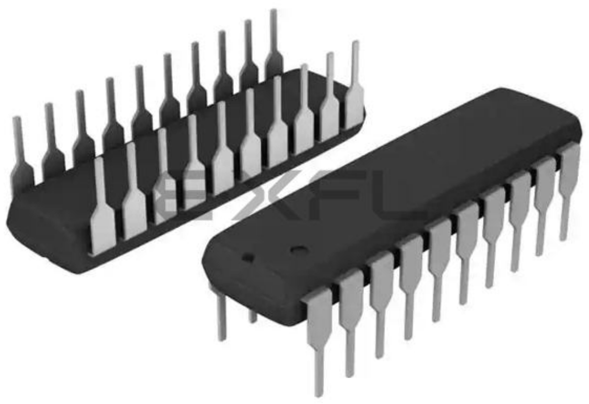
Zilog - Z8F1680PH020SG Datasheet



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	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, LINbus, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	17
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
Data Converters	A/D 7x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	20-DIP (0.300", 7.62mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1680ph020sg

Table of Contents

1.4. An Overview of the eZ8 CPU and its Peripherals

Zilog's eZ8 CPU, latest 8-bit CPU meets the continuing demand for faster and more code-efficient microcontrollers. It executes a superset of the original Z8® instruction set. The eZ8 CPU features include:

- Direct register-to-register architecture allows each register to function as an accumulator, improving execution time and decreasing the required program memory
- Software stack allows greater depth in subroutine calls and interrupts more than hardware stacks
- Compatible with existing Z8 code
- Expanded internal Register File allows access up to 4KB
- New instructions improve execution efficiency for code developed using higher-level programming languages including C
- Pipelined instruction fetch and execution
- New instructions for improved performance including BIT, BSWAP, BTJ, CPC, LDC, LDCI, LEA, MULT and SRL
- New instructions support 12-bit linear addressing of the register file
- Up to 10 MIPS operation
- C-Compiler friendly
- 2 to 9 clock cycles per instruction

For more details about eZ8 CPU, refer to the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com.

1.4.1. General-Purpose Input/Output

The F1680 MCU features 17 to 37 port pins (Ports A–E) for general purpose input/output (GPIO) pins. The number of GPIO pins available is a function of package. Each pin is individually programmable.

1.4.2. Flash Controller

The Flash Controller is used to program and erase Flash memory. The Flash Controller supports protection against accidental program and erasure.

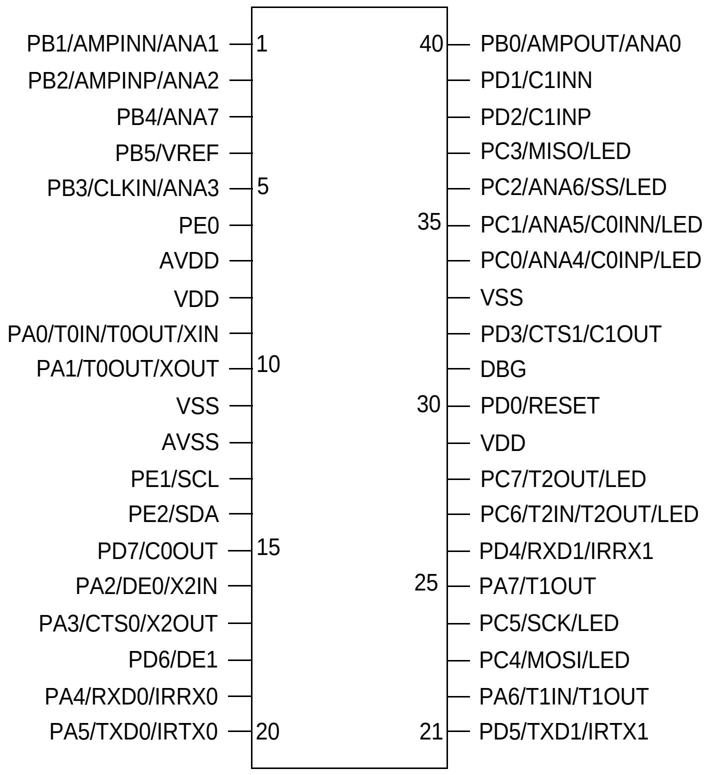


Figure 4. Z8F2480, Z8F1680 and Z8F0880 in 40-Pin Dual Inline Package (PDIP)

2.3. Signal Descriptions

Table 4 describes the signals for each block on the F1680 Series MCU. To determine the signals available for specific package styles, see the Pin Configurations chapter on page 10.

