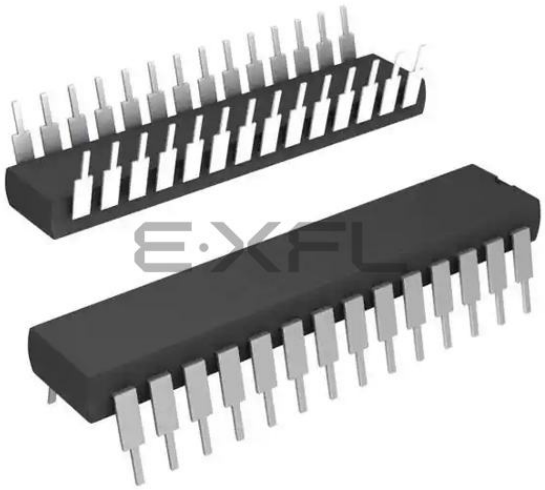

Welcome to E-XFL.COM

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, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	23
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 8x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Through Hole
Package / Case	28-DIP (0.600", 15.24mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1680pj020sg

List of Tables

1.3. Block Diagram

Figure 1 displays the architecture of the F1680 Series MCU.

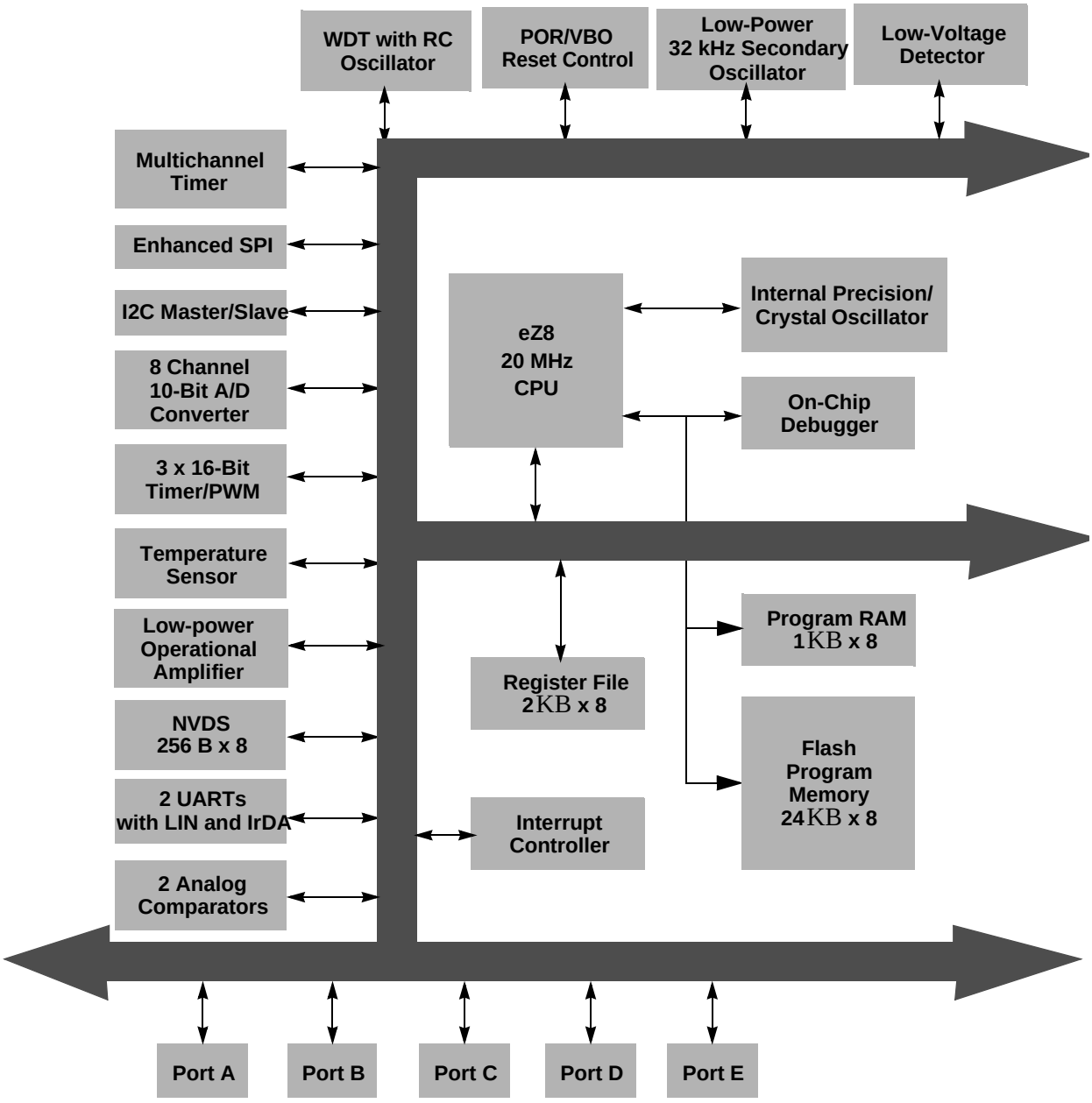

Figure 1. F1680 Series MCU Block Diagram

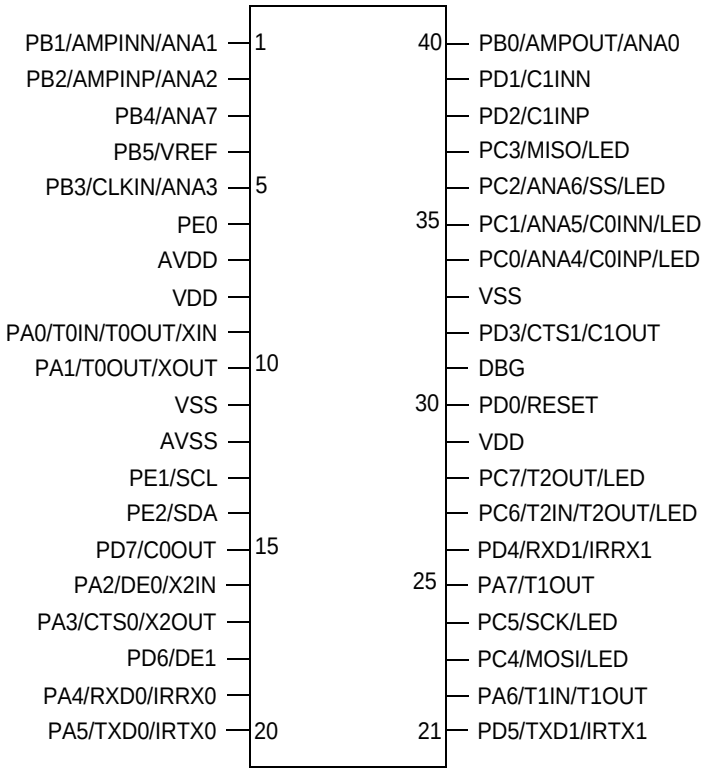

Figure 4. Z8F2480, Z8F1680 and Z8F0880 in 40-Pin Dual Inline Package (PDIP)

Chapter 4. Register Map

Table 8 provides an address map to the register file contained in all Z8 Encore! XP F1680 Series devices. Not all devices and package styles in this product series support the ADC, nor all of the GPIO ports. Therefore, consider the registers for unimplemented peripherals to be reserved.

Table 8. Register File Address Map

Address (Hex)	Register Description	Mnemonic	Reset (Hex)[1]	Page #
General Purpose RAM				
Z8F2480 Device				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved[2]	—	XX	
Z8F1680 Device				
000–7FF	General-Purpose Register File RAM	—	XX	
800–EFF	Reserved[2]	—	XX	
Z8F0880 Device				
