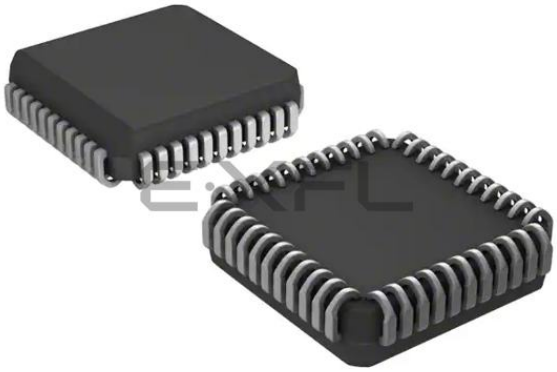


Welcome to E-XFL.COM

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	31
Program Memory Size	16KB (16K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	3V ~ 3.6V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1621vn020sc

Reset and Stop Mode Recovery

Overview

The Reset Controller within the Z8 Encore! XP 64K Series Flash Microcontrollers controls Reset and Stop Mode Recovery operation. In typical operation, the following events cause a Reset to occur:

- Power-On Reset
- Voltage Brownout
- Watchdog Timer time-out (when configured via the WDT_RES Option Bit to initiate a Reset)
- External $\overline{\text{RESET}}$ pin assertion
- On-Chip Debugger initiated Reset (OCDCTL[0] set to 1)

When the 64K Series devices are in STOP mode, a Stop Mode Recovery is initiated by either of the following:

- Watchdog Timer time-out
- GPIO Port input pin transition on an enabled Stop Mode Recovery source
- DBG pin driven Low

Reset Types

The 64K Series provides two different types of reset operation (system reset and Stop Mode Recovery). The type of Reset is a function of both the current operating mode of the 64K Series devices and the source of the Reset. Table 8 lists the types of Reset and their operating characteristics.

Table 12. Port Alternate Function Mapping (Continued)

Port	Pin	Mnemonic	Alternate Function Description
Port C	PC0	T1IN	Timer 1 Input
	PC1	T1OUT	Timer 1 Output
	PC2	SS	SPI Slave Select
	PC3	SCK	SPI Serial Clock
	PC4	MOSI	SPI Master Out/Slave In
	PC5	MISO	SPI Master In/Slave Out
	PC6	T2IN	Timer 2 In
	PC7	T2OUT	Timer 2 Out
Port D	PD0	T3IN	Timer 3 In (unavailable in 44-pin packages)
	PD1	T3OUT	Timer 3 Out (unavailable in 44-pin packages)
	PD2	N/A	No alternate function
	PD3	DE1	UART 1 Driver Enable
	PD4	RXD1/IRRX1	UART 1/IrDA 1 Receive Data
	PD5	TXD1/IRTX1	UART 1/IrDA 1 Transmit Data
	PD6	CTS1	UART 1 Clear to Send
	PD7	RCOUT	Watchdog 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). For more information on interrupts using the GPIO pins, see Interrupt Controller on page 67.

Port A–H Input Data Registers

Reading from the Port A–H Input Data registers (Table 21) returns the sampled values from the corresponding port pins. The Port A–H Input Data registers are Read-only.

Table 21. Port A–H Input Data Registers (PxIN)

BITS	7	6	5	4	3	2	1	0
FIELD	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X							
R/W	R							
ADDR	FD2H, FD6H, FDAH, FDEH, FE2H, FE6H, FEAH, FEEH							

PIN[7:0]—Port Input Data
Sampled data from the corresponding port pin input.
0 = Input data is logical 0 (Low).
1 = Input data is logical 1 (High).

Port A–H Output Data Register

The Port A–H Output Data register (Table 22) writes output data to the pins.