Table 4. Signal Descriptions

Signal Mnemonic	I/O	Description
General-Purpose I/O Ports A–E		
PA[7:0]	I/O	Port A: These pins are used for general-purpose I/O.
PB[5:0]	I/O	Port B: These pins are used for GPIO.
PC[7:0]	I/O	Port C: These pins are used for GPIO.
PD[7:0]	I/O	Port D: These pins are used for GPIO. PD0 is output only.
PE[6:0]	I/O	Port E: These pins are used for GPIO.
LIN-UART Controllers		
TXD0/TXD1	O	Transmit Data 0–1: These signals are the transmit output from the UART0/1 and IrDA0/1.
RXD0/RXD1	I	Receive Data 0–1: These signals are the receive input for the UART0/1 and IrDA0/1.
CTS0/CTS1	I	Clear To Send 0–1: These signals are the flow control input for the UART0/1.
DE0/DE1	O	Driver Enable 0–1: These signals allow automatic control of external RS-485 drivers. These signals are approximately the inverse of the TXE (Transmit Empty) bit in the UART Status 0/1 register. The DE0/1 signal can be used to ensure the external RS-485 driver is enabled when data is transmitted by the UART0/1.
I^{2}C Controller		
SCL	I/O	I^{2}C Serial Clock: The I^{2}C Master supplies this signal. If the F1680 Series MCU is the I^{2}C Master, this pin is an output. If it is the I^{2}C slave, this pin is an input. When the GPIO pin is configured as an alternate function to enable the SCL function, this pin is open-drain.
SDA	I/O	Serial Data: This open-drain pin transfers data between the I^{2}C and an external I^{2}C Master/Slave. When the GPIO pin is configured as an alternate function to enable the SDA function, this pin is open-drain.
ESPI Controller		
$\overline{\text{SS}}$	I/O	Slave Select: This signal can be an output or an input. If the F1680 Series MCU is the SPI master, this pin can be configured as the Slave Select output. If it is the SPI slave, this pin is the input slave select.

Table 8. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)[1]	Page #
Watchdog Timer				
FF2	Watchdog Timer Reload High Byte	WDTH	FF	143
FF3	Watchdog Timer Reload Low Byte	WDTL	FF	143
FF4–FF5	Reserved	—	XX	
Trim Bit Control				
FF6	Trim Bit Address	TRMADR	00	281
FF7	Trim Data	TRMDR	XX	281
Flash Memory Controller				
FF8	Flash Control	FCTL	00	272
	Flash Status	FSTAT	00	272
FF9	Flash Page Select	FPS	00	273
	Flash Sector Protect	FPROT	00	274
FFA	Flash Programming Frequency High Byte	FFREQH	00	275
FFB	Flash Programming Frequency Low Byte	FFREQL	00	275
eZ8 CPU				
FFC	Flags	—	XX	refer to the eZ8 CPU Core User Manual (UM0128)
FFD	Register Pointer	RP	XX	
FFE	Stack Pointer High Byte	SPH	XX	
FFF	Stack Pointer Low Byte	SPL	XX	

Notes:
1. XX=Undefined.
2. The Reserved space can be configured as General-Purpose Register File RAM depending on the user option bits (see the User Option Bits chapter on page 277) and the on-chip PRAM size (see the Ordering Information chapter on page 372). If the PRAM is programmed as General-Purpose Register File RAM on Reserved space, the starting address always begins immediately after the end of General-Purpose Register File RAM.

Chapter 9. Timers

The Z8 Encore! XP F1680 Series products contain three 16-bit reloadable timers that can be used for timing, event counting, or generation of pulse-width modulated signals. The timers' features include:

- 16-bit reload counter
- Programmable prescaler with prescale values ranging from 1 to 128
- PWM output generation
- Capture and compare capability
- Two independent capture/compare channels which reference the common timer
- External input pin for timer input, clock gating, or capture signal. External input pin signal frequency is limited to a maximum of one-fourth the timer clock frequency
- Timer output pin
- Timer interrupt
- Noise Filter on Timer input signal
- Operation in any mode with 32 kHz secondary oscillator

In addition to the timers described in this chapter, the Baud Rate Generator (BRG) of unused UART peripheral can also be used to provide basic timing functionality. For more information about using the Baud Rate Generator as additional timers, see the LIN-UART chapter on page 144.

