop2 with the exception of the RAM. Upon entering stop2, all I/O pin control signals are latched so that the pins retain their states during stop2.

Exit from stop2 is performed by asserting the wake-up pin ($\overline{\text{RESET}}$) on the MCU.

In addition, the real-time counter (RTC) can wake the MCU from stop2, if enabled.

Upon wake-up from stop2 mode, the MCU starts up as from a power-on reset (POR):

- All module control and status registers are reset
- The LVD reset function is enabled and the MCU remains in the reset state if V_{DD} is below the LVD trip point (low trip point selected due to POR)
- The CPU takes the reset vector

In addition to the above, upon waking up from stop2, the PPDF bit in SPMSC2 is set. This flag is used to direct user code to go to a stop2 recovery routine. PPDF remains set and the I/O pin states remain latched until a 1 is written to PPDACK in SPMSC2.

To maintain I/O states for pins that were configured as general-purpose I/O before entering stop2, the user must restore the contents of the I/O port registers, which have been saved in RAM, to the port registers before writing to the PPDACK bit. If the port registers are not restored from RAM before writing to PPDACK, then the pins will switch to their reset states when PPDACK is written.

For pins that were configured as peripheral I/O, the user must reconfigure the peripheral module that interfaces to the pin before writing to the PPDACK bit. If the peripheral module is not enabled before writing to PPDACK, the pins will be controlled by their associated port control registers when the I/O latches are opened.

3.6.3 On-Chip Peripheral Modules in Stop Modes

When the MCU enters any stop mode, system clocks to the internal peripheral modules are stopped. Even in the exception case (ENBDM = 1), where clocks to the background debug logic continue to operate, clocks to the peripheral systems are halted to reduce power consumption. Refer to Section 3.6.2, “Stop2 Mode,” and Section 3.6.1, “Stop3 Mode,” for specific information on system behavior in stop modes.

Chapter 5
Resets, Interrupts, and General System Control

5.1 Introduction

This section discusses basic reset and interrupt mechanisms and the various sources of reset and interrupt in the MC9S08SG32 Series. Some interrupt sources from peripheral modules are discussed in greater detail within other sections of this data sheet. This section gathers basic information about all reset and interrupt sources in one place for easy reference. A few reset and interrupt sources, including the computer operating properly (COP) watchdog are not part of on-chip peripheral systems with their own chapters.

5.2 Features

Reset and interrupt features include:

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

5.3 MCU Reset

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

The MC9S08SG32 Series has the following sources for reset:

- Power-on reset (POR)
- External pin reset (PIN)
- Low-voltage detect (LVD)
- Computer operating properly (COP) timer
- Illegal opcode detect (ILOP)
- Illegal address detect (ILAD) - any address in memory map that is listed as unimplemented will produce an illegal address reset
- Background debug forced reset

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

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

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

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

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

5.5 Interrupts

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

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

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

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

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

5.7.1 System Reset Status Register (SRS)

This high page register includes read-only status flags to indicate the source of the most recent reset. When a debug host forces reset by writing 1 to BDFR in the SBDFR register, none of the status bits in SRS will be set. Writing any value to this register address causes a COP reset when the COP is enabled except the values 0x55 and 0xAA. Writing a 0x55-0xAA sequence to this address clears the COP watchdog timer without affecting the contents of this register. The reset state of these bits depends on what caused the MCU to reset.

	7	6	5	4	3	2	1	0
R	POR	PIN	COP	ILOP	ILAD	0	LVD	0
W	Writing 0x55, 0xAA to SRS address clears COP watchdog timer.							
POR:	1	0	0	0	0	0	1	0
LVR:	u[1]	0	0	0	0	0	1	0
Any other reset:	0	Note[2]	Note[2]	Note[2]	Note[2]	0	0	0

1 u = unaffected

2 Any of these reset sources that are active at the time of reset entry will cause the corresponding bit(s) to be set; bits corresponding to sources that are not active at the time of reset entry will be cleared.

