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, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	37
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	Surface Mount
Package / Case	44-VFQFN Exposed Pad
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f1680qn020sg

1.4. An Overview of the eZ8 CPU and its Peripherals

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

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

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

1.4.1. General-Purpose Input/Output

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

1.4.2. Flash Controller

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

used as Baud Rate Generator (BRG) when UART is enabled and configured as basic 16-bit timers when UART is disabled.

1.4.16. Multi-Channel Timer

The multi-channel timer has a 16-bit up/down counter and a 4-channel Capture/Compare/PWM channel array. This timer enables the support of multiple synchronous Capture/Compare/PWM channels based on a single timer.

1.4.17. Interrupt Controller

The Z8 Encore! XP F1680 Series products support up to thirty-one interrupt sources with twenty-four interrupt vectors. These interrupts consist of up to fifteen internal peripheral interrupts and up to sixteen GPIO pin interrupts. The interrupts have three levels of programmable-interrupt priority.

1.4.18. Reset Controller

The F1680 Series MCU is reset using the $\overline{\text{RESET}}$ pin, POR, WDT time-out, STOP Mode exit, or VBO warning signal. The $\overline{\text{RESET}}$ pin is bidirectional, that is, it functions as reset source as well as a reset indicator.

1.4.19. On-Chip Debugger

The F1680 Series MCU features an integrated OCD. The OCD provides a rich-set of debugging capabilities, such as reading and writing registers, programming Flash memory, setting breakpoints and executing code. The OCD uses one single-pin interface for communication with an external host.

1.4.20. Direct LED Drive

The Port C pins also provide a current synchronized output capable of driving an LED without requiring any external resistor. Up to eight LEDs are driven with individually programmable drive current level from 3 mA to 20 mA.