10.5. Low-Power Modes

The Z8 Encore! XP F1680 Series of MCUs contains power-saving features. The highest level of power reduction is provided by STOP Mode. The next level of power reduction is provided by HALT Mode.

10.5.1. Operation in HALT Mode

When the eZ8 CPU is operating in HALT Mode, the Multi-Channel Timer will continue to operate if enabled. To minimize current in HALT Mode, the Multi-Channel Timer must be disabled by clearing the TEN control bit.

10.5.2. Operation in STOP Mode

When the eZ8 CPU is operating in STOP Mode, the Multi-Channel Timer ceases to operate because the system clock has stopped. The registers are not reset and operation will resume after Stop Mode Recovery occurs.

10.5.3. Power Reduction During Operation

Deassertion of the TEN bit will inhibit clocking of the entire Multi-Channel Timer block. Deassertion of the CHEN bit of individual channels will inhibit clocking of channel-specific logic to minimize power consumption of unused channels. The CPU can still read and write to the registers when the enable bit(s) are deasserted.

10.6. Multi-Channel Timer Applications Examples

This section provides two brief examples that describe how the the F1680 Series multi-channel timer can be used in your application.

10.6.1. PWM Programmable Deadband Generation

The count up/down mode supports motor control applications that require dead time between output signals. Figure 17 displays dead time generation between two channels operating in count up/down mode.

11.1.2.3. WDT Reset in Normal Operation

The WDT forces the device into the Reset state if it is configured to generate a Reset when a time-out occurs; the WDT status bit is set to 1 (for details, see the Reset Status Register section on page 40). For more information about Reset and the WDT status bit, see the Reset, Stop Mode Recovery and Low-Voltage Detection section on page 31. Following a Reset sequence, the WDT Counter is initialized with its reset value.

11.1.2.4. WDT Reset in STOP Mode

If enabled in STOP Mode and configured to generate a Reset when a time-out occurs and the device is in STOP Mode, the WDT initiates a Stop Mode Recovery. Both the WDT status bit and the stop bit in the Reset Status Register (RSTSTAT) are set to 1 following a WDT time-out in STOP Mode. For more information, see the Reset, Stop Mode Recovery and Low-Voltage Detection section on page 31.

11.1.3. Watchdog Timer Reload Unlock Sequence

Writing the unlock sequence to the Watchdog Timer Reload High (WDTH) Register address unlocks the two Watchdog Timer Reload registers (WDTH and WDTL) to allow changes to the time-out period. These write operations to the WDTH Register address produce no effect on the bits in the WDTH Register. The locking mechanism prevents unwarranted writes to the Reload registers. The following sequence is required to unlock the Watchdog Timer Reload registers (WDTH and WDTL) for write access.

1. Write `55H` to the Watchdog Timer Reload High Register (WDTH).
2. Write `AAH` to the Watchdog Timer Reload High Register (WDTH).
3. Write the appropriate value to the Watchdog Timer Reload High Register (WDTH).
4. Write the appropriate value to the Watchdog Timer Reload Low Register (WDTL). After this write occurs, the Watchdog Timer Reload registers are again locked.

All steps of the WDT Reload Unlock sequence must be written in the sequence defined above. The values in these WDT Reload registers are loaded into the counter every time a WDT instruction is executed.

11.2. Watchdog Timer Register Definitions

The two Watchdog Timer Reload registers (WDTH and WDTL) are described in the following tables.

The LIN-UART is now configured for interrupt-driven data reception. When the LIN-UART Receiver interrupt is detected, the associated ISR performs the following:

1. Checks the LIN-UART Status 0 Register to determine the source of the interrupt-error, break, or received data.
2. If the interrupt is due to data available, read the data from the LIN-UART Receive Data Register. If operating in MULTIPROCESSOR (9-bit) Mode, further actions may be required depending on the MULTIPROCESSOR Mode bits MPMD[1:0].
3. Execute the IRET instruction to return from the ISR and await more data.

