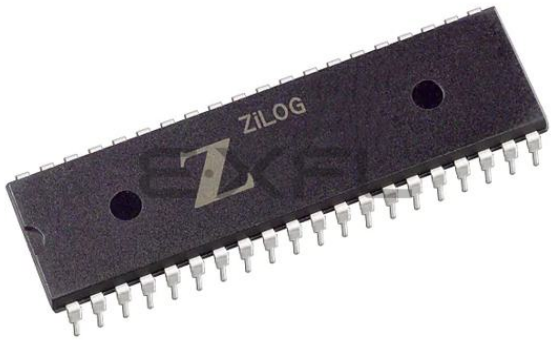

Welcome to E-XFL.COM

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	33
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	40-DIP (0.620", 15.75mm)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1680pm020sg

Table 88. Mode Status Fields 169
Table 89. LIN-UART Control 0 Register (U0CTL0 = F42H) 170
Table 90. Multiprocessor Control Register (U0CTL1 = F43H with MSEL = 000b) . 172
Table 91. Noise Filter Control Register (U0CTL1 = F43H with MSEL = 001b) . . . 174
Table 92. LIN Control Register (U0CTL1 = F43H with MSEL = 010b). 175
Table 93. LIN-UART Address Compare Register (U0ADDR = F45H) 177
Table 94. LIN-UART Baud Rate High Byte Register (U0BRH = F46H) 177
Table 95. LIN-UART Baud Rate Low Byte Register (U0BRL = F47H). 178
Table 96. LIN-UART Baud Rates, 20.0 MHz System Clock 179
Table 97. LIN-UART Baud Rates, 10.0 MHz System Clock 179
Table 98. LIN-UART Baud Rates, 5.5296 MHz System Clock 180
Table 99. LIN-UART Baud Rates, 3.579545 MHz System Clock 180
Table 100. LIN-UART Baud Rates, 1.8432 MHz System Clock 181
Table 101. ADC Control Register 0 (ADCCTL0) 189
Table 102. ADC Raw Data High Byte Register (ADCRD_H). 191
Table 103. ADC Data High Byte Register (ADCD_H) 191
Table 104. ADC Data Low Bits Register (ADCD_L) 192
Table 105. Sample Settling Time (ADCSST). 193
Table 106. Sample Time (ADCST) 194
Table 107. ADC Clock Prescale Register (ADCCP) 195
Table 108. ESPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation . . 201
Table 109. ESPI Data Register (ESPIDATA) 214
Table 110. ESPI Transmit Data Command and Receive Data Buffer Control Register (ESPITDCR) 214
Table 111. ESPI Control Register. 215
Table 112. ESPI Mode Register (ESPIMODE) 217
Table 113. ESPI Status Register (ESPISTAT) 219
Table 114. ESPI State Register (ESPISTATE). 220
Table 115. ESPISTATE Values 221
Table 116. ESPI Baud Rate High Byte Register (ESPIBRH) 222
Table 117. ESPI Baud Rate Low Byte Register (ESPIBRL) 222