Table 22. Port A–H Output Data Register (P*x*OUT)

BITS	7	6	5	4	3	2	1	0
FIELD	POUT7	POUT6	POUT5	POUT4	POUT3	POUT2	POUT1	POUT0
RESET	0							
R/W	R/W							
ADDR	FD3H, FD7H, FDBH, FDFH, FE3H, FE7H, FEBH, FEFH							

POUT[7:0]—Port Output Data
These bits contain the data to be driven out from the port pins. The values are only driven if the corresponding pin is configured as an output and the pin is not configured for alternate function operation.
0 = Drive a logical 0 (Low).
1= Drive a logical 1 (High). High value is not driven if the drain has been disabled by setting the corresponding Port Output Control register bit to 1.

PAD*x*S—PA*x*/PD*x* Selection
0 = PA*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
1 = PD*x* is used for the interrupt for PA*x*/PD*x* interrupt request.
where *x* indicates the specific GPIO Port pin number (0 through 7).

Interrupt Control Register

The Interrupt Control (IRQCTL) register (Table 38) contains the master enable bit for all interrupts.

Table 38. Interrupt Control Register (IRQCTL)

BITS	7	6	5	4	3	2	1	0
FIELD	IRQE	Reserved						
RESET	0							
R/W	R/W	R						
ADDR	FCFH							

IRQE—Interrupt Request Enable
This bit is set to 1 by execution of an EI or IRET instruction, or by a direct register write of a 1 to this bit. It is reset to 0 by executing a DI instruction, eZ8 CPU acknowledgement of an interrupt request, or Reset.
0 = Interrupts are disabled
1 = Interrupts are enabled

Reserved—Must be 0.

Watchdog Timer

Overview

The Watchdog Timer (WDT) helps protect against corrupt or unreliable software, power faults, and other system-level problems which may place the Z8 Encore! XP into unsuitable operating states. The features of Watchdog Timer include:

- On-chip RC oscillator.
- A selectable time-out response.
- WDT Time-out response: Reset or interrupt.
- 24-bit programmable time-out value.

Operation

The Watchdog Timer (WDT) is a retriggerable one-shot timer that resets or interrupts the 64K Series devices when the WDT reaches its terminal count. The Watchdog Timer uses its own dedicated on-chip RC oscillator as its clock source. The Watchdog Timer has only two modes of operation—ON and OFF. Once enabled, it always counts and must be refreshed to prevent a time-out. An enable can be performed by executing the WDT instruction or by setting the WDT_AO Option Bit. The WDT_AO bit enables the Watchdog Timer to operate all the time, even if a WDT instruction has not been executed.

The Watchdog Timer is a 24-bit reloadable downcounter that uses three 8-bit registers in the eZ8™ CPU register space to set the reload value. The nominal WDT time-out period is given by the following equation:

$$\text{WDT Time-out Period (ms)} = \frac{\text{WDT Reload Value}}{10}$$

where the WDT reload value is the decimal value of the 24-bit value given by {WDTU[7:0], WDTH[7:0], WDTL[7:0]} and the typical Watchdog Timer RC oscillator frequency is 10 kHz. The Watchdog Timer cannot be refreshed once it reaches 000002H. The WDT Reload Value must not be set to values below 000004H. Table 47 provides information on approximate time-out delays for the minimum and maximum WDT reload values.

repeated. If the incoming data is a logical 1 (no pulse), the Endec returns to the initial state and waits for the next falling edge. As each falling edge is detected, the Endec clock counter is reset, resynchronizing the Endec to the incoming signal. This action allows the Endec to tolerate jitter and baud rate errors in the incoming data stream. Resynchronizing the Endec does not alter the operation of the UART, which ultimately receives the data. The UART is only synchronized to the incoming data stream when a Start bit is received.

Infrared Encoder/Decoder Control Register Definitions

All Infrared Endec configuration and status information is set by the UART control registers as defined in UART Control Register Definitions on page 114.