000–3FF	General-Purpose Register File RAM	—	XX	
400–EFF	Reserved[2]	—	XX	
Special Purpose Registers				
Timer 0				
F00	Timer 0 High Byte	T0H	00	109
F01	Timer 0 Low Byte	T0L	01	109
F02	Timer 0 Reload High Byte	T0RH	FF	110
F03	Timer 0 Reload Low Byte	T0RL	FF	110
F04	Timer 0 PWM0 High Byte	T0PWM0H	00	110
F05	Timer 0 PWM0 Low Byte	T0PWM0L	00	111
F06	Timer 0 Control 0	T0CTL0	00	112
F07	Timer 0 Control 1	T0CTL1	00	113

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.

Table 18. Port Alternate Function Mapping, 28-Pin Parts[1,2]

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A	PA0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0
		Reserved		AFS1[1]: 1
	PA2	DE0	UART 0 Driver Enable	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PA3	$\overline{\text{CTS0}}$	UART 0 Clear to Send	AFS1[3]: 0
		Reserved		AFS1[3]: 1
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data	AFS1[4]: 0
		Reserved		AFS1[4]: 1
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data	AFS1[5]: 0
		Reserved		AFS1[5]: 1
	PA6	T1IN/$\overline{\text{T1OUT}}$	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0
		SCL	I^2C Serial Clock	AFS1[6]: 1
	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0
		SDA	I^2C Serial Data	AFS1[7]: 1

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A and B, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.

Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2]

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port A	PA0	T0IN/$\overline{\text{T0OUT}}$	Timer 0 Input/Timer 0 Output Complement	AFS1[0]: 0
		Reserved		AFS1[0]: 1
	PA1	T0OUT	Timer 0 Output	AFS1[1]: 0
		Reserved		AFS1[1]: 1
	PA2	DE0	UART 0 Driver Enable	AFS1[2]: 0
		Reserved		AFS1[2]: 1
	PA3	$\overline{\text{CTS0}}$	UART 0 Clear to Send	AFS1[3]: 0
		Reserved		AFS1[3]: 1
	PA4	RXD0/IRRX0	UART 0/IrDA 0 Receive Data	AFS1[4]: 0
				AFS1[4]: 1
	PA5	TXD0/IRTX0	UART 0/IrDA 0 Transmit Data	AFS1[5]: 0
				AFS1[5]: 1
	PA6	T1IN/$\overline{\text{T1OUT}}$	Timer 1 Input/Timer 1 Output Complement	AFS1[6]: 0
		Reserved		AFS1[6]: 1
	PA7	T1OUT	Timer 1 Output	AFS1[7]: 0
		Reserved		AFS1[7]: 1

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

7.11.5. Port A–E Output Control Subregisters

The Port A–E Output Control Subregister, shown in Table 25, is accessed through the Port A–E Control Register by writing 03H to the Port A–E Address Register. Setting the bits in the Port A–E Output Control subregisters to 1 configures the specified port pins for open-drain operation. These subregisters affect the pins directly and, as a result, alternate functions are also affected.

Table 25. Port A–E Output Control Subregisters (PxOC)

Bits	7	6	5	4	3	2	1	0
Field	POC7	POC6	POC5	POC4	POC3	POC2	POC1	POC0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 03H in Port A–E Address Register, accessible through the Port A–E Control Register							

Bit	Description
[7:0] POC	**Port Output Control** These bits function independently of the alternate function bit and always disable the drains if set to 1. 0 = The drains are enabled for any output mode (unless overridden by the alternate function). 1 = The drain of the associated pin is disabled (open-drain mode).

7.11.6. Port A–E High Drive Enable Subregisters

The Port A–E High Drive Enable Subregister, shown in Table 26, is accessed through the Port A–E Control Register by writing 04H to the Port A–E Address Register. Setting the bits in the Port A–E High Drive Enable subregisters to 1 configures the specified port pins for high-current output drive operation. The Port A–E High Drive Enable Subregister affects the pins directly and, as a result, alternate functions are also affected.

Table 26. Port A–E High Drive Enable Subregisters (PxHDE)

Bits	7	6	5	4	3	2	1	0
Field	PHDE7	PHDE6	PHDE5	PHDE4	PHDE3	PHDE2	PHDE1	PHDE0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	If 04H in Port A–E Address Register, accessible through the Port A–E Control Register							

Bit	Description
[7:0] PHDE	**Port High Drive Enabled** 0 = The Port pin is configured for standard output current drive. 1 = The Port pin is configured for high output current drive.

7.11.14. LED Drive Level Registers

Two LED Drive Level registers consist of the LED Drive Level High Bit Register (LEDLVLH[7:0]) and the LED Drive Level Low Bit Register (LEDLVLL[7:0]), as shown in Tables 34 and 35. Two control bits, LEDLVLH[x] and LEDLVLL[x], are used to select one of four programmable current drive levels for each associated Port C[x] pin. Each Port C pin is individually programmable.

Table 34. LED Drive Level High Bit Register (LEDLVLH)

Bits	7	6	5	4	3	2	1	0
Field	LEDLVLH							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F83H							

Table 35. LED Drive Level Low Bit Register (LEDLVLL)

Bits	7	6	5	4	3	2	1	0
Field	LEDLVLL							
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F84H							

Bit	Description
[7:0] LEDLVLH, LEDLVLL	**LED Drive Level High Bit** **LED Drive Level Low Bit** These bits are used to set the LED drive current. {LEDLVLH[x], LEDLVLL[x]}, in which x=Port C[0] to Port C[7]. Select one of the following four programmable current drive levels for each Port C pin. 00 = 3 mA 01 = 7 mA 10 = 13 mA 11 = 20 mA