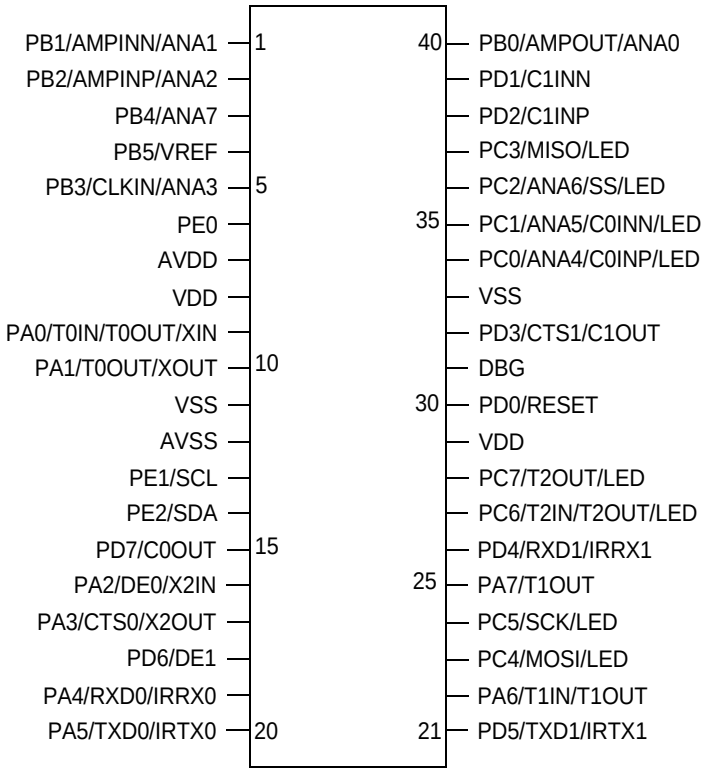

Figure 4. Z8F2480, Z8F1680 and Z8F0880 in 40-Pin Dual Inline Package (PDIP)

9.2.3.10. COMPARE Mode

In COMPARE Mode, the timer counts up to the 16-bit maximum Compare value stored in the Timer Reload High and Low Byte registers. The Timer counts timer clocks up to a 16-bit reload value. Upon reaching the Compare value, the timer generates an interrupt and counting continues (the timer value is not reset to 0001H). Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) on Compare.

If the Timer reaches FFFFH, the timer rolls over to 0000H and continue counting.

Observe the following steps to configure a timer for COMPARE Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for COMPARE Mode
   - Set the prescale valu.
   - Set the initial logic level (High or Low) for the Timer Output alternate function, if required
2. Write to the Timer Control 2 Register to choose the timer clock source.
3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value.
5. Write to the Timer Reload High and Low Byte registers to set the Compare value.
6. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. When using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
8. Write to the Timer Control 1 Register to enable the timer and initiate counting.

In COMPARE Mode, the timer clock always provides the timer input. The Compare time is calculated using the following equation:

9.2.3.11. GATED Mode

In GATED Mode, the timer counts only when the Timer Input signal is in its active state (asserted) as determined by the TPOL bit in the Timer Control 1 Register. When the Timer Input signal is asserted, counting begins. A Timer Interrupt is generated when the Timer Input signal is deasserted or a timer reload occurs. To determine if a Timer Input signal

9.2.3.12. CAPTURE/COMPARE Mode

In CAPTURE/COMPARE Mode, the timer begins counting on the first external Timer Input transition. The appropriate transition (rising edge or falling edge) is set by the TPOL bit in the Timer Control 1 Register. The Timer counts timer clocks up to the 16-bit reload value.

Every subsequent appropriate transition (after the first) of the Timer Input signal captures the current count value. The Capture value is written to the Timer PWM0 High and Low Byte registers. When the Capture event occurs, an interrupt is generated, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. The INPCAP bit in Timer Control 0 Register is set to indicate the timer interrupt is due to an input capture event.

If no Capture event occurs, the timer counts up to the 16-bit Compare value stored in the Timer Reload High and Low Byte registers. Upon reaching the Compare value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. The INPCAP bit in Timer Control 0 Register is cleared to indicate the timer interrupt is not due to an input capture event.

Observe the following steps to configure a timer for CAPTURE/COMPARE Mode and initiate the count:

1. Write to the Timer Control 1 Register to:
   - Disable the timer
   - Configure the timer for CAPTURE/COMPARE Mode
   - Set the prescale value
   - Set the Capture edge (rising or falling) for the Timer Input
2. Write to the Timer High and Low Byte registers to set the starting count value (typically 0001H).
3. Write to the Timer Control 2 Register to choose the timer clock source.
4. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
5. Write to the Timer Reload High and Low Byte registers to set the Compare value.
6. If required, enable the timer interrupt and set the timer-interrupt priority by writing to the relevant interrupt registers. By default, the timer interrupt will be generated for both input capture and reload events. If required, configure the timer interrupt to be generated only at the input Capture event or the Reload event by setting TICONFIG field of the Timer Control 0 Register.
7. Configure the associated GPIO port pin for the Timer Input alternate function.
8. Write to the Timer Control 1 Register to enable the timer.

10.7.2. Multi-Channel Timer High and Low Byte Registers

The High and Low Byte (MCTH and MCTL) registers, shown in Table 70, contain the current 16-bit Multi-Channel Timer count value.