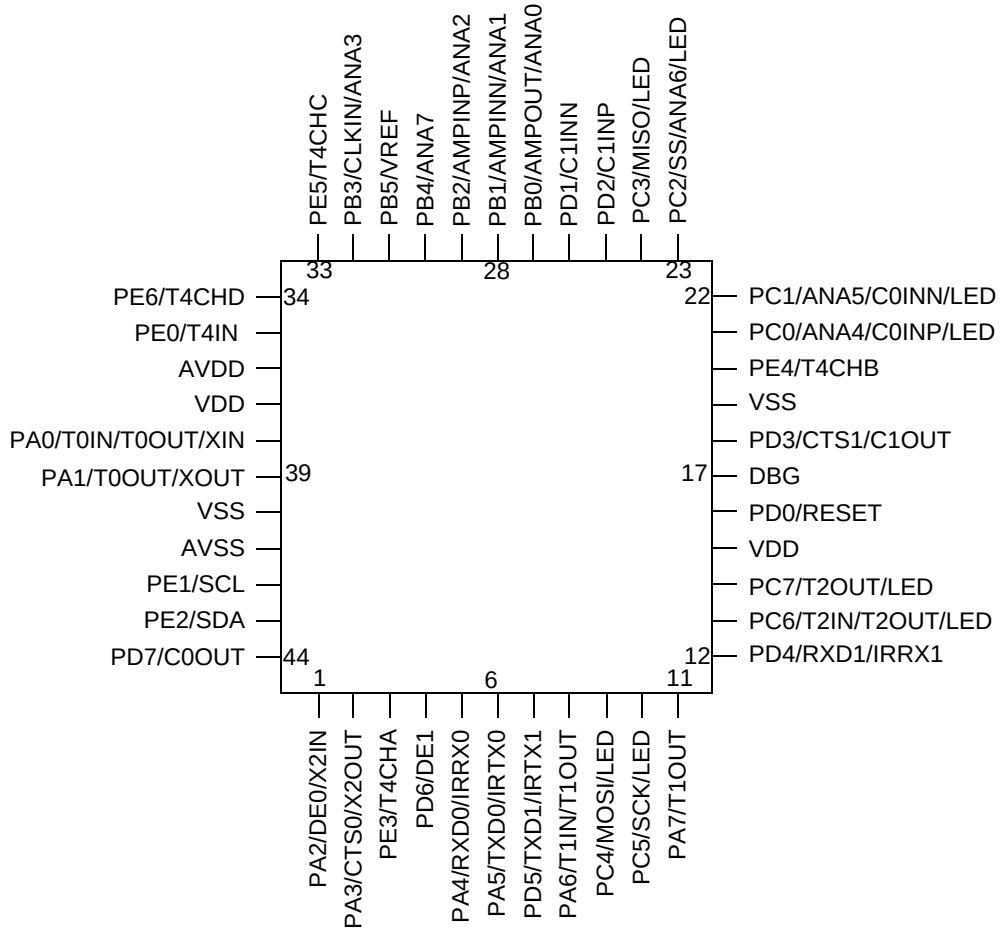

Figure 5. Z8F2480, Z8F1680 and Z8F0880 in 44-Pin Low-Profile Quad Flat Package (LQFP) or Quad Flat No Lead (QFN)

To minimize current in STOP Mode, all GPIO pins which are configured as digital inputs must be driven to one of the supply rails (V_{DD} or GND). The device is brought out of STOP Mode using Stop Mode Recovery. For more details about Stop Mode Recovery, see the Reset, Stop Mode Recovery and Low-Voltage Detection chapter on page 31.

6.2. HALT Mode

Executing the eZ8 CPU's HALT instruction places the device into HALT Mode. In HALT Mode, the operating characteristics are:

- Primary oscillator is enabled and continues to operate
- System clock is enabled and continues to operate
- eZ8 CPU is stopped
- Program counter (PC) stops incrementing
- WDT's internal RC oscillator continues to operate
- If enabled, the WDT continues to operate
- If enabled, the 32kHz secondary oscillator for Timers continues to operate
- All other on-chip peripherals continue to operate

The eZ8 CPU can be brought out of HALT Mode by any of the following operations:

- Interrupt
- Watchdog Timer time-out (Interrupt or Reset)
- Power-On Reset
- Voltage Brown-Out Reset
- External $\overline{\text{RESET}}$ pin assertion

To minimize current in HALT Mode, all GPIO pins which are configured as inputs must be driven to one of the supply rails (V_{DD} or GND).

6.3. Peripheral-Level Power Control

In addition to the STOP and HALT modes, it is possible to disable each peripheral on each of the Z8 Encore! XP F1680 Series devices. Disabling a given peripheral minimizes its power consumption.

7.2. Architecture

Figure 9 displays a simplified block diagram of a GPIO port pin and does not illustrate the ability to accommodate alternate functions and variable port current drive strength.

