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	Obsolete
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	46
Program Memory Size	24KB (24K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	64-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f2402ar020ec

Use of All Uppercase Letters

The use of all uppercase letters designates the names of states and commands.

- Example 1: The bus is considered BUSY after the Start condition.
- Example 2: A START command triggers the processing of the initialization sequence.

Bit Numbering

Bits are numbered from *0* to *n–1* where *n* indicates the total number of bits. For example, the 8 bits of a register are numbered from 0 to 7.

Safeguards

It is important that all users understand the following safety terms, which are defined here.

Caution: Indicates a procedure or file may become corrupted if the user does not follow directions.

Trademarks

ZiLOG, eZ8, Z8 Encore!, and Z8 are trademarks of ZiLOG, Inc. in the U.S.A. and other countries. All other trademarks are the property of their respective corporations.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page #
F62	SPI Status	SPISTAT	01	108
F63	SPI Mode	SPIMODE	00	109
F64-F65	Reserved	—	XX	
F66	SPI Baud Rate High Byte	SPIBRH	FF	110
F67	SPI Baud Rate Low Byte	SPIBRL	FF	110
F68-F69	Reserved	—	XX	
Analog-to-Digital Converter (ADC)				
F70	ADC Control	ADCCTL	20	135
F71	Reserved	—	XX	
F72	ADC Data High Byte	ADCD_H	XX	137
F73	ADC Data Low Bits	ADCD_L	XX	137
F74-FAF	Reserved	—	XX	
DMA 0				
FB0	DMA0 Control	DMA0CTL	00	124
FB1	DMA0 I/O Address	DMA0IO	XX	125
FB2	DMA0 End/Start Address High Nibble	DMA0H	XX	126
FB3	DMA0 Start Address Low Byte	DMA0START	XX	127
FB4	DMA0 End Address Low Byte	DMA0END	XX	128
DMA 1				
FB8	DMA1 Control	DMA1CTL	00	124
FB9	DMA1 I/O Address	DMA1IO	XX	125
FBA	DMA1 End/Start Address High Nibble	DMA1H	XX	126
FBB	DMA1 Start Address Low Byte	DMA1START	XX	127
FBC	DMA1 End Address Low Byte	DMA1END	XX	128
DMA ADC				
FBD	DMA_ADC Address	DMAA_ADDR	XX	128
FBE	DMA_ADC Control	DMAACTL	00	130
FBF	DMA_ADC Status	DMAASTAT	00	131
Interrupt Controller				
FC0	Interrupt Request 0	IRQ0	00	48
FC1	IRQ0 Enable High Bit	IRQ0ENH	00	51
FC2	IRQ0 Enable Low Bit	IRQ0ENL	00	51
FC3	Interrupt Request 1	IRQ1	00	49
FC4	IRQ1 Enable High Bit	IRQ1ENH	00	52
FC5	IRQ1 Enable Low Bit	IRQ1ENL	00	52
FC6	Interrupt Request 2	IRQ2	00	50
FC7	IRQ2 Enable High Bit	IRQ2ENH	00	53
FC8	IRQ2 Enable Low Bit	IRQ2ENL	00	53
FC9-FCC	Reserved	—	XX	
FCD	Interrupt Edge Select	IRQES	00	54

XX=Undefined

Table 11. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	Alternate Function Description
Port D	PD0	T3IN	Timer 3 In (not available in 40- and 44-pin packages)
	PD1	T3OUT	Timer 3 Out (not available in 40- and 44-pin packages)
	PD2	N/A	No alternate function
	PD3	N/A	No alternate function
	PD4	RXD1 / IRRX1	UART 1 / IrDA 1 Receive Data
	PD5	TXD1 / IRTX1	UART 1 / IrDA 1 Transmit Data
	PD6	$\overline{\text{CTS1}}$	UART 1 Clear to Send
	PD7	RCOUT	Watch-Dog Timer RC Oscillator Output
Port E	PE[7:0]	N/A	No alternate functions
Port F	PF[7:0]	N/A	No alternate functions
Port G	PG[7:0]	N/A	No alternate functions
Port H	PH0	ANA8	ADC Analog Input 8
	PH1	ANA9	ADC Analog Input 9
	PH2	ANA10	ADC Analog Input 10
	PH3	ANA11	ADC Analog Input 11

GPIO Interrupts

Many of the GPIO port pins can be used as interrupt sources. Some port pins may be configured to generate an interrupt request on either the rising edge or falling edge of the pin input signal. Other port pin interrupts generate an interrupt when any edge occurs (both rising and falling). Refer to the **Interrupt Controller** chapter for more information on interrupts using the GPIO pins.