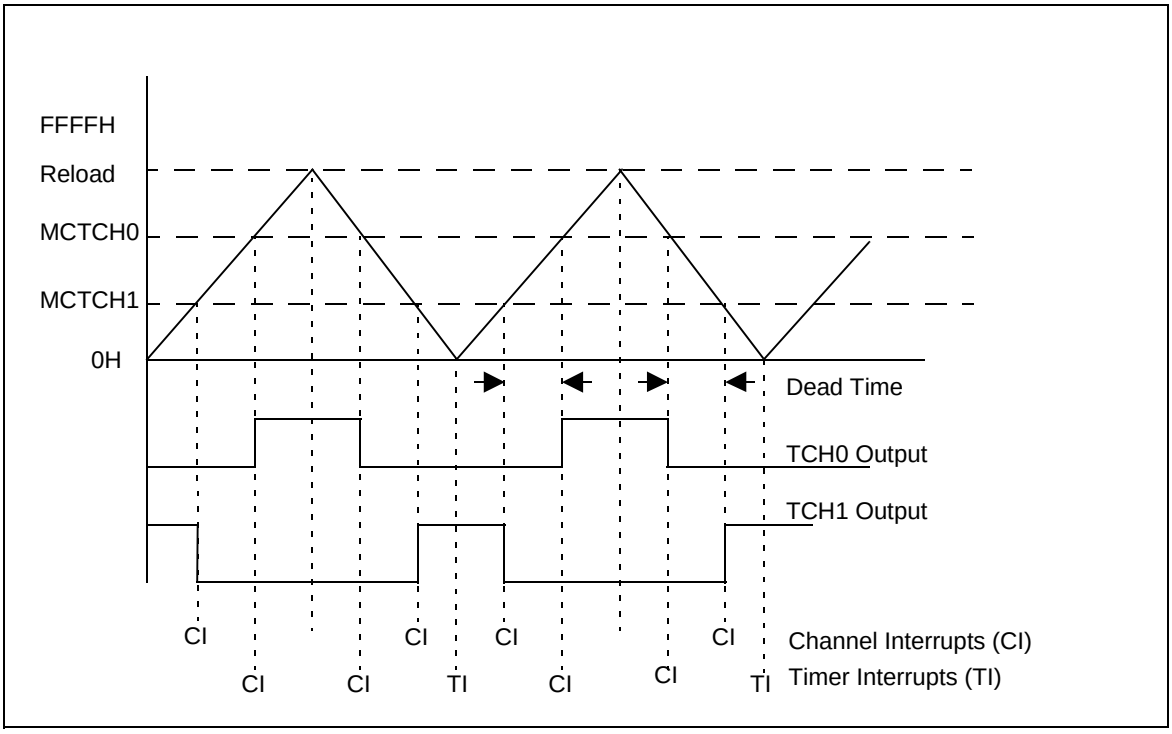

Figure 17. Count Up/Down Mode with PWM Channel Outputs and Deadband

10.6.2. Multiple Timer Intervals Generation

Figure 18 shows a timing diagram featuring two constant time intervals, T0 and T1. The timer is in Count Modulo Mode with reload = FFFFH. Channels 0 and 1 are set up for CONTINUOUS COMPARE operation. After every channel compare interrupt, the channel Capture/Compare registers are updated in the interrupt service routine by adding a constant equal to the time interval required. This operation requires that the Channel Update Enable (CHUE) bit must be set in channels 0 and 1 so that writes to the Capture/ Compare registers take affect immediately.

UART Control 0 Register must be initialized with TEN = 1, REN = 1 and all other bits = 0.

In addition to the LMST, LSLV and ABEN bits in the LIN Control Register, a LinState[1:0] field exists which defines the current state of the LIN logic. This field is initially set by software. In the LIN SLAVE Mode, the LinState field is updated by hardware as the slave moves through the Wait For Break, AutoBaud and Active states.

12.1.10.3. LIN MASTER Mode Operation

LIN MASTER Mode is selected by setting LMST = 1, LSLV = 0, ABEN = 0 and LinState[1:0] = 11B. If the LIN bus protocol indicates the bus is required go into the LIN SLEEP state, the LinState[1:0] bits must be set to 00B by software.

The break is the first part of the message frame transmitted by the master, consisting of at least 13 bit periods of logical zero on the LIN bus. During initialization of the LIN master, the duration (in bit times) of the break is written to the TxBreakLength field of the LIN Control Register. The transmission of the break is performed by setting the SBRK bit in the Control 0 Register. The LIN-UART starts the break after the SBRK bit is set and any character transmission currently underway has completed. The SBRK bit is deasserted by hardware until the break is completed.

If it is necessary to generate a break longer than 15 bit times, the SBRK bit can be used in normal UART mode where software times the duration of the break.

The Synch character is transmitted by writing a 55H to the Transmit Data Register (TDRE must = 1 before writing). The Synch character is not transmitted by the hardware until the break is complete.

The identifier character is transmitted by writing the appropriate value to the Transmit Data Register (TDRE must = 1 before writing).

