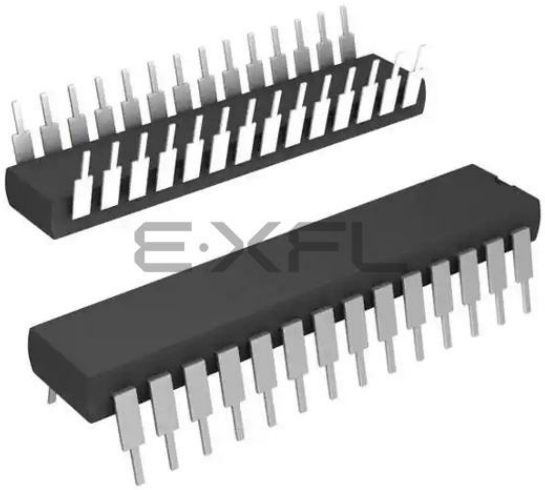

**Welcome to E-XFL.COM**

## 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 | IrDA, UART/USART |
| Peripherals | Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT |
| Number of I/O | 23 |
| Program Memory Size | 8KB (8K x 8) |
| Program Memory Type | FLASH |
| EEPROM Size | - |
| RAM Size | 1K x 8 |
| Voltage - Supply (Vcc/Vdd) | 2.7V ~ 3.6V |
| Data Converters | A/D 8x10b |
| Oscillator Type | Internal |
| Operating Temperature | 0°C ~ 70°C (TA) |
| Mounting Type | Through Hole |
| Package / Case | 28-DIP (0.600", 15.24mm) |
| Supplier Device Package | - |
| Purchase URL | https://www.e-xfl.com/product-detail/zilog/z8f082apj020sg |

# Revision History

Each instance in this document's revision history reflects a change from its previous edition. For more details, refer to the corresponding page(s) or appropriate links furnished in the table below.

| Date | Revision Level | Chapter/Section | Description | Page No. |
|---|---|---|---|---|
| Dec 2012 | 27 | Port Alternate Function Mapping (Non 8-Pin Parts), Port Alternate Function Mapping (8-Pin Parts) | Added missing Port D data to Table 15; corrected active Low status (set overlines) for PA0 ($\overline{T0OUT}$), PA2 ($\overline{RESET}$) and PA5 ($\overline{T1OUT}$) in Table 16. | 40, 43 |
| Sep 2011 | 26 | LED Drive Enable Register | Clarified statement surrounding the Alternate Function Register as it relates to the LED function; revised Flash Sector Protect Register description; revised Packaging chapter. | 53, 157, 245 |
| Sep 2008 | 25 | Overview, Address Space, Register Map, General-Purpose Input/Output, Available Packages, Ordering Information | Added references to F042A Series back in Table 1, Table 5, Table 7 and Table 14. | 2, 8, 16, 18, 36, 246 |
| May 2008 | 24 | Overview, Address Space, Register Map, General-Purpose Input/Output, Available Packages, Ordering Information | Changed title to Z8 Encore! XP F082A Series and removed references to F042A Series in Table 1, Table 5, Table 7 and Table 14. | 2, 8, 16, 18, 36, 246 |
| Dec 2007 | 23 | Pin Description, General-Purpose Input/Output, Watchdog Timer | Updated Figure 3, Table 15, Tables 60 through 62. | 9, 40, 97 |
| Jul 2007 | 22 | Electrical Characteristics | Updated Tables 16 and 132; power consumption data. | 43, 229 |
| Jun 2007 | 21 | n/a | Revision number update. | All |