GPIO Control Register Definitions

Four registers for each Port provide access to GPIO control, input data, and output data. Table 12 lists these Port registers. Use the Port A-H Address and Control registers together to provide access to sub-registers for Port configuration and control.

PADDR[7:0]—Port Address
The Port Address selects one of the sub-registers accessible through the Port Control register.

PADDR[7:0]	Port Control sub-register accessible using the Port A-H Control Registers
00H	No function. Provides some protection against accidental Port reconfiguration.
01H	Data Direction
02H	Alternate Function
03H	Output Control (Open-Drain)
04H	High Drive Enable
05H	Stop Mode Recovery Source Enable.
06H-FFH	No function.

Port A-H Control Registers

The Port A-H Control registers set the GPIO port operation. The value in the corresponding Port A-H Address register determines the control sub-registers accessible using the Port A-H Control register (Table 14).

Table 14. Port A-H Control Registers (P*x*CTL)

BITS	7	6	5	4	3	2	1	0
FIELD	PCTL							
RESET	00H							
R/W	R/W							
ADDR	FD1H, FD5H, FD9H, FDDH, FE1H, FE5H, FE9H, FEDH							

PCTL[7:0]—Port Control
The Port Control register provides access to all sub-registers that configure the GPIO Port operation.

AF[7:0]—Port Alternate Function enabled
0 = The port pin is in normal mode and the `DDx` bit in the Port A-H Data Direction sub-register determines the direction of the pin.
1 = The alternate function is selected. Port pin operation is controlled by the alternate function.

Port A-H Output Control Sub-Registers

The Port A-H Output Control sub-register (Table 17) is accessed through the Port A-H Control register by writing `03H` to the Port A-H Address register. Setting the bits in the Port A-H Output Control sub-registers to 1 configures the specified port pins for open-drain operation. These sub-registers affect the pins directly and, as a result, alternate functions are also affected.

Table 17. Port A-H Output Control Sub-Registers

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
ADDR	If 03H in Port A-H Address Register, accessible via Port A-H Control Register							