If the master is sending the *response* portion of the message, these data and checksum characters are written to the Transmit Data Register when the TDRE bit asserts. If the Transmit Data Register is written after TDRE asserts, but before TXE asserts, the hardware inserts one or two stop bits between each character as determined by the stop bit in the Control 0 Register. Additional idle time occurs between characters, if TXE asserts before the next character is written.

If the selected slave is sending the response portion of the frame to the master, each receive byte will be signalled by the receive data interrupt (RDA bit will be set in the Status 0 Register). If the selected slave is sending the response to a different slave, the master can ignore the response characters by deasserting the REN bit in the Control 0 Register until the frame time slot is completed.

12.1.10.4. LIN SLEEP Mode

While the LIN bus is in the *sleep* state, the CPU can either be in low power STOP Mode, in HALT Mode, or in normal operational state. Any device on the LIN bus can issue a

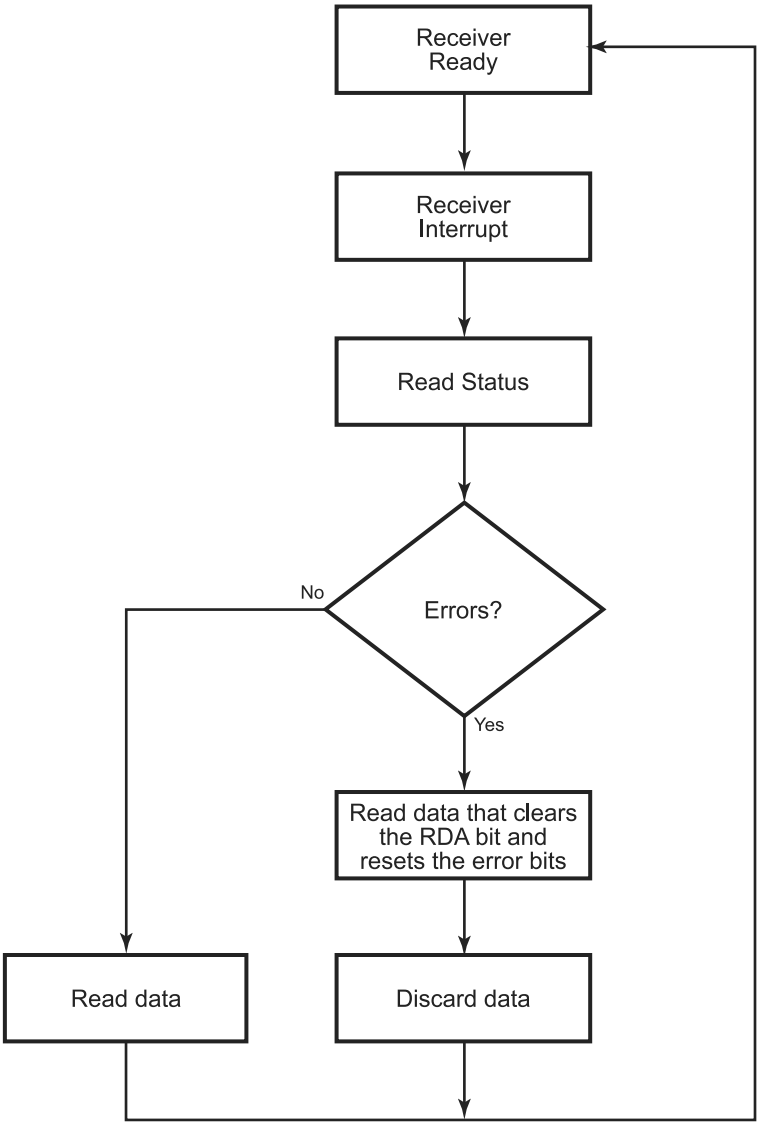

Figure 24. LIN-UART Receiver Interrupt Service Routine Flow

12.1.11.5. Baud Rate Generator Interrupts

If the BRGCTL bit of the Multiprocessor Control Register (LIN-UART Control 1 Register with MSEL = 000b) is set and the REN bit of the Control 0 Register is 0. The LIN-UART Receiver interrupt asserts when the LIN-UART Baud Rate Generator reloads. This action allows the Baud Rate Generator to function as an additional counter, if the LIN-UART receiver functionality is not employed. The transmitter can be enabled in this mode.