**Table 16. Port Alternate Function Mapping (8-Pin Parts)**

| Port | Pin | Mnemonic | Alternate Function Description | Alternate Function Select Register AFS1 | Alternate Function Select Register AFS2 |
|---|---|---|---|---|---|
| **Port A** | PA0 | T0IN | Timer 0 Input | AFS1[0]: 0 | AFS2[0]: 0 |
| | | Reserved | | AFS1[0]: 0 | AFS2[0]: 1 |
| | | Reserved | | AFS1[0]: 1 | AFS2[0]: 0 |
| | | $\overline{\text{T0OUT}}$ | Timer 0 Output Complement | AFS1[0]: 1 | AFS2[0]: 1 |
| | PA1 | T0OUT | Timer 0 Output | AFS1[1]: 0 | AFS2[1]: 0 |
| | | Reserved | | AFS1[1]: 0 | AFS2[1]: 1 |
| | | CLKIN | External Clock Input | AFS1[1]: 1 | AFS2[1]: 0 |
| | | Analog Functions[1] | ADC Analog Input/$V_{REF}$ | AFS1[1]: 1 | AFS2[1]: 1 |
| | PA2 | DE0 | UART 0 Driver Enable | AFS1[2]: 0 | AFS2[2]: 0 |
| | | RESET | External Reset | AFS1[2]: 0 | AFS2[2]: 1 |
| | | T1OUT | Timer 1 Output | AFS1[2]: 1 | AFS2[2]: 0 |
| | | Reserved | | AFS1[2]: 1 | AFS2[2]: 1 |
| | PA3 | $\overline{\text{CTS0}}$ | $\overline{\text{UART 0 Clear to Send}}$ | AFS1[3]: 0 | AFS2[3]: 0 |
| | | COUT | Comparator Output | AFS1[3]: 0 | AFS2[3]: 1 |
| | | T1IN | Timer 1 Input | AFS1[3]: 1 | AFS2[3]: 0 |
| | | Analog Functions[2] | ADC Analog Input/LPO Input (P) | AFS1[3]: 1 | AFS2[3]: 1 |
| | PA4 | RXD0 | UART 0 Receive Data | AFS1[4]: 0 | AFS2[4]: 0 |
| | | Reserved | | AFS1[4]: 0 | AFS2[4]: 1 |
| | | Reserved | | AFS1[4]: 1 | AFS2[4]: 0 |
| | | Analog Functions[2] | ADC/Comparator Input (N)/LPO Input (N) | AFS1[4]: 1 | AFS2[4]: 1 |
| | PA5 | TXD0 | UART 0 Transmit Data | AFS1[5]: 0 | AFS2[5]: 0 |
| | | $\overline{\text{T1OUT}}$ | Timer 1 Output Complement | AFS1[5]: 0 | AFS2[5]: 1 |
| | | Reserved | | AFS1[5]: 1 | AFS2[5]: 0 |
| | | Analog Functions[2] | ADC/Comparator Input (P) LPO Output | AFS1[5]: 1 | AFS2[5]: 1 |

Notes:
1. Analog functions include ADC inputs, ADC reference, comparator inputs and LPO ports.
2. The alternate function selection must be enabled; see the Port A–D Alternate Function Subregisters (PxAF) section on page 47 for details.

delay ensures a time gap between the deassertion of one PWM output to the assertion of its complement.

Observe the following steps for configuring a timer for PWM DUAL OUTPUT Mode and initiating the PWM operation:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for PWM DUAL OUTPUT Mode by writing the TMODE bits in the TxCTL1 Register and the TMODEHI bit in TxCTL0 Register
   - 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 PWM Control Register to set the PWM dead band delay value. The dead-band delay must be less than the duration of the positive phase of the PWM signal (as defined by the PWM high and low byte registers). It must also be less than the duration of the negative phase of the PWM signal (as defined by the difference between the PWM registers and the Timer Reload registers).
5. 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.
6. If appropriate, enable the timer interrupt and set the timer interrupt priority by writing to the relevant interrupt registers.
7. Configure the associated GPIO port pin for the Timer Output and Timer Output Complement alternate functions. The Timer Output Complement function is shared with the Timer Input function for both timers. Setting the timer mode to Dual PWM automatically switches the function from Timer In to Timer Out Complement.
8. Write to the Timer Control Register to enable the timer and initiate counting.

The PWM period is represented by the following equation:

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

If an initial starting value other than `0001H` is loaded into the Timer High and Low Byte registers, the ONE-SHOT Mode equation determines the first PWM time-out period.

## Watchdog Timer Refresh

When first enabled, the Watchdog Timer is loaded with the value in the Watchdog Timer Reload registers. The Watchdog Timer counts down to 000000H unless a WDT instruction is executed by the eZ8 CPU. Execution of the WDT instruction causes the downcounter to be reloaded with the WDT reload value stored in the Watchdog Timer Reload registers. Counting resumes following the reload operation.

When the Z8 Encore! XP F082A Series devices are operating in DEBUG Mode (using the on-chip debugger), the Watchdog Timer is continuously refreshed to prevent any Watchdog Timer time-outs.