Zilog does not recommend writing to the Multi-Channel Timer High and Low Byte registers while the Multi-Channel Timer is enabled. If either or both of the Multi-Channel Timer High or Low Byte registers are written during counting, the 8-bit written value is placed in the counter (High and/or Low byte) at the next system clock edge. The counter continues counting from the new value.

Table 70. Multi-Channel Timer High and Low Byte Registers (MCTH, MCTL)

Bit	7	6	5	4	3	2	1	0
Field	MCTH							
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	FA0H							

Bit	7	6	5	4	3	2	1	0
Field	MCTL							
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	FA1H							

Bit	Description
[7:0] MCTH, MCTL	**Multi-Channel Timer High and Low Byte** These bytes contain the current 16-bit Multi-Channel Timer count value, {MCTH[7:0], MCTL[7:0]}.

When the Multi-Channel Timer is enabled, a read from MCTH causes the value in MCTL to be stored in a temporary holding register. A read from MCTL returns this temporary register when the Multi-Channel Timer is enabled. When the Multi-Channel Timer is disabled, reads from MCTL read the register directory. The Multi-Channel Timer High and Low Byte registers are not reset when TEN = 0.

10.7.3. Multi-Channel Timer Reload High and Low Byte Registers

The Multi-Channel Timer Reload High and Low Byte (MCTRH and MCTRL) registerss, shown in Table 71, store a 16-bit reload value, {MCTRH[7:0], MCTRL[7:0]}. When TEN = 0, writes to this address update the register on the next clock cycle. When TEN = 1,

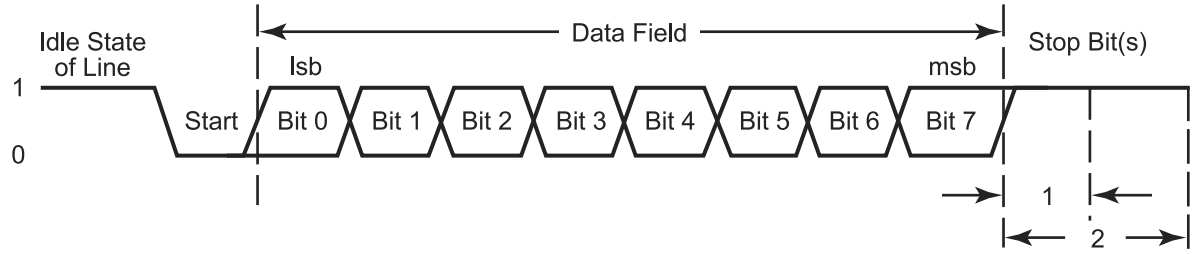

Figure 20. LIN-UART Asynchronous Data Format without Parity

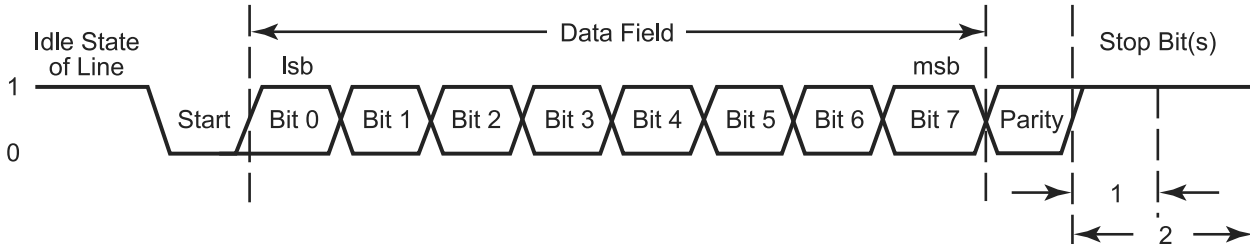

Figure 21. LIN-UART Asynchronous Data Format with Parity

12.1.2. Transmitting Data using the Polled Method

Observe the following steps to transmit data using the polled-operating method:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate-function operation.
3. If MULTIPROCESSOR Mode is appropriate, write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-bit) Mode functions.
4. Set the MULTIPROCESSOR Mode Select bit (MPEN) to enable MULTIPROCESSOR Mode.
5. Write to the LIN-UART Control 0 Register to:
   a. Set the Transmit Enable bit (TEN) to enable the LIN-UART for data transmission.
   b. If parity is appropriate and MULTIPROCESSOR Mode is not enabled, set the parity enable bit (PEN) and select either even-or-odd parity (PSEL).
   c. Set or clear the CTSE bit to enable or disable control from the remote receiver using the $\overline{\text{CTS}}$ pin.