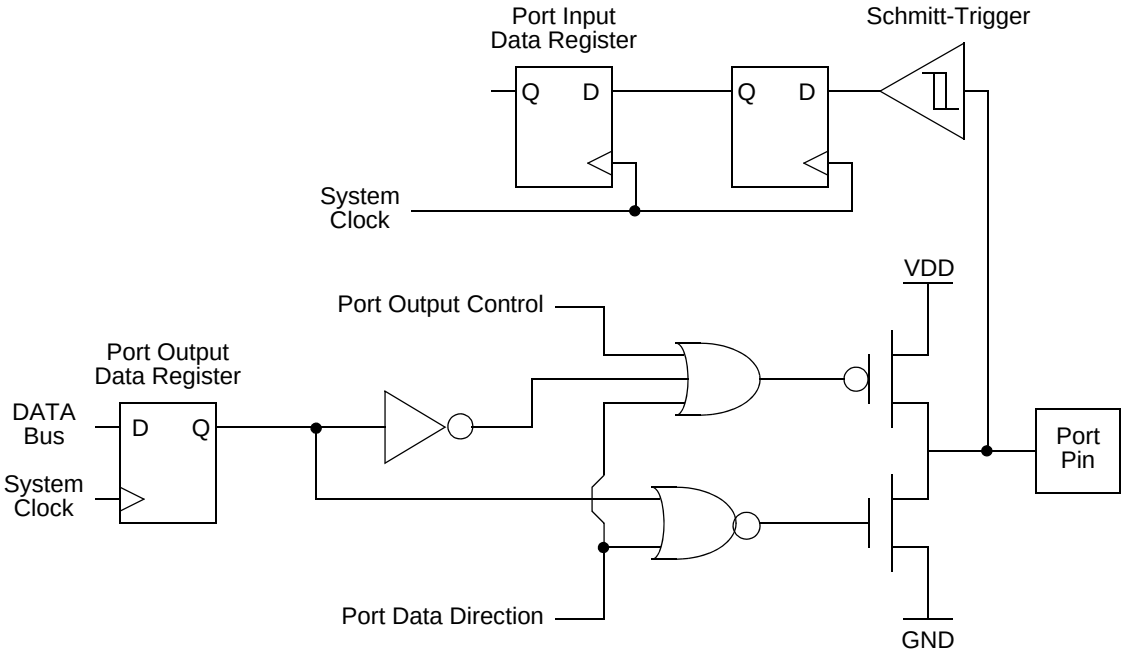

Figure 9. GPIO Port Pin Block Diagram

7.3. GPIO Alternate Functions

Many GPIO port pins are used for GPIO and to access the on-chip peripheral functions like the timers and serial-communication devices. The Port A–E Alternate Function sub-registers configure these pins for either GPIO or alternate function operation. When a pin is configured for alternate function, control of port-pin direction (input/output) is passed from Port A–E Data Direction registers to the alternate functions assigned to this pin. Tables 17 through 19 list the alternate functions possible with each port pin for every package. The alternate function associated at a pin is defined through alternate function sets subregisters AFS1 and AFS2.

The crystal oscillator and the 32 kHz secondary oscillator functionalities are not controlled by the GPIO block. When the crystal oscillator or the 32 kHz secondary oscillator is enabled in the oscillator control block, the GPIO functionality of PA0 and PA1, or PA2 and PA3, is overridden. In such a case, those pins function as input and output for the crystal oscillator.