## Watchdog Timer Time-Out Response

The Watchdog Timer times out when the counter reaches 000000H. A time-out of the Watchdog Timer generates either an interrupt or a system reset. The WDT_RES Flash option bit determines the time-out response of the Watchdog Timer. For information about programming the WDT_RES Flash option bit, see the Flash Option Bits chapter on page 159.

### WDT Interrupt in Normal Operation

If configured to generate an interrupt when a time-out occurs, the Watchdog Timer issues an interrupt request to the interrupt controller and sets the WDT status bit in the Reset Status (RSTSTAT) Register; see the Reset Status Register on page 29. If interrupts are enabled, the eZ8 CPU responds to the interrupt request by fetching the Watchdog Timer interrupt vector and executing code from the vector address. After time-out and interrupt generation, the Watchdog Timer counter rolls over to its maximum value of FFFFFH and continues counting. The Watchdog Timer counter is not automatically returned to its reload value.

The Reset Status (RSTSTAT) Register must be read before clearing the WDT interrupt. This read clears the WDT time-out Flag and prevents further WDT interrupts from immediately occurring.

### WDT Interrupt in STOP Mode

If configured to generate an interrupt when a time-out occurs and the Z8 Encore! XP F082A Series devices are in STOP Mode, the Watchdog Timer automatically initiates a Stop Mode Recovery and generates an interrupt request. Both the WDT status bit and the STOP bit in the Reset Status (RSTSTAT) Register are set to 1 following a WDT time-out in STOP Mode. For more information about Stop Mode Recovery, see the Reset, Stop Mode Recovery and Low Voltage Detection chapter on page 22.

If interrupts are enabled, following completion of the Stop Mode Recovery the eZ8 CPU responds to the interrupt request by fetching the Watchdog Timer interrupt vector and executing code from the vector address.

In MULTIPROCESSOR (9-bit) Mode, the Parity (9th) bit location becomes the multiprocessor control bit. The UART Control 1 and Status 1 registers provide MULTIPROCESSOR (9-bit) Mode control and status information. If an automatic address matching scheme is enabled, the UART Address Compare Register holds the network address of the device.

## MULTIPROCESSOR (9-bit) Mode Receive Interrupts

When MULTIPROCESSOR Mode is enabled, the UART only processes frames addressed to it. The determination of whether a frame of data is addressed to the UART can be made in hardware, software or some combination of the two, depending on the multiprocessor configuration bits. In general, the address compare feature reduces the load on the CPU, because it does not require access to the UART when it receives data directed to other devices on the multi-node network. The following three MULTIPROCESSOR Modes are available in hardware:

- Interrupt on all address bytes
- Interrupt on matched address bytes and correctly framed data bytes
- Interrupt only on correctly framed data bytes

These modes are selected with MPMD[1:0] in the UART Control 1 Register. For all multiprocessor modes, bit MPEN of the UART Control 1 Register must be set to 1.

The first scheme is enabled by writing 01b to MPMD[1:0]. In this mode, all incoming address bytes cause an interrupt, while data bytes never cause an interrupt. The interrupt service routine must manually check the address byte that caused triggered the interrupt. If it matches the UART address, the software clears MPMD[0]. Each new incoming byte interrupts the CPU. The software is responsible for determining the end of the frame. It checks for the end-of-frame by reading the MPRX bit of the UART Status 1 Register for each incoming byte. If MPRX=1, a new frame has begun. If the address of this new frame is different from the UART's address, MPMD[0] must be set to 1 causing the UART interrupts to go inactive until the next address byte. If the new frame's address matches the UART's, the data in the new frame is processed as well.

The second scheme requires the following: set MPMD[1:0] to 10B and write the UART's address into the UART Address Compare Register. This mode introduces additional hardware control, interrupting only on frames that match the UART's address. When an incoming address byte does not match the UART's address, it is ignored. All successive data bytes in this frame are also ignored. When a matching address byte occurs, an interrupt is issued and further interrupts now occur on each successive data byte. When the first data byte in the frame is read, the NEWFRM bit of the UART Status 1 Register is asserted. All successive data bytes have NEWFRM=0. When the next address byte occurs, the hardware compares it to the UART's address. If there is a match, the interrupts continues and the NEWFRM bit is set for the first byte of the new frame. If there is no match, the UART ignores all incoming bytes until the next address match.