Caution: *To prevent spurious signals during IrDA data transmission, set the* `IREN` *bit in the UARTx Control 1 register to 1 to enable the Infrared Encoder/Decoder before enabling the GPIO Port alternate function for the corresponding pin.*

Serial Peripheral Interface

Overview

The Serial Peripheral Interface is a synchronous interface allowing several SPI-type devices to be interconnected. SPI-compatible devices include EEPROMs, Analog-to-Digital Converters, and ISDN devices. Features of the SPI include:

- Full-duplex, synchronous, character-oriented communication
- Four-wire interface
- Data transfers rates up to a maximum of one-half the system clock frequency
- Error detection
- Dedicated Baud Rate Generator

Architecture

The SPI may be configured as either a Master (in single or multi-master systems) or a Slave as displayed in Figure 22 through Figure 24.

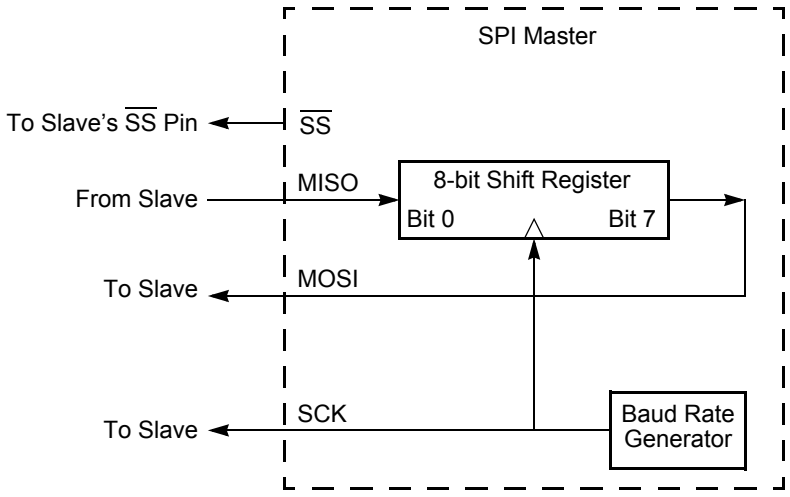


Figure 22. SPI Configured as a Master in a Single Master, Single Slave System

register. The IRQE, PHASE, CLKPOL, WOR bits in the SPICTL register and the NUMBITS field in the SPIMODE register must be set to be consistent with the other SPI devices. The STR bit in the SPICTL register may be used if desired to force a "startup" interrupt. The BIRQ bit in the SPICTL register and the SSV bit in the SPIMODE register are not used in SLAVE mode. The SPI baud rate generator is not used in SLAVE mode so the SPIBRH and SPIBRL registers need not be initialized.

If the slave has data to send to the master, the data must be written to the SPIDAT register before the transaction starts (first edge of SCK when $\overline{SS}$ is asserted). If the SPIDAT register is not written prior to the slave transaction, the MISO pin outputs whatever value is currently in the SPIDAT register.

Due to the delay resulting from synchronization of the SPI input signals to the internal system clock, the maximum SPICLK baud rate that can be supported in SLAVE mode is the system clock frequency (XIN) divided by 8. This rate is controlled by the SPI master.

Error Detection

The SPI contains error detection logic to support SPI communication protocols and recognize when communication errors have occurred. The SPI Status register indicates when a data transmission error has been detected.

Overrun (Write Collision)

An overrun error (write collision) indicates a write to the SPI Data register was attempted while a data transfer is in progress (in either MASTER or SLAVE modes). An overrun sets the OVR bit in the SPI Status register to 1. Writing a 1 to OVR clears this error Flag. The data register is not altered when a write occurs while data transfer is in progress.

Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one Master is trying to communicate at the same time (a multi-master collision). The mode fault is detected when the enabled Master's $\overline{SS}$ pin is asserted. A mode fault sets the COL bit in the SPI Status register to 1. Writing a 1 to COL clears this error Flag.