Wake-up message if it requires the master to initiate a LIN message frame. Following the Wake-up message, the master wakes up and initiates a new message. A Wake-up message is accomplished by pulling the bus Low for at least 250 µs but less than 5 ms. Transmitting a 00H character is one way to transmit the Wake-up message.

If the CPU is in STOP Mode, the LIN-UART is not active and the Wake-up message must be detected by a GPIO edge detect Stop Mode Recovery. The duration of the Stop Mode Recovery sequence can preclude making an accurate measurement of the Wake-up message duration.

If the CPU is in HALT or OPERATIONAL mode, the LIN-UART (if enabled) times the duration of the Wake-up and provides an interrupt following the end of the break sequence if the duration is ≥ 3 bit times. The total duration of the Wake-up message in bit times can be obtained by reading the RxBreakLength field in the Mode Select and Status register. After a Wake-up message has been detected, the LIN-UART can be placed (by software) either into LIN Master or LIN Slave Wait for Break states as appropriate. If the break duration exceeds 15 bit times, the RxBreakLength field contains the value Fh. If the LIN-UART is disabled, Wake-up message is detected via a port pin interrupt and timed by software. If the device is in STOP Mode, the High to Low transition on the port pin will bring the device out of STOP Mode.

The LIN Sleep state is selected by software setting LinState[1:0] = 00. The decision to move from an active state to sleep state is based on the LIN messages as interpreted by software.

12.1.10.5. LIN Slave Operation

LIN SLAVE Mode is selected by setting LMST = 0, LSLV = 1, ABEN = 1 or 0 and LinState[1:0] = 01b (Wait for Break state). The LIN slave detects the start of a new message by the break which appears to the Slave as a break of at least 11 bit times in duration. The LIN-UART detects the break and generates an interrupt to the CPU. The duration of the break is observable in the RxBreakLength field of the Mode Select and Status register. A break of less than 11 bit times in duration does not generate a break interrupt when the LIN-UART is in Wait for Break state. If the break duration exceeds 15 bit times, the RxBreakLength field contains the value Fh.

Following the break, the LIN-UART hardware automatically transits to the *Autobaud* state, where it autobauds by timing the duration of the first 8 bit times of the Synch character as defined in the LIN standard. The duration of the autobaud period is measured by the BRG Counter which will update every 8th system clock cycle between the start bit and the beginning of bit 7 of the autobaud sequence. At the end of the autobaud period, the duration measured by the BRG counter (auto baud period divided by 8) is automatically transferred to the Baud Reload High and Low registers if the ABEN bit of the LIN control register is set. If the BRG Counter overflows before reaching the start of bit 7 in the autobaud sequence the Autobaud Overrun Error interrupt occurs, the OE bit in the Status 0 Register is set and the Baud Reload registers are not updated. To autobaud within 2% of the master's baud rate, the slave system clock must be a minimum of 100 times the baud

the V_{REF} pin. When RBUF is disabled, the ADC must have the reference voltage supplied externally through the V_{REF} pin. RBUF is controlled by the REFEN bit in the ADC Control Register.

14.2.4. Internal Voltage Reference Generator

The Internal Voltage Reference Generator provides the voltage, VR2, for the RBUF. VR2 is 1.6V.

14.2.5. Calibration and Compensation

You can calibrate and store the values into Flash, or the user code can perform a manual offset calibration. There is no provision for manual gain calibration.

14.3. ADC Control Register Definitions

The registers that control analog-to-digital conversion functions are defined in this section.

14.3.1. ADC Control Register 0

The ADC Control Register 0, shown in Table 101, initiates the A/D conversion and provides ADC status information.

Table 101. ADC Control Register 0 (ADCCTL0)

Bits	7	6	5	4	3	2	1	0
Field	START	INTREF_SEL	REFEN	ADCEN	ANAIN[3:0]			
Reset	0	0	0	0	0	0	0	0
R/W	R/W1	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F70h							

Bit Position	Value (H)	Description
[7] START	0	**ADC Start/Busy** Writing a 0 has no effect. Reading a 0 indicates the ADC is available to begin a conversion.
	1	Writing a 1 starts a conversion. Reading a 1 indicates that a conversion is currently in progress.
[6] INTREF_SEL	0	Select 1.6 V as internal reference.
	1	Select AVDD as internal reference.