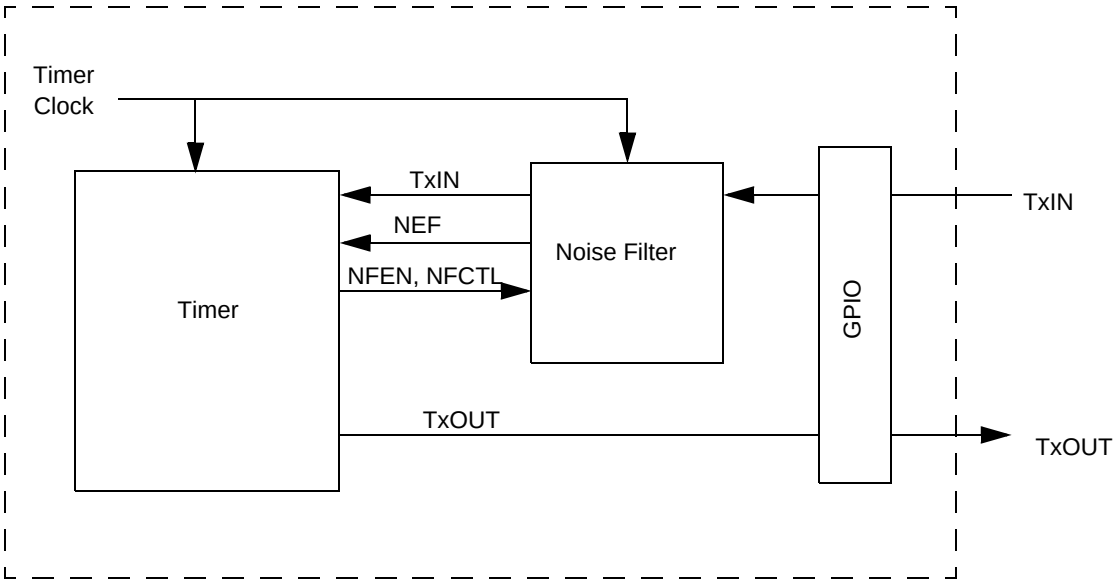

Figure 12. Noise Filter System Block Diagram

9.2.7.1. Operation

Figure 13 displays the operation of the Noise Filter with and without noise. The Noise Filter in this example is a 2-bit up/down counter which saturates at 00 and 11. A 2-bit counter is described for convenience; the operation of wider counters is similar. The output of the filter switches from 1 to 0 when the counter counts down from 01 to 00 and switches from 0 to 1 when the counter counts up from 10 to 11. The Noise Filter delays the receive data by three timer clock cycles.

The NEF output signal is checked when the filtered TxIN input signal is sampled. The Timer samples the filtered TxIN input near the center of the bit time. The NEF signal must be sampled at the same time to detect whether there is noise near the center of the bit time. The presence of noise (NEF = 1 at the center of the bit time) does not mean that the sampled data is incorrect; rather, it is intended to be an indicator of the level of noise in the network.

Table 72. Multi-Channel Timer Subaddress Register (MCTSA)

Bit	7	6	5	4	3	2	1	0
Field	MCTSA							
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
Address	FA4H							

10.7.5. Multi-Channel Timer Subregister x (0, 1, or 2)

The Multi-Channel Timer subregisters 0, 1 or 2 store the 8-bit data write to subregister or 8-bit data read from subregister. The Multi-Channel Timer Subaddress Register selects the subregister to be written to or read from.

Table 73. Multi-Channel Timer Subregister x (MCTSRx)

Bit	7	6	5	4	3	2	1	0
Field	MCTSRx							
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
Address	FA5H, FA6H, FA7H							

10.7.6. Multi-Channel Timer Control 0, Control 1 Registers

The Multi-Channel Timer Control registers (MCTCTL0, MCTCTL1) control Multi-Channel Timer operation. Writes to the PRES field of the MCTCTL1 Register are buffered when TEN = 1 and will not take effect until the next end of the cycle count occurs.

Table 74. Multi-Channel Timer Control 0 Register (MCTCTL0)

Bit	7	6	5	4	3	2	1	0
Field	TCTST	CHST	TCIEN	Reserved	Reserved	TCLKS		
Reset	0	0	0	0	0	0	0	0
R/W	R/W	R	R/W	R	R	R/W	R/W	R/W
Address	See note.							
Note: If a 00H is in the Subaddress Register, it is accessible through Subregister 0.								

Note: In MULTIPROCESSOR Mode (MPEN=1), the receive-data interrupts are dependent on the multiprocessor configuration and the most recent address byte.

- A break is received
- A Receive Data Overrun or LIN Slave Autobaud Overrun Error is detected
- A Data Framing Error is detected
- A Parity Error is detected (physical layer error in LIN mode)

12.1.11.3. LIN-UART Overrun Errors

When an overrun error condition occurs, the LIN-UART prevents overwriting of the valid data currently in the Receive Data Register. The Break Detect and Overrun status bits are not displayed until after the valid data has been read.

After the valid data has been read, the OE bit of the Status 0 register is updated to indicate the overrun condition (and Break Detect, if applicable). The RDA bit is set to 1 to indicate that the Receive Data Register contains a data byte. However, because the overrun error occurred, this byte cannot contain valid data and must be ignored. The BRKD bit indicates if the overrun is caused by a break condition on the line. After reading the status byte indicating an overrun error, the Receive Data Register must be read again to clear the error bits in the LIN-UART Status 0 Register.

In LIN mode, an Overrun Error is signalled for receive-data overruns as described above and in the LIN Slave if the BRG Counter overflows during the autobaud sequence (the ATB bit will also be set in this case). There is no data associated with the autobaud overflow interrupt; however the Receive Data Register must be read to clear the OE bit. In this case, software must write a 10B to the LinState field, forcing the LIN slave back to a Wait for Break state.