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

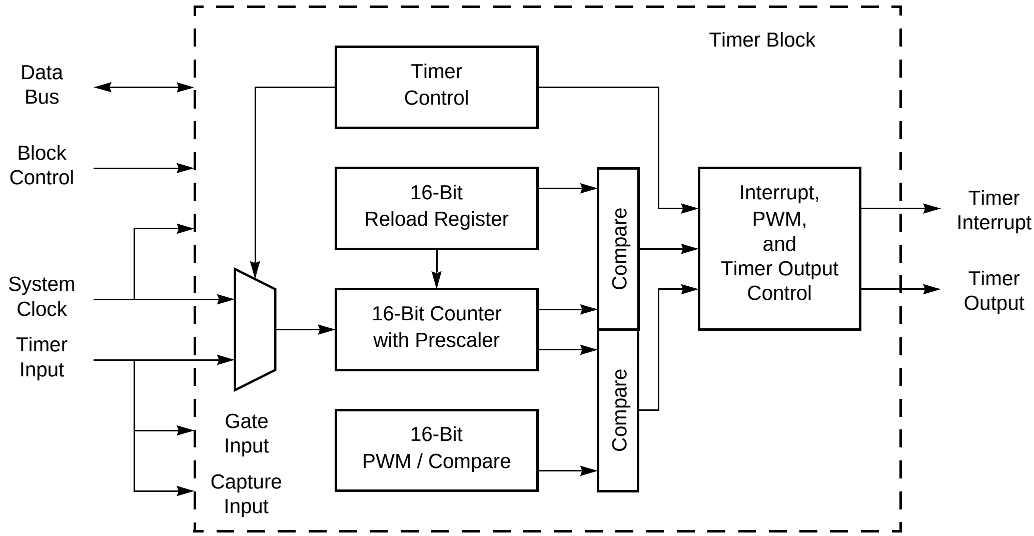

Figure 66. Timer Block Diagram

Operation

The timers are 16-bit up-counters. Minimum time-out delay is set by loading the value `0001H` into the Timer Reload High and Low Byte registers and setting the prescale value to 1. Maximum time-out delay is set by loading the value `0000H` into the Timer Reload High and Low Byte registers and setting the prescale value to 128. If the Timer reaches `FFFFH`, the timer rolls over to `0000H` and continues counting.

Timer Operating Modes

The timers can be configured to operate in the following modes:

One-Shot Mode

In One-Shot mode, the timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Reload value, the timer generates an interrupt and the count value in the Timer High and Low Byte registers is reset to `0001H`. Then, the timer is automatically disabled and stops counting.

Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state for one system clock cycle (from Low to High or from High to Low) upon timer Reload. If it is desired to have the Timer Output make a permanent state change upon One-Shot time-out, first set the TPOL bit in the Timer Control Register to the start value before beginning One-Shot mode. Then, after starting the timer, set TPOL to the opposite bit value.

The steps for configuring a timer for One-Shot mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for One-Shot mode.
   - Set the prescale value.
   - If using the Timer Output alternate function, set the initial output level (High or Low).
2. Write to the Timer High and Low Byte registers to set the starting count value.
3. Write to the Timer Reload High and Low Byte registers to set the Reload value.
4. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
5. If using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
6. Write to the Timer Control register to enable the timer and initiate counting.

In One-Shot mode, the system clock always provides the timer input. The timer period is given by the following equation:

$$\textbf{One-Shot Mode Time-Out Period (s)} = \frac{(\textbf{Reload Value} - \textbf{Start Value}) \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

Continuous Mode

In Continuous mode, the timer counts up to the 16-bit Reload value stored in the Timer Reload High and Low Byte registers. The timer input is the system clock. Upon reaching the Reload value, the timer generates an interrupt, the count value in the Timer High and Low Byte registers is reset to 0001H and counting resumes. Also, if the Timer Output alternate function is enabled, the Timer Output pin changes state (from Low to High or from High to Low) upon timer Reload.

The steps for configuring a timer for Continuous mode and initiating the count are as follows:

1. Write to the Timer Control register to:
   - Disable the timer
   - Configure the timer for Continuous mode.
   - Set the prescale value.

- Disable the timer
- Configure the timer for PWM mode.
- Set the prescale value.
- Set the initial logic level (High or Low) and PWM High/Low transition for the Timer Output alternate function.

2. Write to the Timer High and Low Byte registers to set the starting count value (typically `0001H`). This only affects the first pass in PWM mode. After the first timer reset in PWM mode, counting always begins at the reset value of `0001H`.
3. Write to the PWM High and Low Byte registers to set the PWM value.
4. Write to the Timer Reload High and Low Byte registers to set the Reload value (PWM period). The Reload value must be greater than the PWM value.
5. If desired, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin for the Timer Output alternate function.
7. Write to the Timer Control register to enable the timer and initiate counting.

The PWM period is given by the following equation:

$$\textbf{PWM Period (s)} = \frac{\textbf{Reload Value} \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

If an initial starting value other than `0001H` is loaded into the Timer High and Low Byte registers, the One-Shot mode equation must be used to determine the first PWM time-out period.