Chapter 15. Low-Power Operational Amplifier

The low-power operational amplifier is a standard operational amplifier designed for current measurements. Each of the three ports of the amplifier is accessible from the package pins. The inverting input is commonly used to connect to the current source. The output node connects an external feedback network to the inverting input. The non-inverting output is required to apply a nonzero bias point. In a standard single-supply system, this bias point must be substantially above ground to measure positive input currents. The noninverting input can also be used for offset correction.

Note: This amplifier is a voltage gain operational amplifier. Its transimpedance nature is determined by the feedback network.

The low-power operational amplifier contains only one pin configuration; ANA0 is the output/feedback node, ANA1 is the inverting input and ANA2 is the noninverting input.

To use the low-power operational amplifier, it must be enabled in the Power Control Register Definitions, discussed on page 44. The default state of the low-power operational amplifier is OFF. To use the low-power operational amplifier, the TRAM bit must be cleared, turning it ON (see Power Control Register 0 on page 44). When making normal ADC (i.e., not transimpedance) measurements on ANA0, the TRAM bit must be OFF. Turning the TRAM bit ON interferes with normal ADC measurements. Finally, this bit enables the amplifier even in STOP Mode. If the amplifier is not required in STOP Mode, disable it. Failing to perform this results in STOP Mode currents greater than specified.

As with other ADC measurements, any pins used for analog purposes must be configured as in the GPIO registers (see the Port A–E Alternate Function Subregisters on page 61).

Standard transimpedance measurements are made on ANA0 as selected by the ANAIN[3:0] bits of the ADC Control Register 0, discussed on page 189. It is also possible to make single-ended measurements on ANA1 and ANA2 when the amplifier is enabled which is often useful for determining offset conditions.

16.3.1. Throughput

In MASTER Mode, the maximum SCK rate supported is one-half the system clock frequency. This rate is achieved by programming the value 0001H into the Baud Rate High/Low register pair. Though each character will be transferred at this rate it is unlikely that software interrupt routines could keep up with this rate. In SPI mode the transfer will automatically pause between characters until the current receive character is read and the next transmit data value is written.

In SLAVE Mode, the transfer rate is controlled by the Master. As long as the TDRE and RDRNE interrupt are serviced before the next character transfer completes, the Slave will keep up with the Master. In SLAVE Mode the baud rate must be restricted to a maximum of one-eighth of the system clock frequency to allow for synchronization of the SCK input to the internal system clock.

16.3.2. ESPI Clock Phase and Polarity Control

The ESPI supports four combinations of serial clock phase and polarity using two bits in the ESPI Control Register. The clock polarity bit, CLKPOL, selects an active High or active Low clock and has no effect on the transfer format. Table 108 lists the ESPI Clock Phase and Polarity Operation parameters. The clock phase bit, PHASE, selects one of two fundamentally different transfer formats. The data is output a half-cycle before the receive clock edge which provides a half cycle of setup and hold time.

Table 108. ESPI Clock Phase (PHASE) and Clock Polarity (CLKPOL) Operation

PHASE	CLKPOL	SCK Transmit Edge	SCK Receive Edge	SCK Idle State
0	0	Falling	Rising	Low
0	1	Rising	Falling	High
1	0	Rising	Falling	Low
1	1	Falling	Rising	High

16.3.2.1. Transfer Format when Phase Equals Zero

Figure 34 displays the timing diagram for an SPI type transfer, in which PHASE=0. For SPI transfers the clock only toggles during the character transfer. The two SCK waveforms show polarity with CLKPOL = 0 and CLKPOL = 1. The diagram can be interpreted as either a Master or Slave timing diagram because the SCK Master-In/Slave-Out (MISO) and Master-Out/Slave-In (MOSI) pins are directly connected between the Master and the Slave.