12.1.11.4. LIN-UART Data- and Error-Handling Procedure

Figures 24 displays the recommended procedure for use in LIN-UART receiver interrupt service routines.

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.

! **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.

21.1.2. Option Bit Types

This section describes the User, Trim and Calibration option bit types.

21.1.2.1. User Option Bits

The user option bits are contained in the first two bytes of Program Memory. User access to these bits has been provided because these locations contain application-specific device configurations. The information contained here is lost when page 0 of the Program Memory is erased.

21.1.2.2. Trim Option Bits

The trim option bits are contained in the Flash memory information page. These bits are factory programmed values required to optimize the operation of onboard analog circuitry and cannot be permanently altered by the user. Program Memory can be erased without endangering these values. It is possible to alter working values of these bits by accessing the Trim Bit Address and Data registers, but these working values are lost after a power loss.

There are 32 bytes of trim data. To modify one of these values the user code must first write a value between 00H and 1FH into the Trim Bit Address Register. The next write to the Trim Bit Data Register changes the working value of the target trim data byte.

Reading the trim data requires the user code to write a value between 00H and 1FH into the Trim Bit Address Register. The next read from the Trim Bit Data Register returns the working value of the target trim data byte.

Note: The trim address ranges from information address 20-3F only. The remainder of the information page is not accessible via the trim bit address and data registers.

21.1.2.3. Calibration Option Bits

The calibration option bits are also contained in the information page. These bits are factory programmed values intended for use in software correcting the device's analog performance. To read these values, the user code must employ the LDC instruction to access the information area of the address space as defined in the Flash Information Area section on page 21.

The following code example shows how to read the calibration data from the Flash Information Area.

```
; get value at info address 60 (FE60h)
ldx FPS, #%80 ; enable access to flash info page
ld R0, #%FE
```

Chapter 23. On-Chip Debugger

The Z8 Encore! XP F1680 Series device contains 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
- Executing eZ8 CPU instructions

23.1. Architecture

The OCD consists of four primary functional blocks:

- Transmitter
- Receiver
- Autobaud detector/generator
- Debug controller

Figure 55 displays the architecture of the On-Chip Debugger.

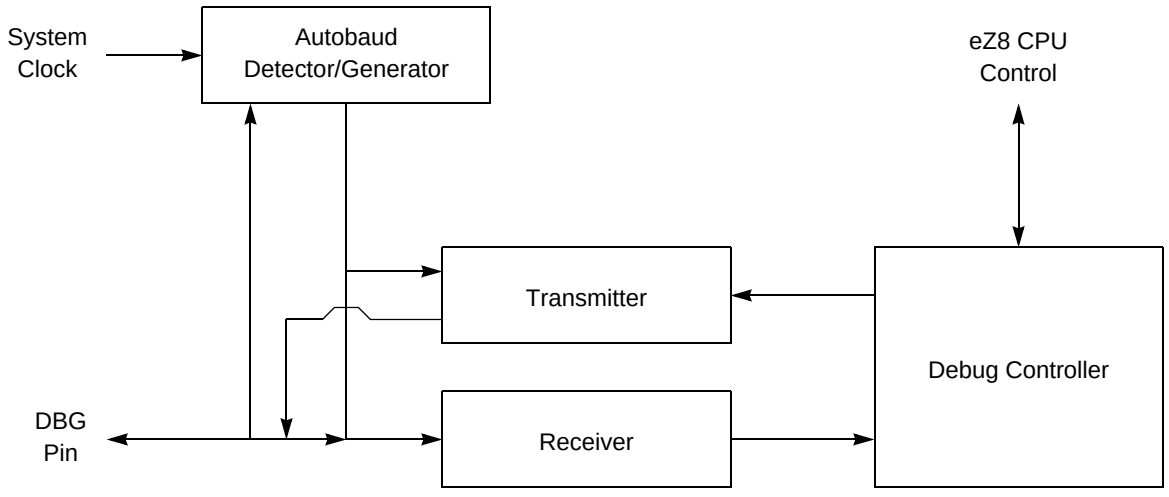

Figure 55. On-Chip Debugger Block Diagram

Chapter 27. eZ8 CPU Instruction Set

The eZ8 CPU instruction set is described in this chapter.