Slave Mode Abort

In SLAVE mode of operation if the $\overline{SS}$ pin deasserts before all bits in a character have been transferred, the transaction is aborted. When this condition occurs the ABT bit is set in the SPISTAT register as well as the IRQ bit (indicating the transaction is complete). The next time $\overline{SS}$ asserts, the MISO pin outputs SPIDAT[7], regardless of where the previous transaction left off. Writing a 1 to ABT clears this error Flag.

SPI Interrupts

When SPI interrupts are enabled, the SPI generates an interrupt after character transmission/reception completes in both MASTER and SLAVE modes. A character can be

- Master receives from a 7-bit slave
- Master receives from a 10-bit slave

SDA and SCL Signals

I²C sends all addresses, data and acknowledge signals over the SDA line, most-significant bit first. SCL is the common clock for the I²C Controller. When the SDA and SCL pin alternate functions are selected for their respective GPIO ports, the pins are automatically configured for open-drain operation.

The master (I²C) is responsible for driving the SCL clock signal, although the clock signal can become skewed by a slow slave device. During the low period of the clock, the slave pulls the SCL signal Low to suspend the transaction. The master releases the clock at the end of the low period and notices that the clock remains low instead of returning to a high level. When the slave releases the clock, the I²C Controller continues the transaction. All data is transferred in bytes and there is no limit to the amount of data transferred in one operation. When transmitting data or acknowledging read data from the slave, the SDA signal changes in the middle of the low period of SCL and is sampled in the middle of the high period of SCL.

I²C Interrupts

The I²C Controller contains four sources of interrupts—Transmit, Receive, Not Acknowledge and baud rate generator. These four interrupt sources are combined into a single interrupt request signal to the Interrupt Controller. The Transmit interrupt is enabled by the IEN and TXI bits of the Control register. The Receive and Not Acknowledge interrupts are enabled by the IEN bit of the Control register. The baud rate generator interrupt is enabled by the BIRQ and IEN bits of the Control register.

Not Acknowledge interrupts occur when a Not Acknowledge condition is received from the slave or sent by the I²C Controller and neither the START or STOP bit is set. The Not Acknowledge event sets the NCKI bit of the I²C Status register and can only be cleared by setting the START or STOP bit in the I²C Control register. When this interrupt occurs, the I²C Controller waits until either the STOP or START bit is set before performing any action. In an interrupt service routine, the NCKI bit should always be checked prior to servicing transmit or receive interrupt conditions because it indicates the transaction is being terminated.

Receive interrupts occur when a byte of data has been received by the I²C Controller (master reading data from slave). This procedure sets the RDRF bit of the I²C Status register. The RDRF bit is cleared by reading the I²C Data register. The RDRF bit is set during the acknowledge phase. The I²C Controller pauses after the acknowledge phase until the receive interrupt is cleared before performing any other action.

TXRXSTATE	State Description
0_0000	Idle State
0_0001	START State
0_0010	Send/Receive data bit 7
0_0011	Send/Receive data bit 6
0_0100	Send/Receive data bit 5
0_0101	Send/Receive data bit 4
0_0110	Send/Receive data bit 3
0_0111	Send/Receive data bit 2
0_1000	Send/Receive data bit 1
0_1001	Send/Receive data bit 0
0_1010	Data Acknowledge State
0_1011	Second half of data Acknowledge State used only for not acknowledge
0_1100	First part of STOP state
0_1101	Second part of STOP state
0_1110	10-bit addressing: Acknowledge State for 2nd address byte 7-bit addressing: Address Acknowledge State
0_1111	10-bit address: Bit 0 (Least significant bit) of 2nd address byte 7-bit address: Bit 0 (Least significant bit) (R/W) of address byte
1_0000	10-bit addressing: Bit 7 (Most significant bit) of 1st address byte
1_0001	10-bit addressing: Bit 6 of 1st address byte
1_0010	10-bit addressing: Bit 5 of 1st address byte
1_0011	10-bit addressing: Bit 4 of 1st address byte
1_0100	10-bit addressing: Bit 3 of 1st address byte
1_0101	10-bit addressing: Bit 2 of 1st address byte
1_0110	10-bit addressing: Bit 1 of 1st address byte
1_0111	10-bit addressing: Bit 0 (R/W) of 1st address byte
1_1000	10-bit addressing: Acknowledge state for 1st address byte
1_1001	10-bit addressing: Bit 7 of 2nd address byte 7-bit addressing: Bit 7 of address byte
1_1010	10-bit addressing: Bit 6 of 2nd address byte 7-bit addressing: Bit 6 of address byte
1_1011	10-bit addressing: Bit 5 of 2nd address byte 7-bit addressing: Bit 5 of address byte
1_1100	10-bit addressing: Bit 4 of 2nd address byte 7-bit addressing: Bit 4 of address byte