12.1.6. Clear To Send Operation

The Clear To Send ($\overline{\text{CTS}}$) pin, if enabled by the CTSE bit of the LIN-UART Control 0 Register performs flow control on the outgoing transmit data stream. The Clear To Send ($\overline{\text{CTS}}$) input pin is sampled one system clock before any new character transmission begins. To delay transmission of the next data character, an external receiver must reduce $\overline{\text{CTS}}$ at least one system clock cycle before a new data transmission begins. For multiple character transmissions, this operation is typically performed during the stop bit transmission. If $\overline{\text{CTS}}$ stops in the middle of a character transmission, the current character is sent completely.

12.1.7. External Driver Enable

The LIN-UART provides a Driver Enable (DE) signal for off-chip bus transceivers. This feature reduces the software overhead associated using a GPIO pin to control the transceiver when communicating on a multitransceiver bus, such as RS-485.

Driver Enable is a programmable polarity signal which envelopes the entire transmitted data frame including parity and stop bits as illustrated in Figure 22. The Driver Enable signal asserts when a byte is written to the LIN-UART Transmit Data Register. The Driver Enable signal asserts at least one bit period and no greater than two bit periods before the start bit is transmitted. This allows a set-up time to enable the transceiver. The Driver Enable signal deasserts one system clock period after the last stop bit is transmitted. This system clock delay allows both time for data to clear the transceiver before disabling it, as well as the ability to determine if another character follows the current character. In the event of back-to-back characters (new data must be written to the Transmit Data Register before the previous character is completely transmitted) the DE signal is not deasserted between characters. The DEPOL bit in the LIN-UART Control Register 1 sets the polarity of the Driver Enable signal.

Table 100. LIN-UART Baud Rates, 1.8432 MHz System Clock

Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error(%)	Applicable Rate (kHz)	BRG Divisor (Decimal)	Actual Rate (kHz)	Error(%)
1250.0	N/A	N/A	N/A	9.60	12	9.60	0.00
625.0	N/A	N/A	N/A	4.80	24	4.80	0.00
250.0	N/A	N/A	N/A	2.40	48	2.40	0.00
115.2	1	115.2	0.00	1.20	96	1.20	0.00
57.6	2	57.6	0.00	0.60	192	0.60	0.00
38.4	3	38.4	0.00	0.30	384	0.30	0.00
19.2	6	19.2	0.00				

14.3.4. ADC Data Low Bits Register

The ADC Data Low Bits Register, shown in Table 104, contains the lower bits of the ADC output as well as an overflow status bit. Access to the ADC Data Low Bits Register is read-only. Reading the ADC Data High Byte Register latches data in the ADC Low Bits Register.

Table 104. ADC Data Low Bits Register (ADCD_L)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	ADCDL		Reserved					
Reset	X		X					
R/W	R		R					
Address	F73H							

Bit Position	**Value (H)**	**Description**
[7:6] ADCDL	00– 11b	**ADC Low Bit** These bits are the 2 least significant bits of the 10-bit ADC output; they are undefined after a Reset. The Low bits are latched into this register whenever the ADC Data High Byte register is read.
[5:0]	0	Reserved; must be 0.

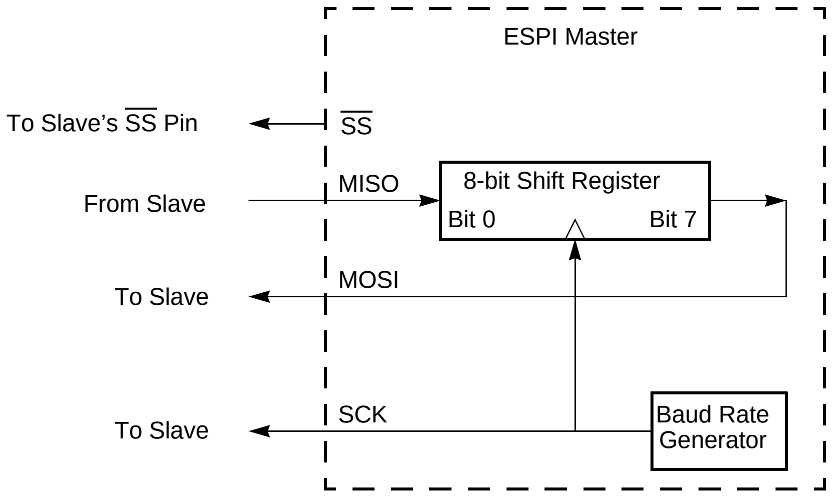


Figure 39. ESPI Configured as an SPI Master in a Single Master, Single Slave System