Chapter 17. I²C Master/Slave Controller

The I²C Master/Slave Controller ensures that the Z8 Encore! XP F1680 Series devices are bus-compatible with the I²C protocol. The I²C bus consists of the serial data signal (SDA) and a serial clock signal (SCL) bidirectional lines. The features of I²C controller include:

- Operates in MASTER/SLAVE or SLAVE ONLY modes
- Supports arbitration in a multimaster environment (MASTER/SLAVE Mode)
- Supports data rates up to 400 Kbps
- 7-bit or 10-bit slave address recognition (interrupt only on address match)
- Optional general call address recognition
- Optional digital filter on receive SDA, SCL lines
- Optional interactive receive mode allows software interpretation of each received address and/or data byte before acknowledging
- Unrestricted number of data bytes per transfer
- Baud Rate Generator can be used as a general-purpose timer with an interrupt, if the I²C controller is disabled

17.1. Architecture

Figure 42 displays the architecture of the I²C controller.

14. The software responds by writing the data to be written out to the I^2C Control Register.
15. The I^2C controller shifts out the remainder of the second byte of the slave address (or ensuring data bytes, if looping) via the SDA signal.
16. The I^2C slave sends an Acknowledge by pulling the SDA signal Low during the next High period of SCL. The I^2C controller sets the ACK bit in the I^2C Status Register. If the slave does not acknowledge, see the second paragraph of Step 11.
17. The I^2C controller shifts the data out by the SDA signal. After the first bit is sent, the transmit interrupt asserts.
18. If more bytes remain to be sent, return to Step 14.
19. The software responds by asserting the stop bit of the I^2C Control Register.
20. The I^2C controller completes transmission of the data on the SDA signal.
21. The I^2C controller sends a stop condition to the I^2C bus.

Note: If the slave responds with a Not Acknowledge during the transfer, the I^2C controller asserts the NCKI bit, sets the ACKV bit, clears the ACK bit in the I^2C State Register and halts. The software terminates the transaction by setting either the stop bit (end transaction) or the start bit (end this transaction, start a new one). The Transmit Data Register is flushed automatically.

17.2.5.6. Master Read Transaction with a 7-Bit Address

Figure 45 displays the data transfer format for a Read operation to a 7-bit addressed slave.

S	Slave Address	R = 1	A	Data	A	Data	A	P/S

Figure 45. Data Transfer Format—Master Read Transaction with a 7-Bit Address

Observe the following steps for a Master Read operation to a 7-bit addressed slave:

1. The software initializes the MODE field in the I^2C Mode Register for MASTER/SLAVE Mode with 7- or 10-bit addressing (the I^2C bus protocol allows the mixing of slave address types). The MODE field selects the address width for this mode when addressed as a slave (but not for the remote slave). The software asserts the IEN bit in the I^2C Control Register.
2. The software writes the I^2C Data Register with a 7-bit slave address, plus the Read bit (which is set to 1).
3. The software asserts the start bit of the I^2C Control Register.

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

2. The Master initiates a transfer by sending the address byte. The SLAVE Mode I^2C controller finds an address match and detects that the R/$\overline{W}$ bit = 1 (read by the master from the slave). The I^2C 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^2C 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^2C 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^2C controller holds SCL Low until the Data Register has been written. When a Not Acknowledge instruction is received by the slave, the I^2C 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^2C 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^2C Master/Slave Controller operating as a slave in 10-bit addressing mode, transmitting data to the bus master.

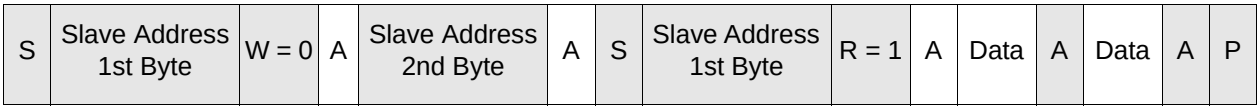

Figure 50. Data Transfer Format—Slave Transmit Transaction with 10-Bit Address

1. The software configures the controller for operation as a slave in 10-bit addressing mode.
   a. Initialize the MODE field in the I^2C 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^2C MODE Register.
   d. Set IEN = 1 and NAK = 0 in the I^2C Control Register.
2. The Master initiates a transfer by sending the first address byte. The SLAVE Mode I^2C 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^2C controller acknowledges indicating it is available to accept the transaction.
3. The Master sends the second address byte. The SLAVE Mode I^2C 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^2C controller acknowledges on the I^2C 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^2C 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^2C 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^2C 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^2C controller has data available to transmit, the SCL is released and the master proceeds to shift the first data byte.

```
nop      ; wait for output to settle
ldx IRQ0,#0 ; clear any spurious interrupts pending
ei
```

18.2. Comparator Control Register Definitions

This section defines the features of the following Comparator Control registers.