27.1. Assembly Language Programming Introduction

The eZ8 CPU assembly language provides a means for writing an application program without concern for actual memory addresses or machine instruction formats. A program written in assembly language is called a source program. Assembly language allows the use of symbolic addresses to identify memory locations. It also allows mnemonic codes (opcodes and operands) to represent the instructions themselves. The opcodes identify the instruction while the operands represent memory locations, registers, or immediate data values.

Each assembly language program consists of a series of symbolic commands called statements. Each statement contains labels, operations, operands and comments.

Labels are assigned to a particular instruction step in a source program. The label identifies that step in the program as an entry point for use by other instructions.

The assembly language also includes assembler directives that supplement the machine instruction. The assembler directives or pseudo-ops are not translated into a machine instruction. Rather, the pseudo-ops are interpreted as directives that control or assist the assembly process.

The source program is processed (assembled) by the assembler to obtain a machine language program called the object code. The object code is executed by the eZ8 CPU. An example segment of an assembly language program is detailed in the following example.

27.1.0.1. Assembly Language Source Program Example

```
JP START        ; Everything after the semicolon is a comment.
START:          ; A label called START. The first instruction (JP START) in this
                ; example causes program execution to jump to the point within the
                ; program where the start label occurs.
LD R4, R7       ; A Load (LD) instruction with two operands. The first operand,
                ; Working Register R4, is the destination. The second operand,
                ; Working Register R7, is the source. The contents of R7 is
                ; written into R4.
```

27.3. eZ8 CPU Instruction Notation

In the eZ8 CPU Instruction Summary section on page 336, the operands, condition codes, status flags and address modes are represented by the notational shorthand provided in Table 176.

Table 176. Notational Shorthand

Notation	Description	Operand	Range
b	Bit	b	b represents a value from 0 to 7 (000B to 111B)
cc	Condition Code	—	See the Condition Codes overview in the eZ8 CPU Core User Manual (UM0128)
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 184. Program Control Instructions

Mnemonic	Operands	Instruction
BRK	—	On-Chip Debugger Break
BTJ	p, bit, src, DA	Bit Test and Jump
BTJNZ	bit, src, DA	Bit Test and Jump if Non-Zero
BTJZ	bit, src, DA	Bit Test and Jump if Zero
CALL	dst	Call Procedure
DJNZ	dst, src, RA	Decrement and Jump Non-Zero
IRET	—	Interrupt Return
JP	dst	Jump
JP cc	dst	Jump Conditional
JR	DA	Jump Relative
JR cc	DA	Jump Relative Conditional
RET	—	Return
TRAP	vector	Software Trap

Table 185. Rotate and Shift Instructions

Mnemonic	Operands	Instruction
BSWAP	dst	Bit Swap
RL	dst	Rotate Left
RLC	dst	Rotate Left through Carry
RR	dst	Rotate Right
RRC	dst	Rotate Right through Carry
SRA	dst	Shift Right Arithmetic
SRL	dst	Shift Right Logical
SWAP	dst	Swap Nibbles

29.4.2. General Purpose I/O Port Output Timing

Figure 76 and Table 205 provide timing information for the GPIO port pins.