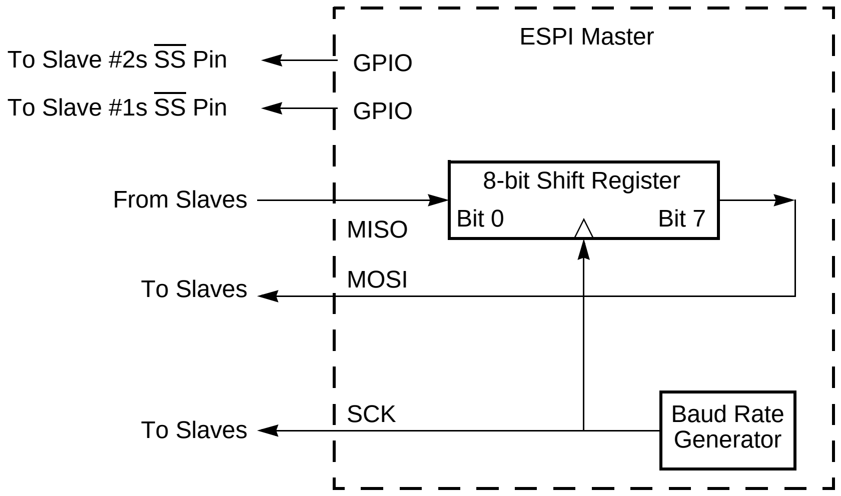


Figure 40. ESPI Configured as an SPI Master in a Single Master, Multiple Slave System

16.3.4.2. Multi-Master SPI Operation

In a Multi-Master SPI system, all SCK pins are tied together, all MOSI pins are tied together and all MISO pins are tied together. All SPI pins must be configured in open-drain mode to prevent bus contention. At any time, only one SPI device is configured as the Master and all other devices on the bus are configured as slaves. The Master asserts the

d. Set IEN = 1 in the I^{2}C Control Register. Set NAK = 0 in the I^{2}C Control Register.

2. The Master initiates a transfer by sending the address byte. The SLAVE Mode I^{2}C controller finds an address match and detects that the R/$\overline{W}$ bit = 1 (read by the master from the slave). The I^{2}C controller acknowledges, indicating that it is ready to accept the transaction. The SAM bit in the I2CISTAT Register is set to 1, causing an interrupt. The RD bit is set to 1, indicating a Read from the slave.
3. The software responds to the interrupt by reading the I2CISTAT Register, thereby clearing the SAM bit. Because RD = 1, the software responds by loading the first data byte into the I2CDATA Register. The software sets the TXI bit in the I2CCTL Register to enable transmit interrupts. When the master initiates the data transfer, the I^{2}C controller holds SCL Low until the software has written the first data byte to the I2CDATA Register.
4. SCL is released and the first data byte is shifted out.
5. After the first bit of the first data byte has been transferred, the I^{2}C controller sets the TDRE bit, which asserts the transmit data interrupt.
6. The software responds to the transmit data interrupt (TDRE = 1) by loading the next data byte into the I2CDATA Register, which clears TDRE.
7. After the data byte has been received by the master, the master transmits an Acknowledge instruction (or Not Acknowledge instruction if this byte is the final data byte).
8. The bus cycles through Step 5 to Step 7 until the final byte has been transferred. If the software has not yet loaded the next data byte when the master brings SCL Low to transfer the most significant data bit, the slave I^{2}C controller holds SCL Low until the Data Register has been written. When a Not Acknowledge instruction is received by the slave, the I^{2}C controller sets the NCKI bit in the I2CISTAT Register causing the Not Acknowledge interrupt to be generated.
9. The software responds to the Not Acknowledge interrupt by clearing the TXI bit in the I2CCTL Register and by asserting the FLUSH bit of the I2CCTL Register to *empty* the Data Register.
10. When the Master has completed the final acknowledge cycle, it asserts a stop or restart condition on the bus.
11. The Slave I^{2}C controller asserts the stop/restart interrupt (set SPRS bit in I2CISTAT Register).
12. The software responds to the stop/restart interrupt by reading the I2CISTAT Register, which clears the SPRS bit.