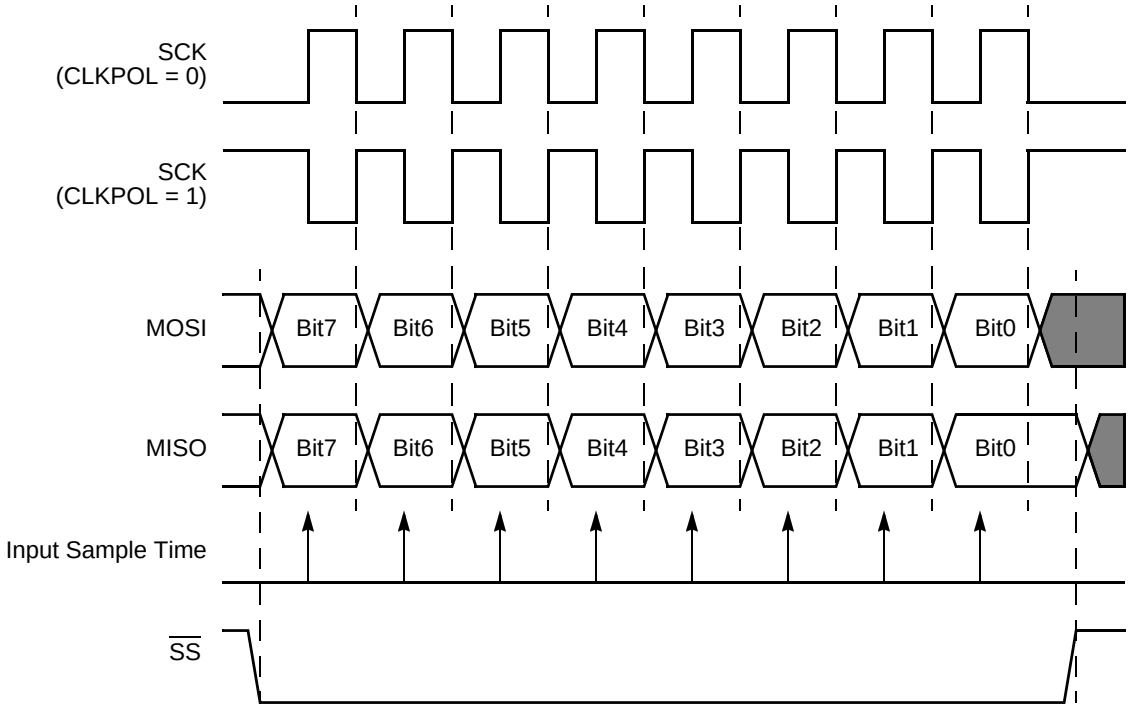

Figure 34. ESPI Timing when PHASE=0

16.3.2.2. Transfer Format When Phase Equals One

Figure 35 displays a timing diagram for an SPI type transfer in which PHASE is one. For SPI transfers the clock only toggles during the character transfer. Two waveforms are depicted for SCK, one for CLKPOL = 0 and another for CLKPOL = 1.

Table 116. ESPI Baud Rate High Byte Register (ESPIBRH)

Bits	7	6	5	4	3	2	1	0
Field	BRH							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F66H							

Bit	Description
[7:0] BRH	**ESPI Baud Rate High Byte** The most significant byte, BRG[15:8], of the ESPI Baud Rate Generator's reload value.

Table 117. ESPI Baud Rate Low Byte Register (ESPIBRL)

Bits	7	6	5	4	3	2	1	0
Field	BRL							
Reset	1	1	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/w
Address	F67H							

Bit	Description
[7:0] BRL	**ESPI Baud Rate Low Byte** The least significant byte, BRG[7:0], of the ESPI Baud Rate Generator's reload value.

17.2.3. Start and Stop Conditions

The Master generates the start and stop conditions to start or end a transaction. To start a transaction, the I^2C controller generates a start condition by pulling the SDA signal Low while SCL is High. To complete a transaction, the I^2C controller generates a stop condition by creating a Low-to-High transition of the SDA signal while the SCL signal is High. These start and stop events occur when the start and stop bits in the I^2C Control Register are written by software to begin or end a transaction. Any byte transfer currently under way including the Acknowledge phase finishes before the start or stop condition occurs.

17.2.4. Software Control of I^2C Transactions

The I^2C controller is configured via the I^2C Control and I^2C Mode registers. The MODE[1:0] field of the I^2C Mode Register allows the configuration of the I^2C controller for MASTER/SLAVE or SLAVE ONLY mode and configures the slave for 7-bit or 10-bit addressing recognition.

MASTER/SLAVE Mode can be used for:

- MASTER ONLY operation in a Single Master/One or More Slave I^2C system
- MASTER/SLAVE in a Multimaster/multislave I^2C system
- SLAVE ONLY operation in an I^2C system

In SLAVE ONLY mode, the start bit of the I^2C Control Register is ignored (software cannot initiate a master transaction by accident) and operation to SLAVE ONLY Mode is restricted thereby preventing accidental operation in MASTER Mode. The software controls I^2C transactions by enabling the I^2C controller interrupt in the interrupt controller or by polling the I^2C Status Register.