Comparator 0 Control Register: see page 257

Comparator 1 Control Register: see page 258

18.2.1. Comparator 0 Control Register

The Comparator 0 Control Register (CMP0), shown in Table 130, configures the Comparator 0 inputs and sets the value of the internal voltage reference.

Table 130. Comparator 0 Control Register (CMP0)

Bits	7	6	5	4	3	2	1	0
Field	INPSEL	INNSEL	REFLVL				TIMTRG	
Reset	0	0	0	1	0	1	0	0
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Address	F90H							

Bit	Description
[7] INPSEL	**Signal Select for Positive Input** 0 = GPIO pin used as positive comparator 0 input. 1 = Temperature sensor used as positive comparator 0 input.
[6] INNSEL	**Signal Select for Negative Input** 0 = Internal reference disabled, GPIO pin used as negative comparator 0 input. 1 = Internal reference enabled as negative comparator 0 input.

Read Baud Reload Register (1BH). The Read Baud Reload Register command returns the current value in the Baud Reload Register.

```
DBG ← 1BH
DBG → BAUD[15:8]
DBG → BAUD[7:0]
```

Write Test Mode Register (F0H). The Write Test Mode Register command writes the data that follows to the Test Mode register.

```
DBG ← F0H
DBG ← TESTMODE[7:0]
```

Read Test Mode Register (F1H). The Read Test Mode Register command returns the current value in the Test Mode register.

```
DBG ← F1H
DBG → TESTMODE[7:0]
```

Write Option Bit Registers (F2H). The Write Option Bit Registers command is used to write to the option bit configuration registers. The option bit configuration registers store the device configuration and are loaded from Flash every time the Z8 Encore! XP F1680 Series is reset. The registers may be temporarily written using this OCD command to test peripherals without having to program the Flash Information Area and resetting the Z8 Encore! XP F1680 Series. The ZilogUserSel bit selects between Zilog option bits (1) and user option bits (0).

```
DBG ← F2H
DBG ← {ZilogUserSel,1'b0,OptAddr[4:0]}
DBG ← OptData[7:0]
```

Read Option Bit Registers (F3H). The Read Option Bit Registers command is used to read the option bit registers that store the device configuration that is read out of flash when the Z8 Encore! XP F1680 Series is reset. The ZilogUserSel bit selects between reading Zilog option bits (1) or user option bits (0).

```
DBG ← F1H
DBG ← {ZilogUserSel,1'b0,OptAddr[4:0]}
DBG → OptData[7:0]
```

23.4. On-Chip Debugger Control Register Definitions

This section defines the features of the following On-Chip Debugger Control registers.

OCD Control Register: see page 310

OCD Status Register: see page 312

Line Control Register: see page 313

Baud Reload Register: see page 314

27.5. eZ8 CPU Instruction Summary

Table 186 summarizes the eZ8 CPU instructions. The table identifies the addressing modes employed by the instruction, the effect upon the Flags register, the number of CPU clock cycles required for the instruction fetch and the number of CPU clock cycles required for the instruction execution.

Table 186. eZ8 CPU Instruction Summary

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
ADC dst, src	dst ← dst + src + C	r	r	12	*	*	*	*	0	*	2	3
		r	Ir	13							2	4
		R	R	14							3	3
		R	IR	15							3	4
		R	IM	16							3	3
		IR	IM	17							3	4
ADCX dst, src	dst ← dst + src + C	ER	ER	18	*	*	*	*	0	*	4	3
		ER	IM	19							4	3
ADD dst, src	dst ← dst + src	r	r	02	*	*	*	*	0	*	2	3
		r	Ir	03							2	4
		R	R	04							3	3
		R	IR	05							3	4
		R	IM	06							3	3
		IR	IM	07							3	4
ADDX dst, src	dst ← dst + src	ER	ER	08	*	*	*	*	0	*	4	3
		ER	IM	09							4	3