Figure 5-2. System Reset Status (SRS)

Table 5-3. SRS Register Field Descriptions

Field	Description
7 POR	**Power-On Reset** — Reset was caused by the power-on detection logic. Because the internal supply voltage was ramping up at the time, the low-voltage reset (LVR) status bit is also set to indicate that the reset occurred while the internal supply was below the LVR threshold. 0 Reset not caused by POR. 1 POR caused reset.
6 PIN	**External Reset Pin** — Reset was caused by an active-low level on the external reset pin. 0 Reset not caused by external reset pin. 1 Reset came from external reset pin.
5 COP	**Computer Operating Properly (COP) Watchdog** — Reset was caused by the COP watchdog timer timing out. This reset source can be blocked by COPE = 0. 0 Reset not caused by COP timeout. 1 Reset caused by COP timeout.
4 ILOP	**Illegal Opcode** — Reset was caused by an attempt to execute an unimplemented or illegal opcode. The STOP instruction is considered illegal if stop is disabled by STOPE = 0 in the SOPT register. The BGND instruction is considered illegal if active background mode is disabled by ENBDM = 0 in the BDCSC register. 0 Reset not caused by an illegal opcode. 1 Reset caused by an illegal opcode.

7.3.5 Extended Addressing Mode (EXT)

In extended addressing mode, the full 16-bit address of the operand is located in the next two bytes of program memory after the opcode (high byte first).

7.3.6 Indexed Addressing Mode

Indexed addressing mode has seven variations including five that use the 16-bit H:X index register pair and two that use the stack pointer as the base reference.

7.3.6.1 Indexed, No Offset (IX)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction.

7.3.6.2 Indexed, No Offset with Post Increment (IX+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair as the address of the operand needed to complete the instruction. The index register pair is then incremented (H:X = H:X + 0x0001) after the operand has been fetched. This addressing mode is only used for MOV and CBEQ instructions.

7.3.6.3 Indexed, 8-Bit Offset (IX1)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.4 Indexed, 8-Bit Offset with Post Increment (IX1+)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction. The index register pair is then incremented (H:X = H:X + 0x0001) after the operand has been fetched. This addressing mode is used only for the CBEQ instruction.

7.3.6.5 Indexed, 16-Bit Offset (IX2)

This variation of indexed addressing uses the 16-bit value in the H:X index register pair plus a 16-bit offset included in the instruction as the address of the operand needed to complete the instruction.

7.3.6.6 SP-Relative, 8-Bit Offset (SP1)

This variation of indexed addressing uses the 16-bit value in the stack pointer (SP) plus an unsigned 8-bit offset included in the instruction as the address of the operand needed to complete the instruction.

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

Source Form	Operation	Address Mode	Object Code	Cycles	Cyc-by-Cyc Details	Affecton CCR	
						V 1 1 H	I N Z C

CMP #*oprx8i*
CMP *oprx8a*
CMP *oprx16a*
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 *oprx8a* 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 #*oprx8i* CPX *oprx8a* CPX *oprx16a* 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 *oprx8*,*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 *oprx8a* 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	fffffp	– 1 1 –	– – ↕ ↕
EOR #*oprx8i* EOR *oprx8a* EOR *oprx16a* 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 –	– ↕ ↕ –

8.7 Functional Description

The analog comparator can be used to compare two analog input voltages applied to ACMP+ and ACMP-; or it can be used to compare an analog input voltage applied to ACMP- with an internal bandgap reference voltage. ACBGS is used to select between the bandgap reference voltage or the ACMP+ pin as the input to the non-inverting input of the analog comparator. The comparator output is high when the non-inverting input is greater than the inverting input, and is low when the non-inverting input is less than the inverting input. ACMOD is used to select the condition which will cause ACF to be set. ACF can be set on a rising edge of the comparator output, a falling edge of the comparator output, or either a rising or a falling edge (toggle). The comparator output can be read directly through ACO. The comparator output can be driven onto the ACMPO pin using ACOPE.

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 $V_{DDA} / (2^N * I_{LEAK})$ 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 that 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_{DDA} to V_{SSA}.
- If inductive isolation is used from the primary supply, an additional 1 µF capacitor is placed from V_{DDA} to V_{SSA}.
- V_{SSA} (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 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_{SSA} (this improves noise issues, but affects the sample rate based on the external analog source resistance).