Direct Memory Access Controller

Overview

The 64K Series 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 ADC operation and transfers 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).

DMA*x*_IO[7:0]}. When the DMA is configured for two-byte word transfers, the DMA*x* I/O Address register must contain an even numbered address.

Table 78. DMA*x* I/O Address Register (DMA*x*IO)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_IO							
RESET	X							
R/W	R/W							
ADDR	FB1H, FB9H							

DMA_IO—DMA on-chip peripheral control register address
This byte sets the low byte of the on-chip peripheral control register address on Register File Page FH (addresses F00H to FFFH).

DMA*x* Address High Nibble Register

The DMA*x* Address High register (Table 79) specifies the upper four bits of address for the Start/Current and End Addresses of DMA*x*.

Table 79. DMA*x* Address High Nibble Register (DMA*x*H)

BITS	7	6	5	4	3	2	1	0
FIELD	DMA_END_H				DMA_START_H			
RESET	X							
R/W	R/W							
ADDR	FB2H, FBAH							

DMA_END_H—DMA*x* End Address High Nibble
These bits, used with the DMA*x* End Address Low register, form a 12-bit End Address. The full 12-bit address is given by {DMA_END_H[3:0], DMA_END[7:0]}.

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

DMAA_ADDR—DMA_ADC Address
These bits specify the seven most-significant bits of the 12-bit Register File addresses used for storing the ADC output data. The ADC Analog Input Number defines the five least-significant bits of the Register File address. Full 12-bit address is {DMAA_ADDR[7:1], 4-bit ADC Analog Input Number, 0}.

Reserved
This bit is reserved and must be 0.

DMA_ADC Control Register

The DMA_ADC Control register (Table 84 on page 172) enables and sets options (DMA enable and interrupt enable) for ADC operation.

Table 84. DMA_ADC Control Register (DMAACTL)

BITS	7	6	5	4	3	2	1	0
FIELD	DAEN	IRQEN	Reserved		ADC_IN			
RESET	0							
R/W	R/W							
ADDR	FBEH							

DAEN—DMA_ADC Enable
0 = DMA_ADC is disabled and the ADC Analog Input Number (ADC_IN) is reset to 0.
1 = DMA_ADC is enabled.

IRQEN—Interrupt Enable
0 = DMA_ADC does not generate any interrupts.
1 = DMA_ADC generates an interrupt after transferring data from the last ADC Analog Input specified by the ADC_IN field.

Reserved
These bits are reserved and must be 0.

ADC_IN—ADC Analog Input Number
These bits set the number of ADC Analog Inputs to be used in the continuous update (data conversion followed by DMA data transfer). The conversion always begins with ADC Analog Input 0 and then progresses sequentially through the other selected ADC Analog Inputs.
0000 = ADC Analog Input 0 updated.
0001 = ADC Analog Inputs 0-1 updated.
0010 = ADC Analog Inputs 0-2 updated.
0011 = ADC Analog Inputs 0-3 updated.
0100 = ADC Analog Inputs 0-4 updated.