To use interrupts, the I^2C interrupt must be enabled in the interrupt controller and followed by executing an EI instruction. The TXI bit in the I^2C Control Register must be set to enable transmit interrupts. An I^2C interrupt service routine then checks the I^2C Status Register to determine the cause of the interrupt.

To control transactions by polling, the TDRE, RDRF, SAM, ARBLST, SPRS and NCKI interrupt bits in the I^2C Status Register should be polled. The TDRE bit asserts regardless of the state of the TXI bit.

17.2.5. Master Transactions

The following sections describe Master Read and Write transactions to both 7-bit and 10-bit slaves.

from the Shift Register after it is received from the I^2C bus. The I^2C Shift Register is not accessible in the Register File address space, but is used only to buffer incoming and outgoing data.

Writes by the software to the I2CDATA Register are blocked if a slave Write transaction is underway (the I^2C controller is in SLAVE Mode and data is being received).

Table 119. I^2C Data Register (I2CDATA = F50H)

Bits	7	6	5	4	3	2	1	0
Field	Data 7	Data 6	Data 5	Data 4	Data 3	Data 2	Data 1	Data 0
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F50H							

Bit Position	Value	Description
[7:0] DATA	—	I^2C Data Byte

Chapter 19. Temperature Sensor

The on-chip Temperature Sensor allows you to measure temperature on the die to an accuracy of roughly ±7°C over a range of –40°C to +105°C. Over a reduced range, the accuracy is significantly better. This block is a moderately accurate temperature sensor for low-power applications where high accuracy is not required. Uncalibrated accuracy is significantly worse, therefore the temperature sensor is not recommended for untrimmed use:

- On-chip temperature sensor
- ±7°C full-range accuracy for calibrated version
- ±1.5°C accuracy over the range of 20°C to 30°C
- Flash recalibration capability

19.1. Operation

The on-chip temperature sensor is a Proportional To Absolute Temperature (PTAT) topology which provides for zero-point calibration. A pair of Flash option bytes contain the calibration data. The temperature sensor can be disabled by a bit in the Power Control Register 0 (see page 44) to reduce power consumption.

The temperature sensor can be directly read by the ADC to determine the absolute value of its output. The temperature sensor output is also available as an input to the comparator for threshold type measurement determination. The accuracy of the sensor when used with the comparator is substantially less than when measured by the ADC. Maximum accuracy can be obtained by customer retrimming the sensor using an external reference and a high-precision external reference in the target application.

During normal operation, the die undergoes heating that will cause a mismatch between the ambient temperature and that measured by the sensor. For best results, the XP device should be placed into STOP Mode for sufficient time such that the die and ambient temperatures converge (this time will be dependent on the thermal design of the system). The temperature sensor should be measured immediately after recovery from STOP Mode.

The following two equations define the relationship between the ADC reading and the die temperature. In each equation, T is the temperature in degrees Celsius, and ADC is the 10-bit compensated ADC value.

Equation #1. If bit 2 of TEMPCALH calibration option byte is 0, then:

T = (25/128) * (ADC + {TEMPCALH_bit1, TEMPCALH_bit0, TEMPCALL}) – 77

Equation #2. If bit 2 of TEMPCALH calibration option byte is 1, then:

T = (25/128) * (ADC – {TEMPCALH_bit1, TEMPCALH_bit0, TEMPCALL}) –77

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.

! **Caution:** The byte at each Flash memory address cannot be programmed (any bits written to 0) more than twice before an erase cycle occurs.

20.2.5. Page Erase

The Flash memory can be erased one page (512 bytes) at a time. Page-erasing Flash memory sets all bytes in that page to the value FFH. The Flash Page Select register identifies the page to be erased. Only a page residing in an unprotected sector can be erased. With the Flash Controller unlocked, writing the value 95h to the Flash Control Register initiates the Page Erase operation on the active page. While the Flash Controller executes the Page Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. The eZ8 CPU resumes operation after the Page Erase operation completes. If the Page Erase operation is performed using the OCD, poll the Flash Status register to determine when the Page Erase operation is complete. When the Page Erase is complete, the Flash Controller returns to its locked state.

20.2.6. Mass Erase

Flash memory can also be mass-erased using the Flash Controller, but only by using the On-Chip Debugger. Mass-erasing Flash memory sets all bytes to the value FFH. With the Flash Controller unlocked, writing the value 63H to the Flash Control Register initiates the Mass Erase operation. While the Flash Controller executes the Mass Erase operation, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. Using the On-Chip Debugger, poll the Flash Status register to determine when the Mass Erase operation is complete. When the Mass Erase is complete, the Flash Controller returns to its locked state.