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

9.6.2.4 Code Width and Quantization Error

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

$$1_{LSB} = (V_{REFH} - V_{REFL}) / 2^N \qquad \textit{Eqn. 9-2}$$

There is an inherent quantization error due to the digitization of the result. For 8-bit or 10-bit conversions the code transitions when the voltage is at the midpoint between the points where the straight line transfer function is exactly represented by the actual transfer function. Therefore, the quantization error will be ± 1/2LSB in 8- or 10-bit mode. As a consequence, however, the code width of the first (0x000) conversion is only 1/2LSB and the code width of the last (0xFF or 0x3FF) is 1.5LSB.

9.6.2.5 Linearity Errors

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

- Zero-scale error (E_{ZS}) (sometimes called offset) — This error is defined as the difference between the actual code width of the first conversion and the ideal code width (1/2LSB). If the first conversion is 0x001, then the difference between the actual 0x001 code width and its ideal (1LSB) is used.
- Full-scale error (E_{FS}) — This error is defined as the difference between the actual code width of the last conversion and the ideal code width (1.5LSB). If the last conversion is 0x3FE, then the difference between the actual 0x3FE code width and its ideal (1LSB) is used.
- Differential non-linearity (DNL) — This error is defined as the worst-case difference between the actual code width and the ideal code width for all conversions.
- Integral non-linearity (INL) — This error is defined as the highest-value the (absolute value of the) running sum of DNL achieves. More simply, this is the worst-case difference of the actual transition voltage to a given code and its corresponding ideal transition voltage, for all codes.
- Total unadjusted error (TUE) — This error is defined as the difference between the actual transfer function and the ideal straight-line transfer function and includes all forms of error.

9.6.2.6 Code Jitter, Non-Monotonicity, and Missing Codes

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

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

converter yields the lower code (and vice-versa). However, even small amounts of system noise can cause the converter to be indeterminate (between two codes) for a range of input voltages around the transition voltage. This range is normally around 1/2LSB and increases with noise. This error may be reduced by repeatedly sampling the input and averaging the result. Additionally the techniques discussed in Section 9.6.2.3 reduces this error.

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

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

The TPM channels are programmable independently as input capture, output compare, or edge-aligned PWM channels. Alternately, the TPM can be configured to produce CPWM outputs on all channels. When the TPM is configured for CPWMs, the counter operates as an up/down counter; input capture, output compare, and EPWM functions are not practical.

If a channel is configured as input capture, an internal pullup device may be enabled for that channel. The details of how a module interacts with pin controls depends upon the chip implementation because the I/O pins and associated general purpose I/O controls are not part of the module. Refer to the discussion of the I/O port logic in a full-chip specification.

Because center-aligned PWMs are usually used to drive 3-phase AC-induction motors and brushless DC motors, they are typically used in sets of three or six channels.

16.2 Signal Description

Table 16-2 shows the user-accessible signals for the TPM. The number of channels may be varied from one to eight. When an external clock is included, it can be shared with the same pin as any TPM channel; however, it could be connected to a separate input pin. Refer to the I/O pin descriptions in full-chip specification for the specific chip implementation.

Table 16-2. Signal Properties

Name	Function
EXTCLK[1]	External clock source which may be selected to drive the TPM counter.
TPMxCHn[2]	I/O pin associated with TPM channel n

1 When preset, this signal can share any channel pin; however depending upon full-chip implementation, this signal could be connected to a separate external pin.

2 n=channel number (1 to 8)

Refer to documentation for the full-chip for details about reset states, port connections, and whether there is any pullup device on these pins.

TPM channel pins can be associated with general purpose I/O pins and have passive pullup devices which can be enabled with a control bit when the TPM or general purpose I/O controls have configured the associated pin as an input. When no TPM function is enabled to use a corresponding pin, the pin reverts to being controlled by general purpose I/O controls, including the port-data and data-direction registers. Immediately after reset, no TPM functions are enabled, so all associated pins revert to general purpose I/O control.

16.2.1 Detailed Signal Descriptions

This section describes each user-accessible pin signal in detail. Although Table 16-2 grouped all channel pins together, any TPM pin can be shared with the external clock source signal. Since I/O pin logic is not part of the TPM, refer to full-chip documentation for a specific derivative for more details about the interaction of TPM pin functions and general purpose I/O controls including port data, data direction, and pullup controls.

are used for PWM & output compare operation once normal execution resumes. Writes to the channel registers while BDM is active do not interfere with partial completion of a coherency sequence. After the coherency mechanism has been fully exercised, the channel registers are updated using the buffered values written (while BDM was not active) by the user.