On-Chip Debugger

Overview

The 64K Series products contain an integrated On-Chip Debugger (OCD) that provides advanced debugging features including:

- Reading and writing of the Register File
- Reading and writing of Program and Data Memory
- Setting of Breakpoints
- Execution of eZ8 CPU instructions

Architecture

The On-Chip Debugger consists of four primary functional blocks: transmitter, receiver, auto-baud generator, and debug controller. Figure 36 displays the architecture of the On-Chip Debugger.

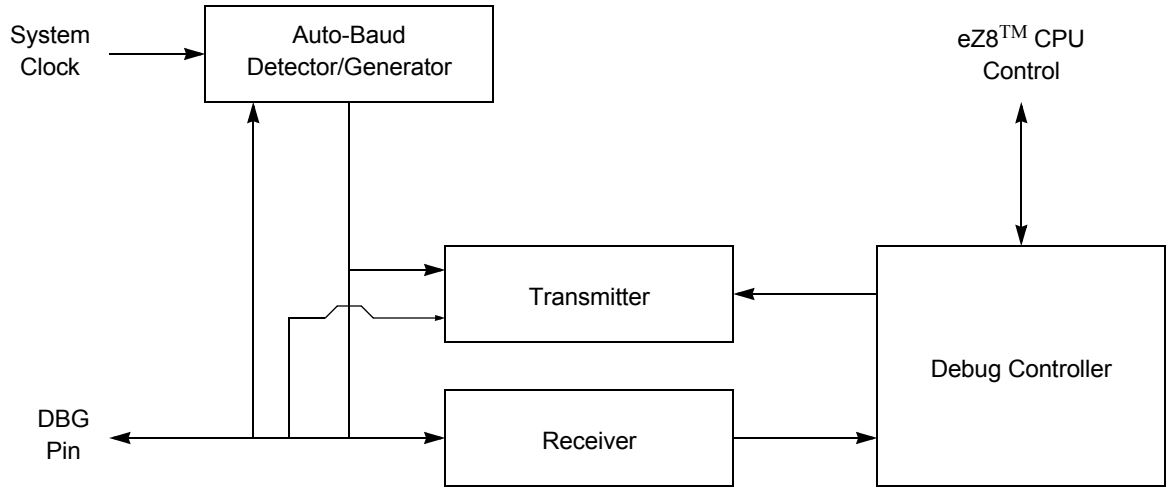


Figure 36. On-Chip Debugger Block Diagram

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 64K Series products 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 Figure 37 and Figure 38 on page 201.

Caution: *For operation of the On-Chip Debugger, **all** power pins (V_{DD} and AV_{DD}) must be supplied with power, and **all** ground pins (V_{SS} and AV_{SS}) 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.