17.2.6.8. Slave Transmit Transaction With 10-Bit Address

The data transfer format for a master reading data from a slave with 10-bit addressing is displayed in Figure 50. The following procedure describes the I^{2}C Master/Slave Controller operating as a slave in 10-bit addressing mode, transmitting data to the bus master.

S	Slave Address 1st Byte	W = 0	A	Slave Address 2nd Byte	A	S	Slave Address 1st Byte	R = 1	A	Data	A	Data	A	P

Figure 50. Data Transfer Format—Slave Transmit Transaction with 10-Bit Address

1. The software configures the controller for operation as a slave in 10-bit addressing mode.
   a. Initialize the MODE field in the I^{2}C Mode Register for either SLAVE ONLY mode or MASTER/SLAVE Mode with 10-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[7:0] bits in the I2CSLVAD Register and SLA[9:8] in the I^{2}C MODE Register.
   d. Set IEN = 1 and NAK = 0 in the I^{2}C Control Register.
2. The Master initiates a transfer by sending the first address byte. The SLAVE Mode I^{2}C controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects R/W bit = 0 (a Write from the master to the slave). The I^{2}C controller acknowledges indicating it is available to accept the transaction.
3. The Master sends the second address byte. The SLAVE Mode I^{2}C controller compares the second address byte with the value in SLA[7:0]. If there is a match, the SAM bit in the I2CISTAT Register is set = 1, causing a slave address match interrupt. The RD bit is set = 0, indicating a write to the slave. If a match occurs, the I^{2}C controller acknowledges on the I^{2}C bus, indicating it is available to accept the data.
4. The software responds to the slave address match interrupt by reading the I2CISTAT Register, which clears the SAM bit. Because the RD bit = 0, no further action is required.
5. The Master sees the Acknowledge and sends a restart instruction, followed by the first address byte with R/W set to 1. The SLAVE Mode I^{2}C controller recognizes the restart instruction followed by the first address byte with a match to SLA[9:8] and detects R/W = 1 (the master reads from the slave). The slave I^{2}C controller sets the SAM bit in the I2CISTAT Register which causes the slave address match interrupt. The RD bit is set = 1. The SLAVE Mode I^{2}C controller acknowledges on the bus.
6. The software responds to the interrupt by reading the I2CISTAT Register clearing the SAM bit. The software loads the initial data byte into the I2CDATA Register and sets the TXI bit in the I2CCTL Register.
7. The Master starts the data transfer by asserting SCL Low. After the I^{2}C controller has data available to transmit, the SCL is released and the master proceeds to shift the first data byte.

17.3.2. I²C Interrupt Status Register

The read-only I²C Interrupt Status Register, shown in Table 120, indicates the cause of any current I²C interrupt and provides status of the I²C controller. When an interrupt occurs, one or more of the TDRE, RDRF, SAM, ARBLST, SPRS or NCKI bits is set. The GCA and RD bits do not generate an interrupt but rather provide status associated with the SAM bit interrupt.

Table 120. I²C Interrupt Status Register (I2CISTAT = F51H)

Bits	**7**	**6**	**5**	**4**	**3**	**2**	**1**	**0**
Field	TDRE	RDRF	SAM	GCA	RD	ARBLST	SPRS	NCKI
Reset	1	0	0	0	0	0	0	0
R/W	R	R	R	R	R	R	R	R
Address	F51H							