16.4 Functional Description

All TPM functions are associated with a central 16-bit counter which allows flexible selection of the clock source and prescale factor. There is also a 16-bit modulo register associated with the main counter.

The CPWMS control bit chooses between center-aligned PWM operation for all channels in the TPM (CPWMS=1) or general purpose timing functions (CPWMS=0) where each channel can independently be configured to operate in input capture, output compare, or edge-aligned PWM mode. The CPWMS control bit is located in the main TPM status and control register because it affects all channels within the TPM and influences the way the main counter operates. (In CPWM mode, the counter changes to an up/down mode rather than the up-counting mode used for general purpose timer functions.)

The following sections describe the main counter and each of the timer operating modes (input capture, output compare, edge-aligned PWM, and center-aligned PWM). Because details of pin operation and interrupt activity depend upon the operating mode, these topics will be covered in the associated mode explanation sections.

16.4.1 Counter

All timer functions are based on the main 16-bit counter (TPMxCNTH:TPMxCNTL). This section discusses selection of the clock source, end-of-count overflow, up-counting vs. up/down counting, and manual counter reset.

16.4.1.1 Counter Clock Source

The 2-bit field, CLKSB:CLKSA, in the timer status and control register (TPMxSC) selects one of three possible clock sources or OFF (which effectively disables the TPM). See Table 16-4. After any MCU reset, CLKSB:CLKSA=0:0 so no clock source is selected, and the TPM is in a very low power state. These control bits may be read or written at any time and disabling the timer (writing 00 to the CLKSB:CLKSA field) does not affect the values in the counter or other timer registers.

A force-type breakpoint waits for the current instruction to finish and then acts upon the breakpoint request. The usual action in response to a breakpoint is to go to active background mode rather than continuing to the next instruction in the user application program.

The tag vs. force terminology is used in two contexts within the debug module. The first context refers to breakpoint requests from the debug module to the CPU. The second refers to match signals from the comparators to the debugger control logic. When a tag-type break request is sent to the CPU, a signal is entered into the instruction queue along with the opcode so that if/when this opcode ever executes, the CPU will effectively replace the tagged opcode with a BGND opcode so the CPU goes to active background mode rather than executing the tagged instruction. When the TRGSEL control bit in the DBGT register is set to select tag-type operation, the output from comparator A or B is qualified by a block of logic in the debug module that tracks opcodes and only produces a trigger to the debugger if the opcode at the compare address is actually executed. There is separate opcode tracking logic for each comparator so more than one compare event can be tracked through the instruction queue at a time.

17.3.5 Trigger Modes

The trigger mode controls the overall behavior of a debug run. The 4-bit TRG field in the DBGT register selects one of nine trigger modes. When TRGSEL = 1 in the DBGT register, the output of the comparator must propagate through an opcode tracking circuit before triggering FIFO actions. The BEGIN bit in DBGT chooses whether the FIFO begins storing data when the qualified trigger is detected (begin trace), or the FIFO stores data in a circular fashion from the time it is armed until the qualified trigger is detected (end trigger).

A debug run is started by writing a 1 to the ARM bit in the DBGC register, which sets the ARMF flag and clears the AF and BF flags and the CNT bits in DBGS. A begin-trace debug run ends when the FIFO gets full. An end-trace run ends when the selected trigger event occurs. Any debug run can be stopped manually by writing a 0 to ARM or DBGEN in DBGC.

In all trigger modes except event-only modes, the FIFO stores change-of-flow addresses. In event-only trigger modes, the FIFO stores data in the low-order eight bits of the FIFO.

The BEGIN control bit is ignored in event-only trigger modes and all such debug runs are begin type traces. When TRGSEL = 1 to select opcode fetch triggers, it is not necessary to use R/W in comparisons because opcode tags would only apply to opcode fetches that are always read cycles. It would also be unusual to specify TRGSEL = 1 while using a full mode trigger because the opcode value is normally known at a particular address.

The following trigger mode descriptions only state the primary comparator conditions that lead to a trigger. Either comparator can usually be further qualified with R/W by setting RWAEN (RWBEN) and the corresponding RWA (RWB) value to be matched against R/W. The signal from the comparator with optional R/W qualification is used to request a CPU breakpoint if BRKEN = 1 and TAG determines whether the CPU request will be a tag request or a force request.