**Table 63. UART Control 0 Register (U0CTL0)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | TEN | REN | CTSE | PEN | PSEL | SBRK | STOP | LBEN |
| 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 | F42H | | | | | | | |

| Bit | Description |
|---|---|
| [7]<br>TEN | **Transmit Enable**<br>This bit enables or disables the transmitter. The enable is also controlled by the $\overline{\text{CTS}}$ signal and the CTSE bit. If the $\overline{\text{CTS}}$ signal is Low and the CTSE bit is 1, the transmitter is enabled.<br>0 = Transmitter disabled.<br>1 = Transmitter enabled. |
| [6]<br>REN | **Receive Enable**<br>This bit enables or disables the receiver.<br>0 = Receiver disabled.<br>1 = Receiver enabled. |
| [5]<br>CTSE | **CTS Enable**<br>0 = The $\overline{\text{CTS}}$ signal has no effect on the transmitter.<br>1 = The UART recognizes the $\overline{\text{CTS}}$ signal as an enable control from the transmitter. |
| [4]<br>PEN | **Parity Enable**<br>This bit enables or disables parity. Even or odd is determined by the PSEL bit.<br>0 = Parity is disabled.<br>1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit. |
| [3]<br>PSEL | **Parity Select**<br>0 = Even parity is transmitted and expected on all received data.<br>1 = Odd parity is transmitted and expected on all received data. |
| [2]<br>SBRK | **Send Break**<br>This bit pauses or breaks data transmission. Sending a break interrupts any transmission in progress, so ensure that the transmitter has finished sending data before setting this bit.<br>0 = No break is sent.<br>1 = Forces a break condition by setting the output of the transmitter to zero. |
| [1]<br>STOP | **Stop Bit Select**<br>0 = The transmitter sends one stop bit.<br>1 = The transmitter sends two stop bits. |
| [0]<br>LBEN | **Loop Back Enable**<br>0 = Normal operation.<br>1 = All transmitted data is looped back to the receiver. |

| Bit | Description (Continued) |
|---|---|
| [2]<br>BRGCTL | **Baud Rate Control**<br>This bit causes an alternate UART behavior depending on the value of the REN bit in the UART Control 0 Register. When the UART receiver is not enabled (REN=0), this bit determines whether the Baud Rate Generator issues interrupts.<br>0 = Reads from the Baud Rate High and Low Byte registers return the BRG reload value.<br>1 = The Baud Rate Generator generates a receive interrupt when it counts down to 0. Reads from the Baud Rate High and Low Byte registers return the current BRG count value.<br>When the UART receiver is enabled (REN=1), this bit allows reads from the Baud Rate registers to return the BRG count value instead of the reload value.<br>0 = Reads from the Baud Rate High and Low Byte registers return the BRG reload value.<br>1 = Reads from the Baud Rate High and Low Byte registers return the current BRG count value. Unlike the Timers, there is no mechanism to latch the Low Byte when the High Byte is read. |
| [1]<br>RDAIRQ | **Receive Data Interrupt Enable**<br>0 = Received data and receiver errors generates an interrupt request to the Interrupt Controller.<br>1 = Received data does not generate an interrupt request to the Interrupt Controller. Only receiver errors generate an interrupt request. |
| [0]<br>IREN | **Infrared Encoder/Decoder Enable**<br>0 = Infrared Encoder/Decoder is disabled. UART operates normally.<br>1 = Infrared Encoder/Decoder is enabled. The UART transmits and receives data through the Infrared Encoder/Decoder. |

# Infrared Encoder/Decoder

Z8 Encore! XP F082A Series products contain a fully-functional, high-performance UART to Infrared Encoder/Decoder (endec). The infrared endec is integrated with an on-chip UART to allow easy communication between the Z8 Encore! XP MCU 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 16 displays the architecture of the infrared endec.