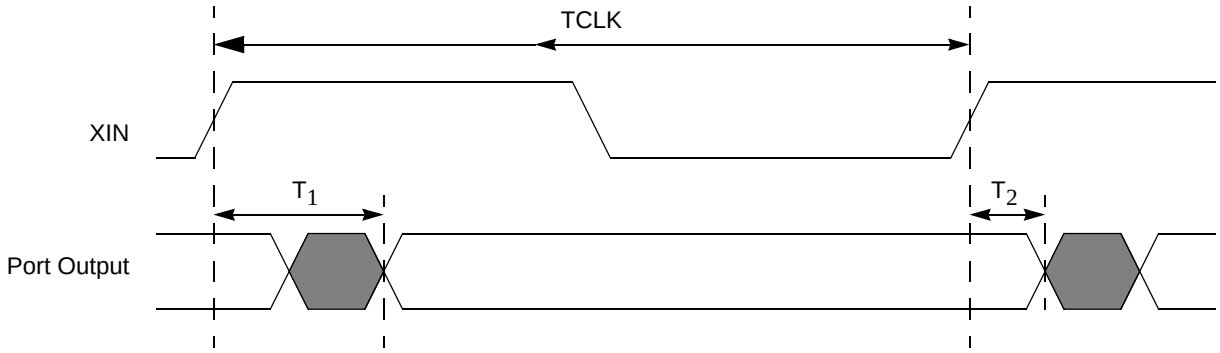

Figure 76. GPIO Port Output Timing

Table 205. GPIO Port Output Timing

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
GPIO Port Pins			
T_1	XIN Rise to Port Output Valid Delay	–	15
T_2	XIN Rise to Port Output Hold Time	2	–

29.4.4. UART Timing

Figure 78 and Table 207 provide timing information for the UART pins for situations in which CTS is used for flow control. The CTS to DE assertion delay (T1) assumes that the Transmit Data Register has been loaded with data prior to CTS assertion.

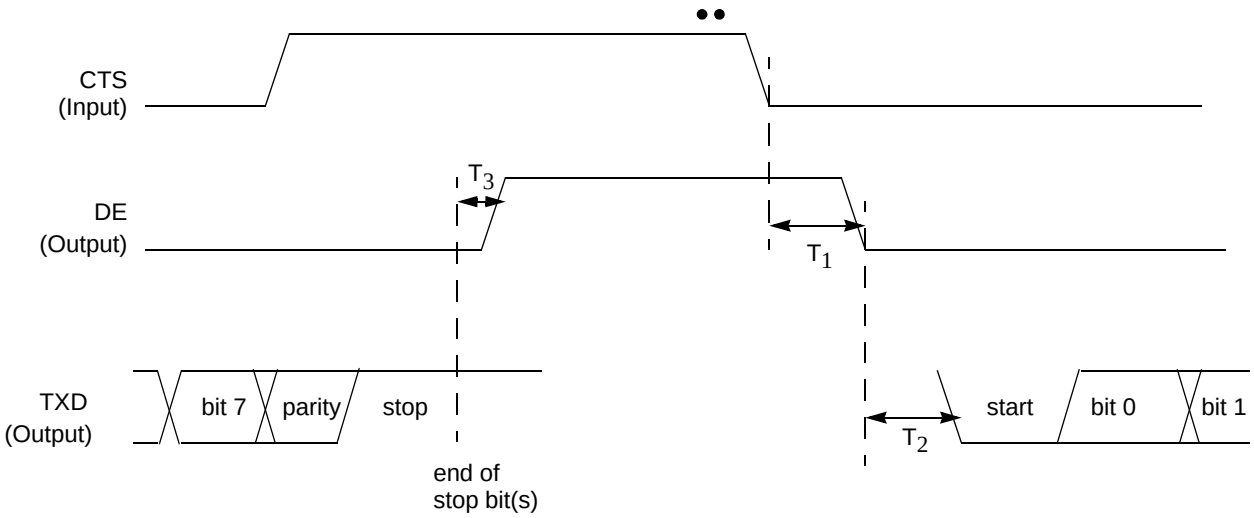

Figure 78. UART Timing With CTS

Table 207. UART Timing with CTS

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
UART			
T_1	CTS Fall to DE output delay	2 * X_{IN} period	2 * X_{IN} period + 1 bit time
T_2	DE assertion to TXD falling edge (start bit) delay	± 5	
T_3	End of stop bit(s) to DE deassertion delay	± 5	

Figure 79 and Table 208 provide timing information for the UART pins for situations in which CTS is not used for flow control. DE asserts after the Transmit Data Register has been written. DE remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

Index