If `TPOL` is set to 0, the ratio of the PWM output High time to the total period is given by:

$$\textbf{PWM Output High Time Ratio (\%)} = \frac{\textbf{Reload Value} - \textbf{PWM Value}}{\textbf{Reload Value}} \times \textbf{100}$$

If `TPOL` is set to 1, the ratio of the PWM output High time to the total period is given by:

$$\textbf{PWM Output High Time Ratio (\%)} = \frac{\textbf{PWM Value}}{\textbf{Reload Value}} \times \textbf{100}$$

Capture Mode

In Capture mode, the current timer count value is recorded when the desired external Timer Input transition occurs. The Capture count value is written to the Timer PWM High and Low Byte Registers. The timer input is the system clock. The `TPOL` bit in the Timer Control register determines if the Capture occurs on a rising edge or a falling edge of the

5. Configure the associated GPIO port pin for the Timer Input alternate function.
6. Write to the Timer Control register to enable the timer.
7. Counting begins on the first appropriate transition of the Timer Input signal. No interrupt is generated by this first edge.

In Capture/Compare mode, the elapsed time from timer start to Capture event can be calculated using the following equation:

$$\textbf{Capture Elapsed Time (s)} = \frac{(\textbf{Capture Value} - \textbf{Start Value}) \times \textbf{Prescale}}{\textbf{System Clock Frequency (Hz)}}$$

Reading the Timer Count Values

The current count value in the timers can be read while counting (enabled). This capability has no effect on timer operation. When the timer is enabled and the Timer High Byte register is read, the contents of the Timer Low Byte register are placed in a holding register. A subsequent read from the Timer Low Byte register returns the value in the holding register. This operation allows accurate reads of the full 16-bit timer count value while enabled. When the timers are not enabled, a read from the Timer Low Byte register returns the actual value in the counter.

Timer Output Signal Operation

Timer Output is a GPIO Port pin alternate function. Generally, the Timer Output is toggled every time the counter is reloaded.

Timer Control Register Definitions

Timers 0–2 are available in all packages. Timer 3 is available only in the 64-, 68- and 80-pin packages.

Timer 0-3 High and Low Byte Registers

The Timer 0-3 High and Low Byte (TxH and TxL) registers (Tables 38 and 39) contain the current 16-bit timer count value. When the timer is enabled, a read from TxH causes the value in TxL to be stored in a temporary holding register. A read from TMRL always returns this temporary register when the timers are enabled. When the timer is disabled, reads from the TMRL reads the register directly.

Writing to the Timer High and Low Byte registers while the timer is enabled is not recommended. There are no temporary holding registers available for write operations, so simultaneous 16-bit writes are not possible. If either the Timer High or Low Byte registers are

Infrared Encoder/Decoder

Overview

The Z8F640x family products contain two fully-functional, high-performance UART to Infrared Encoder/Decoders (Endecs). Each Infrared Endec is integrated with an on-chip UART to allow easy communication between the Z8F640x family device and IrDA Physical Layer Specification Version 1.3-compliant infrared transceivers. Infrared communication provides secure, reliable, low-cost, point-to-point communication between PCs, PDAs, cell phones, printers and other infrared enabled devices.

Architecture

Figure 71 illustrates the architecture of the Infrared Endec.