20.2.7. Flash Controller Bypass

The Flash Controller can be bypassed and the control signals for Flash memory are brought out to the GPIO pins. Bypassing the Flash Controller allows faster row programming algorithms by controlling these Flash programming signals directly.

Row programming is recommended for gang programming applications and large-volume customers who do not require in-circuit initial programming of Flash memory. Mass Erase and Page Erase operations are also supported when the Flash Controller is bypassed.

For more information about bypassing the Flash Controller, please contact Zilog Technical Support.

```
DBG ← Size[7:0]
DBG → 1-65536 data bytes
```

Read Program Memory CRC (0EH). The Read Program Memory CRC command computes and returns the CRC (cyclic redundancy check) of Program memory using the 16-bit CRC-CCITT polynomial ($x^{16} + x^{12} + x^5 + 1$). The CRC is preset to all 1s. The least-significant bit of the data is shifted through the polynomial first. The CRC is inverted when it is transmitted. If the device is not in DEBUG mode, this command returns FFFFH for the CRC value. Unlike most other OCD Read commands, there is a delay from issuing of the command until the OCD returns the data. The OCD reads Program memory, calculates the CRC value and returns the result. The delay is a function of the Program memory size and is approximately equal to the system clock period multiplied by the number of bytes in Program memory.

```
DBG ← 0EH
DBG → CRC[15:8]
DBG → CRC[7:0]
```

Step Instruction (10H). The Step Instruction command steps one assembly instruction at the current Program Counter (PC) location. If the device is not in DEBUG mode or the Read Protect Option bit is enabled, the OCD ignores this command.

```
DBG ← 10H
```

Stuff Instruction (11H). The Stuff Instruction command steps one assembly instruction and allows specification of the first byte of the instruction. The remaining 0-4 bytes of the instruction are read from Program memory. This command is useful for stepping over instructions where the first byte of the instruction has been overwritten by a breakpoint. If the device is not in DEBUG mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 11H
DBG ← opcode[7:0]
```

Execute Instruction (12H). The Execute Instruction command allows sending an entire instruction to be executed to the eZ8 CPU. This command can also step over breakpoints. The number of bytes to send for the instruction depends on the op code. If the device is not in DEBUG mode or the Read Protect option bit is enabled, the OCD ignores this command.

```
DBG ← 12H
DBG ← 1-5 byte opcode
```

Write Line Control Register (18H). The Write Line Control Register command writes the data that follows to the Line Control Register.

```
DBG ← 18H
DBG ← LCR[7:0]
```

Read Line Control Register (19H). The Read Line Control Register command returns the current value in the Line Control Register.

```
DBG ← 19H
DBG → LCR[7:0]
```

Table 186. eZ8 CPU Instruction Summary (Continued)

Assembly Mnemonic	Symbolic Operation	Address Mode dst	Address Mode src	Op Code(s) (Hex)	Flags C	Z	S	V	D	H	Fetch Cycles	Instr. Cycles
AND dst, src	dst ← dst AND src	r	r	52	–	*	*	0	–	–	2	3
		r	Ir	53							2	4
		R	R	54							3	3
		R	IR	55							3	4
		R	IM	56							3	3
		IR	IM	57							3	4
ANDX dst, src	dst ← dst AND src	ER	ER	58	–	*	*	0	–	–	4	3
		ER	IM	59							4	3
ATM	Block all interrupt and DMA requests during execution of the next 3 instructions			2F	–	–	–	–	–	–	1	2
BCLR bit, dst	dst[bit] ← 0	r		E2	–	*	*	0	–	–	2	2
BIT p, bit, dst	dst[bit] ← p	r		E2	–	*	*	0	–	–	2	2
BRK	Debugger Break			00	–	–	–	–	–	–	1	1
BSET bit, dst	dst[bit] ← 1	r		E2	–	*	*	0	–	–	2	2
BSWAP dst	dst[7:0] ← dst[0:7]	R		D5	X	*	*	0	–	–	2	2
BTJ p, bit, src, dst	if src[bit] = p PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
BTJNZ bit, src, dst	if src[bit] = 1 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
BTJZ bit, src, dst	if src[bit] = 0 PC ← PC + X		r	F6	–	–	–	–	–	–	3	3
			Ir	F7							3	4
CALL dst	SP ← SP -2 @SP ← PC PC ← dst	IRR		D4	–	–	–	–	–	–	2	6
		DA		D6							3	3
CCF	C ← ~C			EF	*	–	–	–	–	–-	1	2

Flags notation:
* = Value is a function of the result of the operation.
– = Unaffected.
X = Undefined.
0 = Reset to 0.
1 = Set to 1.