Bit	**Description**
[7] TDRE	**Transmit Data Register Empty** When the I²C controller is enabled, this bit is 1 when the I²C Data Register is empty. When set, this bit causes the I²C controller to generate an interrupt, except when the I²C controller is shifting in data during the reception of a byte or when shifting an address and the RD bit is set. This bit clears by writing to the I2CDATA Register.
[6] RDRF	**Receive Data Register Full** This bit is set = 1 when the I²C controller is enabled and the I²C controller has received a byte of data. When asserted, this bit causes the I²C controller to generate an interrupt. This bit clears by reading the I2CDATA Register.
[5] SAM	**Slave Address Match** This bit is set = 1 if the I²C controller is enabled in SLAVE Mode and an address is received that matches the unique slave address or General Call Address (if enabled by the GCE bit in the I²C Mode Register). In 10-bit addressing mode, this bit is not set until a match is achieved on both address bytes. When this bit is set, the RD and GCA bits are also valid. This bit clears by reading the I2CISTAT Register.
[4] GCA	**General Call Address** This bit is set in SLAVE Mode when the General Call Address or Start byte is recognized (in either 7 or 10 bit SLAVE Mode). The GCE bit in the I²C Mode Register must be set to enable recognition of the General Call Address and Start byte. This bit clears when IEN = 0 and is updated following the first address byte of each SLAVE Mode transaction. A General Call Address is distinguished from a Start byte by the value of the RD bit (RD = 0 for General Call Address, 1 for Start byte).
[3] RD	**Read** This bit indicates the direction of transfer of the data. It is set when the Master is reading data from the Slave. This bit matches the least-significant bit of the address byte after the start condition occurs (for both MASTER and SLAVE modes). This bit clears when IEN = 0 and is updated following the first address byte of each transaction.

The Flash Sector Protect Register can be configured to prevent sectors from being programmed or erased. After a sector is protected, it cannot be unprotected by user code. The Flash Sector Protect Register is cleared after reset and any previously written protection values is lost. User code must write this register in their initialization routine if they want to enable sector protection.

The Flash Sector Protect Register shares its Register File address with the Page Select Register. The Flash Sector Protect Register is accessed by writing the Flash Control Register with `5EH`. After the Flash Sector Protect Register is selected, it can be accessed at the Page Select Register address. When user code writes the Flash Sector Protect Register, bits can only be set to 1. Thus, sectors can be protected, but not unprotected, via register write operations. Writing a value other than `5EH` to the Flash Control Register deselects the Flash Sector Protect Register and reenables access to the Page Select Register. code:

1. Write `00H` to the Flash Control Register to reset the Flash Controller.
2. Write `5EH` to the Flash Control Register to select the Flash Sector Protect Register.
3. Read and/or write the Flash Sector Protect Register which is now at Register File address `FF9H`.
4. Write `00H` to the Flash Control Register to return the Flash Controller to its reset state.

The Sector Protect Register is initialized to 0 on Reset, putting each sector into an unprotected state. When a bit in the Sector Protect Register is written to 1, the corresponding sector can no longer be written or erased. After a bit of the Sector Protect Register has been set, it can not be cleared except by a System Reset.

20.2.4. Byte Programming

Flash memory is enabled for byte programming on the active page after unlocking the Flash Controller. Erase the address(es) to be programmed using either the Page Erase or Mass Erase command prior to performing byte programming. An erased Flash byte contains all 1s (`FFH`). The programming operation can only be used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from 0 to 1 requires execution of either the Page Erase or Mass Erase command.

Byte programming can be accomplished using the On-Chip Debugger's Write Memory command or eZ8 CPU execution of the LDC or LDCI instructions. For a description of the LDC and LDCI instructions, refer to the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com. While the Flash Controller programs the contents of Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate.

After a byte is written, the page remains unlocked, allowing for subsequent writes to other bytes on the same page. To exit programming mode and lock Flash memory, write any value to the Flash Control Register except the Mass Erase or Page Erase commands.

enabled, the OCD ignores the BRK signal and the BRK instruction operates as an NOP instruction.

If breakpoints are enabled, the OCD can be configured to automatically enter DEBUG mode, or to loop on the break instruction. If the OCD is configured to loop on the BRK instruction, then the CPU remains able to service interrupt requests.

The loop on a BRK instruction can service interrupts in the background. For interrupts to be serviced in the background, there cannot be any breakpoints in the interrupt service routine. Otherwise, the CPU stops on the breakpoint in the interrupt routine. For interrupts to be serviced in the background, interrupts must also be enabled. Interrupts are typically disabled during critical sections of code where interrupts do not occur (such as adjusting the stack pointer or modifying shared data).

Through the OCD, host debugger software can poll the IDLE bit of the OCDSTAT Register to determine if the OCD is looping on a BRK instruction. When the host must stop the CPU on the BRK instruction on which it is looping, the host must not set the DBGMODE bit of the OCDCTL register. The CPU may have vectored to an interrupt service routine. Instead, the host clears the BRKLOOP bit, thereby allowing the CPU to finish the interrupt service routine and return to the BRK instruction. When the CPU returns to the BRK instruction on which it was previously looping, it automatically sets the DBGMODE bit and enters DEBUG mode.