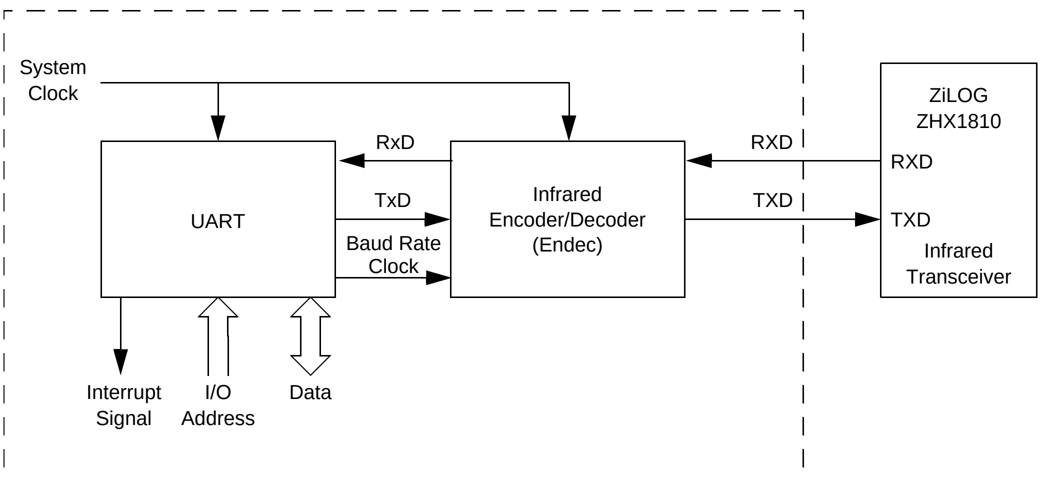

Figure 71. Infrared Data Communication System Block Diagram

4. The I^2C Controller loads the I^2C Shift register with the contents of the I^2C Data register.
5. After the first bit has been shifted out, a Transmit interrupt is asserted.
6. Software responds by writing eight bits of address to the I^2C Data register.
7. The I^2C Controller completes shifting of the two address bits and a 0 (write).
8. The I^2C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
9. The I^2C Controller loads the I^2C Shift register with the contents of the I^2C Data register.
10. The I^2C Controller shifts out the next eight bits of address. After the first bits are shifted, the I^2C Controller generates a Transmit interrupt.
11. Software responds by setting the `START` bit of the I^2C Control register to generate a repeated START.
12. Software responds by writing `11110B` followed by the 2-bit slave address and a 1 (read).
13. Software responds by setting the NAK bit of the I^2C Control register, so that a Not Acknowledge is sent after the first byte of data has been read. If you want to read only one byte, software responds by setting the NAK bit of the I^2C Control register.
14. After the I^2C Controller shifts out the address bits mentioned in step 9, the I^2C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
15. The I^2C Controller sends the repeated START condition.
16. The I^2C Controller loads the I^2C Shift register with the contents of the I^2C Data register.
17. The I^2C Controller sends `11110B` followed by the 2-bit slave read and a 1 (read).
18. The I^2C slave sends an acknowledge by pulling the SDA signal Low during the next high period of SCL.
19. The I^2C slave sends a byte of data.
20. A Receive interrupt is generated.
21. Software responds by reading the I^2C Data register.
22. Software responds by setting the `STOP` bit of the I^2C Control register.
23. A NAK condition is sent to the I^2C slave.
24. A STOP condition is sent to the I^2C slave.

Direct Memory Access Controller

Overview

The Z8F640x family device’s Direct Memory Access (DMA) Controller provides three independent Direct Memory Access channels. Two of the channels (DMA0 and DMA1) transfer data between the on-chip peripherals and the Register File. The third channel (DMA_ADC) controls the Analog-to-Digital Converter (ADC) operation and transfers the Single-Shot mode ADC output data to the Register File.

Operation

DMA0 and DMA1 Operation

DMA0 and DMA1, referred to collectively as DMA*x*, transfer data either from the on-chip peripheral control registers to the Register File, or from the Register File to the on-chip peripheral control registers. The sequence of operations in a DMA*x* data transfer is:

1. DMA*x* trigger source requests a DMA data transfer.
2. DMA*x* requests control of the system bus (address and data) from the eZ8 CPU.
3. After the eZ8 CPU acknowledges the bus request, DMA*x* transfers either a single byte or a two-byte word (depending upon configuration) and then returns system bus control back to the eZ8 CPU.
4. If Current Address equals End Address:
   - DMA*x* reloads the original Start Address
   - If configured to generate an interrupt, DMA*x* sends an interrupt request to the Interrupt Controller
   - If configured for single-pass operation, DMA*x* resets the `DEN` bit in the DMA*x* Control register to 0 and the DMA is disabled.

   If Current Address does not equal End Address, the Current Address increments by 1 (single-byte transfer) or 2 (two-byte word transfer).

Table 74. DMA*x* Start/Current Address Low Byte Register (DMA*x*START)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_START							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FB3H, FHBH							

DMA_START—DMA*x* Start/Current Address Low
These bits, with the four lower bits of the DMA*x*_H register, form the 12-bit Start/Current address. The full 12-bit address is given by {DMA_START_H[3:0], DMA_START[7:0]}.

DMA*x* End Address Low Byte Register

The DMA*x* End Address Low Byte register, in conjunction with the DMA*x*_H register, forms a 12-bit End Address.