17.4.3.9 Debug Status Register (DBGS)

This is a read-only status register.

	7	6	5	4	3	2	1	0
R	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
W								
Reset	0	0	0	0	0	0	0	0

= Unimplemented or Reserved

Figure 17-9. Debug Status Register (DBGS)

Table 17-6. DBGS Register Field Descriptions

Field	Description
7 AF	**Trigger Match A Flag** — AF is cleared at the start of a debug run and indicates whether a trigger match A condition was met since arming. 0 Comparator A has not matched 1 Comparator A match
6 BF	**Trigger Match B Flag** — BF is cleared at the start of a debug run and indicates whether a trigger match B condition was met since arming. 0 Comparator B has not matched 1 Comparator B match
5 ARMF	**Arm Flag** — While DBGEN = 1, this status bit is a read-only image of ARM in DBGC. This bit is set by writing 1 to the ARM control bit in DBGC (while DBGEN = 1) and is automatically cleared at the end of a debug run. A debug run is completed when the FIFO is full (begin trace) or when a trigger event is detected (end trace). A debug run can also be ended manually by writing 0 to ARM or DBGEN in DBGC. 0 Debugger not armed 1 Debugger armed
3:0 CNT[3:0]	**FIFO Valid Count** — These bits are cleared at the start of a debug run and indicate the number of words of valid data in the FIFO at the end of a debug run. The value in CNT does not decrement as data is read out of the FIFO. The external debug host is responsible for keeping track of the count as information is read out of the FIFO. 0000 Number of valid words in FIFO = No valid data 0001 Number of valid words in FIFO = 1 0010 Number of valid words in FIFO = 2 0011 Number of valid words in FIFO = 3 0100 Number of valid words in FIFO = 4 0101 Number of valid words in FIFO = 5 0110 Number of valid words in FIFO = 6 0111 Number of valid words in FIFO = 7 1000 Number of valid words in FIFO = 8

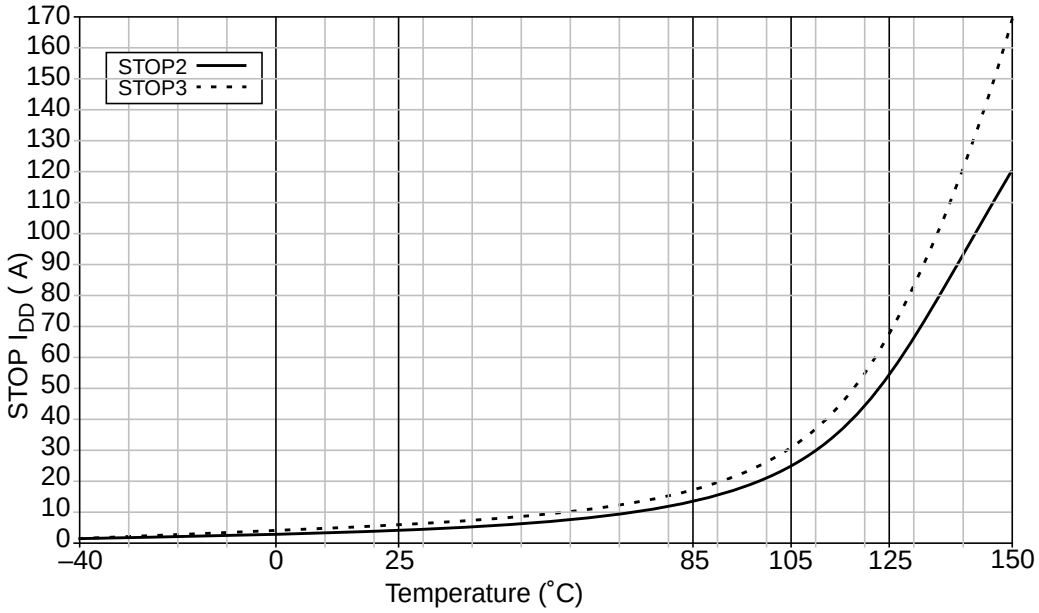


Figure A-7. Typical Stop I_{DD} vs. Temperature (V_{DD} = 5V)