The majority of the OCD commands remain disabled when the eZ8 CPU is looping on a BRK instruction. The eZ8 CPU must be in DEBUG mode before these commands can be issued.

23.2.8.1. Breakpoints in Flash Memory

The BRK instruction is op code `00H`, which corresponds to the fully programmed state of a byte in Flash memory. To implement a breakpoint, write `00H` to the appropriate address, overwriting the current instruction. To remove a breakpoint, erase the corresponding page of Flash memory and reprogram with the original data.

23.2.9. OCDCNTR Register

The On-Chip Debugger contains a multipurpose 16-bit Counter Register. It can be used for the following:

- Count system clock cycles between breakpoints
- Generate a BRK when it counts down to 0
- Generate a BRK when its value matches the Program Counter

When configured as a counter, the OCDCNTR Register starts counting when the On-Chip Debugger exits DEBUG mode and stops counting when it enters DEBUG mode again or

---

! **Caution:** When using the external RC oscillator mode, the oscillator can stop oscillating if the power supply drops below 1.6V but remains above the Voltage Brown-Out threshold. The oscillator resumes oscillation when the supply voltage exceeds 1.6V.

25.4. Secondary Crystal Oscillator Operation

Figure 65 displays the recommended configuration for connection with an external fundamental-mode, parallel-resonant crystal operating at 32kHz. The recommended 32kHz crystal specifications are provided in Table 173. Printed circuit board layout must add no more than 4pF of stray capacitance to either the X_{IN} or X_{OUT} pins. If oscillation does not occur, reduce the values of capacitors C_1 and C_2 to decrease loading.

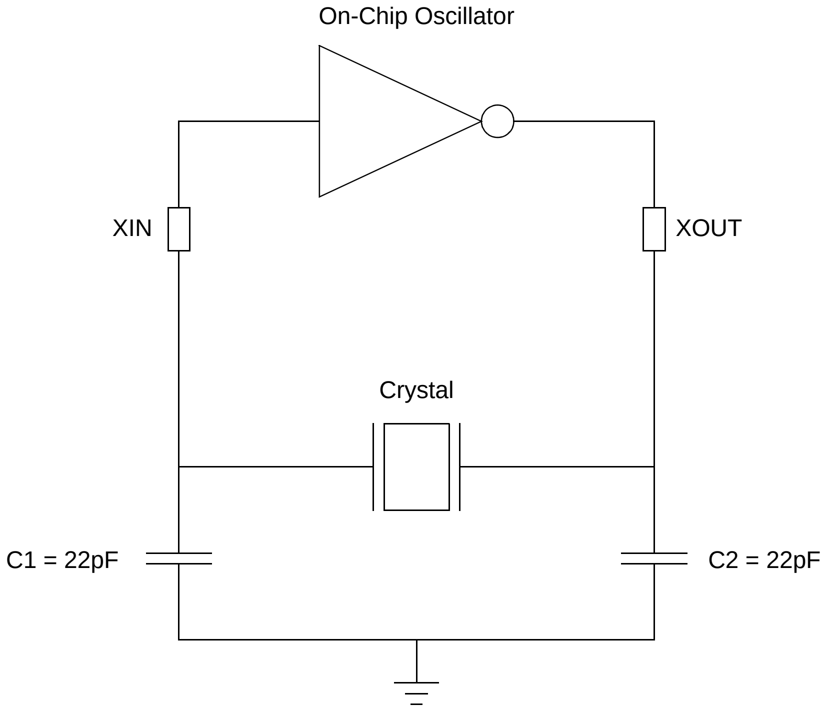


Figure 65. Recommended 32kHz Crystal Oscillator Configuration

Customer Support

To share comments, get your technical questions answered, or report issues you may be experiencing with our products, please visit Zilog's Technical Support page at http://support.zilog.com.

To learn more about this product, find additional documentation, or to discover other facets about Zilog product offerings, please visit the Zilog Knowledge Base at http://zilog.com/kb or consider participating in the Zilog Forum at http://zilog.com/forum.

This publication is subject to replacement by a later edition. To determine whether a later edition exists, please visit the Zilog website at http://www.zilog.com.