Table 75. DMA*x* End Address Low Byte Register (DMA*x*END)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_END							
RESET	X	X	X	X	X	X	X	X
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
ADDR	FB4H, FBCH							

DMA_END—DMA*x* End Address Low
These bits, with the four upper bits of the DMA*x*_H register, form a 12-bit address. This address is the ending location of the DMA*x* transfer. The full 12-bit address is given by {DMA_END_H[3:0], DMA_END[7:0]}.

DMA_ADC Address Register

The DMA_ADC Address register points to a block of the Register File to store ADC conversion values as illustrated in Table 76. This register contains the seven most-significant bits of the 12-bit Register File addresses. The five least-significant bits are calculated from the ADC Analog Input number (5-bit base address is equal to twice the ADC Analog Input number). The 10-bit ADC conversion data is stored as two bytes with the most significant byte of the ADC data stored at the even numbered Register File address.

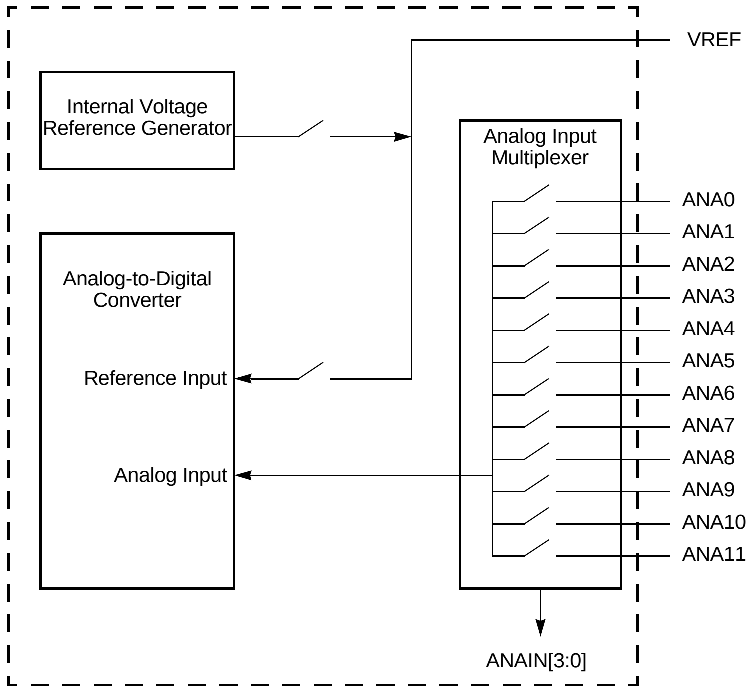

Figure 83. Analog-to-Digital Converter Block Diagram

Operation

Automatic Power-Down

If the ADC is idle (no conversions in progress) for 160 consecutive system clock cycles, portions of the ADC are automatically powered-down. From this power-down state, the ADC requires 40 system clock cycles to power-up. The ADC powers up when a conversion is requested using the ADC Control register.

Single-Shot Conversion

When configured for single-shot conversion, the ADC performs a single analog-to-digital conversion on the selected analog input channel. After completion of the conversion, the ADC shuts down. The steps for setting up the ADC and initiating a single-shot conversion are as follows:

Operation

OCD Interface

The On-Chip Debugger uses the DBG pin for communication with an external host. This one-pin interface is a bi-directional open-drain interface that transmits and receives data. Data transmission is half-duplex, in that transmit and receive cannot occur simultaneously. The serial data on the DBG pin is sent using the standard asynchronous data format defined in RS-232. This pin can interface the Z8F640x family device to the serial port of a host PC using minimal external hardware.Two different methods for connecting the DBG pin to an RS-232 interface are depicted in Figures 87 and 88.

Caution: For operation of the On-Chip Debugger, ***all*** power pins (VDD and AVDD) must be supplied with power, and ***all*** ground pins (VSS and AVSS) must be properly grounded.
The DBG pin is open-drain and must always be connected to V_{DD} through an external pull-up resistor to ensure proper operation.