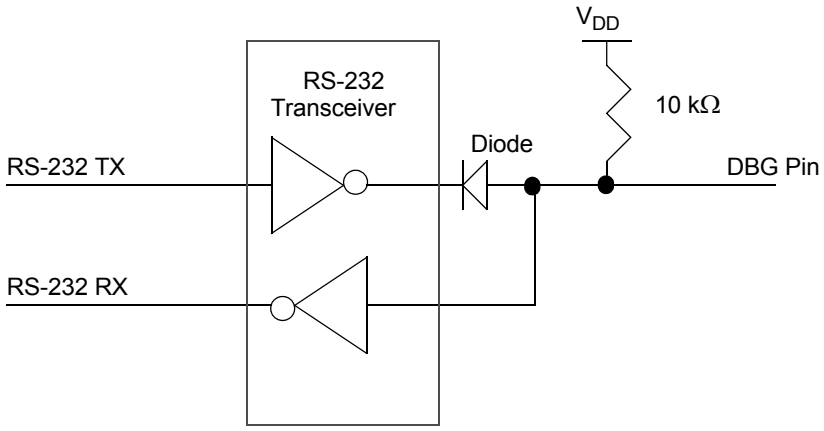


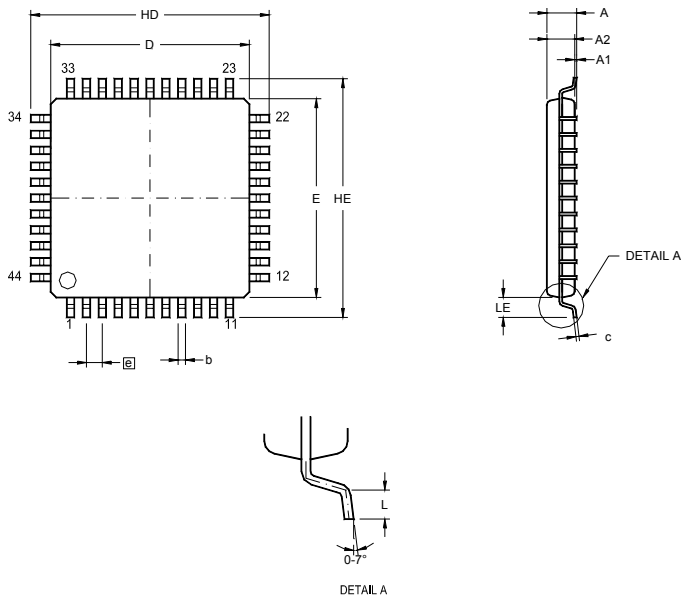
Figure 37. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface (1)

Table 122. Notational Shorthand

Notation	Description	Operand	Range
b	Bit	b	b represents a value from 0 to 7 (000B to 111B).
cc	Condition Code	—	Refer to Condition Codes overview in the eZ8 CPU User Manual.
DA	Direct Address	Addrs	Addrs. represents a number in the range of 0000H to FFFFH
ER	Extended Addressing Register	Reg	Reg. represents a number in the range of 000H to FFFH
IM	Immediate Data	#Data	Data is a number between 00H to FFH
Ir	Indirect Working Register	@Rn	n = 0 –15
IR	Indirect Register	@Reg	Reg. represents a number in the range of 00H to FFH
Irr	Indirect Working Register Pair	@RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
IRR	Indirect Register Pair	@Reg	Reg. represents an even number in the range 00H to FEH
p	Polarity	p	Polarity is a single bit binary value of either 0B or 1B.
r	Working Register	Rn	n = 0 – 15
R	Register	Reg	Reg. represents a number in the range of 00H to FFH
RA	Relative Address	X	X represents an index in the range of +127 to -128 which is an offset relative to the address of the next instruction
rr	Working Register Pair	RRp	p = 0, 2, 4, 6, 8, 10, 12, or 14
RR	Register Pair	Reg	Reg. represents an even number in the range of 00H to FEH
Vector	Vector Address	Vector	Vector represents a number in the range of 00H to FFH
X	Indexed	#Index	The register or register pair to be indexed is offset by the signed Index value (#Index) in a +127 to -128 range.

Table 123 contains additional symbols that are used throughout the Instruction Summary and Instruction Set Description sections.

Figure 63 displays the 44-pin Low Profile Quad Flat Package (LQFP) available for the Z8X1621, Z8X2421, Z8X3221, Z8X4821, and Z8X6421 devices.



SYMBOL	MILLIMETER		INCH	
	MIN	MAX	MIN	MAX
A	1.40	1.60	0.055	0.063
A1	0.05	0.15	0.002	0.006
A2	1.35	1.45	0.053	0.057
b	0.30	0.45	0.012	0.018
c	0.09	0.20	0.004	0.008
HD	11.75	12.25	0.463	0.482
D	9.90	10.10	0.390	0.398
HE	11.75	12.25	0.463	0.482
E	9.90	10.10	0.390	0.398
e	0.80 BSC		0.031 BSC	
L	0.45	0.75	0.018	0.030
LE	1.00 REF		0.039 REF	

1. CONTROLLING DIMENSIONS : mm
2. MAX. COPLANARITY : .10mm 0.004"

Figure 63. 44-Lead Low-Profile Quad Flat Package (LQFP)