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

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
LDX dst, src	dst ← src	r	ER	84	–	–	–	–	–	–	3	2
		Ir	ER	85							3	3
		R	IRR	86							3	4
		IR	IRR	87							3	5
		r	X(rr)	88							3	4
		X(rr)	r	89							3	4
		ER	r	94							3	2
		ER	Ir	95							3	3
		IRR	R	96							3	4
		IRR	IR	97							3	5
		ER	ER	E8							4	2
		ER	IM	E9							4	2
LEA dst, X(src)	dst ← src + X	r	X(r)	98	–	–	–	–	–	–	3	3
		rr	X(rr)	99							3	5
MULT dst	dst[15:0] ← dst[15:8] * dst[7:0]	RR		F4	–	–	–	–	–	–	2	8
NOP	No operation			0F	–	–	–	–	–	–	1	2
OR dst, src	dst ← dst OR src	r	r	42	–	*	*	0	–	–	2	3
		r	Ir	43							2	4
		R	R	44							3	3
		R	IR	45							3	4
		R	IM	46							3	3
		IR	IM	47							3	4
ORX dst, src	dst ← dst OR src	ER	ER	48	–	*	*	0	–	–	4	3
		ER	IM	49							4	3
POP dst	dst ← @SP SP ← SP + 1	R		50	–	–	–	–	–	–	2	2
		IR		51							2	3

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

Table 196. Analog-to-Digital Converter Electrical Characteristics and Timing

		T_A = 0°C to +70°C T_A = –40°C to +105°C				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
N	Resolution	–	10	–	Bit	
INL	Integral Nonlinearity	–5		5	LSB	
DNL	Differential Nonlinearity	–1		4	LSB	
	Gain Error		15		LSB	
	Offset Error	–15		15	LSB	PDIP package
		–9		9	LSB	Other packages
I_{DD}ADC	ADC Active Current	–	–	2.5	mA	
I_{DDQ}ADC	ADC Quiescent Current	–	5	–	nA	
V_{INT_REF}	Internal Reference Voltage	–	1.6	–	V	REFEN=1, INTREF_SEL=0. See Table 101 on page 189.
		–	AVDD	–	V	REFEN=1, INTREF_SEL=1. See Table 101 on page 189.
V_{EXT_REF}	External Reference Voltage	1.6	–	90% AV_{DD}	V	REFEN=0. See Table 101 on page 189.
V_{INANA}	Analog Input Range	0	–	1.6	V	Internal reference = 1.6V
		0	–	90% AV_{DD}	V	External reference or use AVDD as internal reference
C_{IN}	Analog Input Load	–	–	5	pF	
T_S	Sample Time	1.8	–	–	µs	
T_H	Hold Time	0.5	–	–	µs	
T_{CONV}	Conversion Time	–	13	–	clock cycles	
GBW_{IN}	Input Bandwidth	–	200	–	kHz	
T_{WAKE}	Wake-up Time	–	–	10	µs	External reference
		–	–	10	µs	Internal reference
f_{ADC_CLK}	Maximum Frequency of adc_clk	–	–	5	MHz	V_{DD} = 2.7V to 3.6V
		–	–	2.5	MHz	V_{DD} = 1.8V to 2.7V

Table 203. Low Power 32 kHz Secondary Oscillator Characteristics

		T_A = 0°C to +70°C T_A = –40°C to +105°C							
		V_{DD} = 2.7 to 3.6 V			V_{DD} = 1.8 to 2.7 V				
Symbol	**Parameter**	**Min**	**Typ**	**Max**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
$I_{DD}XTAL2$	32 kHz Secondary Oscillator Active Current	–	–	20	–	–	10	µA	
$I_{DDQ}XTAL2$	32 kHz Secondary Oscillator Quiescent Current	–	5	–	–	5	–	nA	
S_{CLK}	Clk_out State in Crystal Disable	1	1	1	1	1	1		
F_{XTAL2}	External Crystal Oscillator Frequency	–	32.768	–	–	32.768	–	kHz	
T_{SET}	Startup Time After Enable	–	400	1000	–	400	1000	mS	
	Clk_out Duty Cycle	40	50	60	40	50	60	%	
	Clk_out Jitter	–	1	–	–	1	–	%	