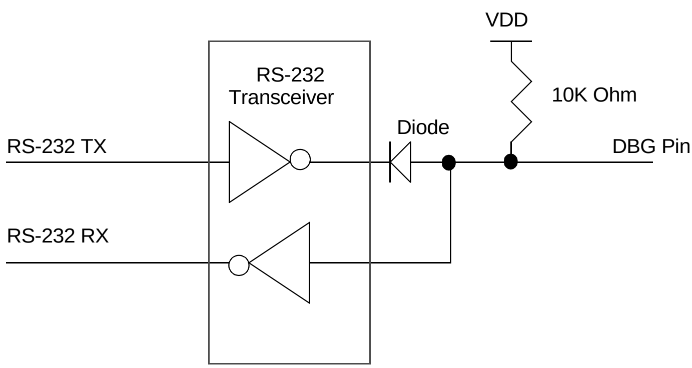

Figure 87. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1)

ister. When the Watchpoint event occurs, the Z8F640x family device enters Debug mode and the DBGMODE bit in the OCDCTL register becomes 1.

Runtime Counter

The On-Chip Debugger contains a 16-bit Runtime Counter. It counts system clock cycles between Breakpoints. The counter starts counting when the On-Chip Debugger leaves Debug mode and stops counting when it enters Debug mode again or when it reaches the maximum count of FFFFH.

On-Chip Debugger Commands

The host communicates to the On-Chip Debugger by sending OCD commands using the DBG interface. During normal operation of the Z8F640x family device, only a subset of the OCD commands are available. In Debug mode, all OCD commands become available unless the user code and control registers are protected by programming the Read Protect Option Bit (RP). The Read Protect Option Bit prevents the code in memory from being read out of the Z8F640x family device. When this option is enabled, several of the OCD commands are disabled. Table 93 contains a summary of the On-Chip Debugger commands. Each OCD command is described in further detail in the bulleted list following Table 93. Table 93 indicates those commands that operate when the Z8F640x family device is not in Debug mode (normal operation) and those commands that are disabled by programming the Read Protect Option Bit.

Table 93. On-Chip Debugger Commands

Debug Command	Command Byte	Enabled when NOT in Debug mode?	Disabled by Read Protect Option Bit
Read OCD Revision	00H	Yes	-
Reserved	01H	-	-
Read OCD Status Register	02H	Yes	-
Read Runtime Counter	03H	-	-
Write OCD Control Register	04H	Yes	Cannot clear DBGMODE bit
Read OCD Control Register	05H	Yes	-
Write Program Counter	06H	-	Disabled
Read Program Counter	07H	-	Disabled
Write Register	08H	-	Only writes of the Flash Memory Control registers are allowed. Additionally, only the Mass Erase command is allowed to be written to the Flash Control register.
Read Register	09H	-	Disabled

AC Characteristics

The section provides information on the AC characteristics and timing of the Z8 Encore!™. All AC timing information assumes a standard load of 50 pF on all outputs.

Table 102. AC Characteristics

Symbol	Parameter	V_{DD} = 3.0 - 3.6V T_A =-40^{0}C to 105^{0}C		Units	Conditions
		Minimum	Maximum		
F_{sysclk}	System Clock Frequency	–	20.0	MHz	Read-only from Flash memory.
		0.032768	20.0	MHz	Program or erasure of the Flash memory.
F_{XTAL}	Crystal Oscillator Frequency	1.0	20.0	MHz	System clock frequencies below the crystal oscillator minimum require an external clock driver.
T_{XIN}	System Clock Period	50	–	ns	$T_{CLK} = 1/F_{sysclk}$
T_{XINH}	System Clock High Time	20	30	ns	T_{CLK} = 50ns
T_{XINL}	System Clock Low Time	20	30	ns	T_{CLK} = 50ns
T_{XINR}	System Clock Rise Time	–	3	ns	T_{CLK} = 50ns
T_{XINF}	System Clock Fall Time	–	3	ns	T_{CLK} = 50ns

Problem Description or Suggestion

Provide a complete description of the problem or your suggestion. If you are reporting a specific problem, include all steps leading up to the occurrence of the problem. Attach additional pages as necessary.