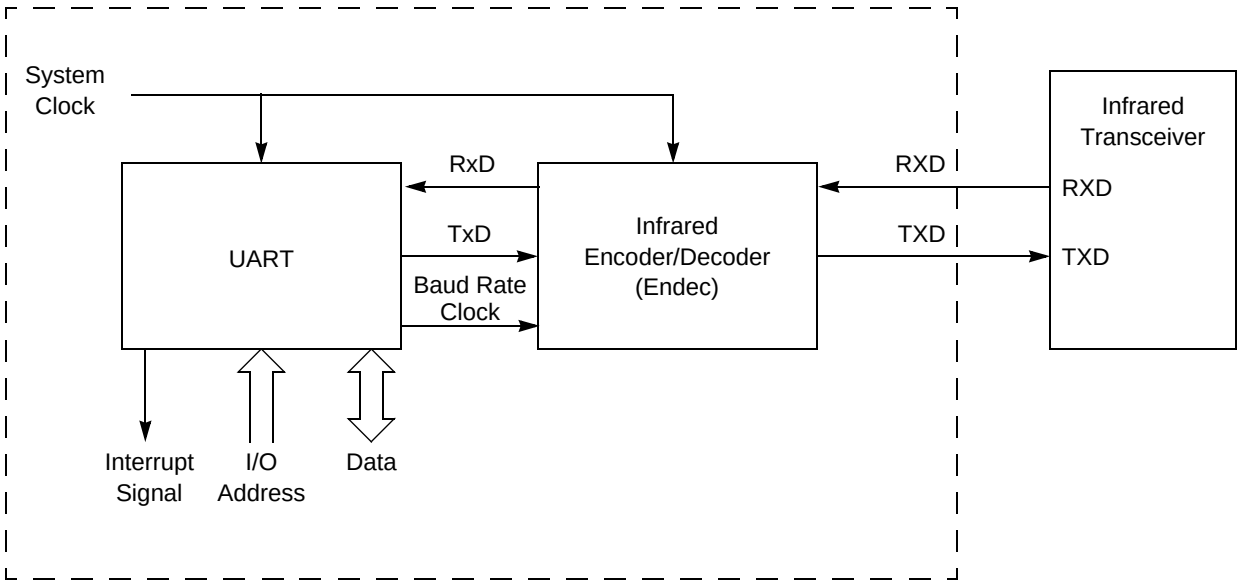


**Figure 16. Infrared Data Communication System Block Diagram**

## Operation

When the infrared endec is enabled, the transmit data from the associated on-chip UART is encoded as digital signals in accordance with the IrDA standard and output to the infrared transceiver through the TXD pin. Likewise, data received from the infrared transceiver is passed to the infrared endec through the RXD pin, decoded by the infrared endec and passed to the UART. Communication is half-duplex, which means simultaneous data transmission and reception is not allowed.

# Flash Option Bits

Programmable Flash option bits allow user configuration of certain aspects of Z8 Encore! XP F082A Series operation. The feature configuration data is stored in Flash program memory and loaded into holding registers during Reset. The features available for control through the Flash option bits include:

- Watchdog Timer time-out response selection–interrupt or system reset
- Watchdog Timer always on (enabled at Reset)
- The ability to prevent unwanted read access to user code in Program Memory
- The ability to prevent accidental programming and erasure of all or a portion of the user code in Program Memory
- Voltage Brown-Out configuration-always enabled or disabled during STOP Mode to reduce STOP Mode power consumption
- Oscillator mode selection-for high, medium and low power crystal oscillators, or external RC oscillator
- Factory trimming information for the internal precision oscillator and low voltage detection
- Factory calibration values for ADC, temperature sensor and Watchdog Timer compensation
- Factory serialization and randomized lot identifier (optional)

## Operation

This section describes the type and configuration of the programmable Flash option bits.

### Option Bit Configuration By Reset

Each time the Flash option bits are programmed or erased, the device must be Reset for the change to take effect. During any reset operation (System Reset, Power-On Reset, or Stop Mode Recovery), the Flash option bits are automatically read from Flash program memory and written to the Option Configuration registers. The Option Configuration registers control the operation of the devices within the Z8 Encore! XP F082A Series. Option bit control is established before the device exits Reset and the eZ8 CPU begins code execution. The Option Configuration registers are not part of the Register File and are not accessible for read or write access.

**Table 105. Randomized Lot ID Locations (Continued)**

| Info Page Address | Memory Address | Usage |
|---|---|---|
| 6A | FE6A | Randomized Lot ID Byte 13. |
| 6B | FE6B | Randomized Lot ID Byte 12. |
| 6D | FE6D | Randomized Lot ID Byte 11. |
| 6E | FE6E | Randomized Lot ID Byte 10. |
| 70 | FE70 | Randomized Lot ID Byte 9. |
| 71 | FE71 | Randomized Lot ID Byte 8. |
| 73 | FE73 | Randomized Lot ID Byte 7. |
| 74 | FE74 | Randomized Lot ID Byte 6. |
| 76 | FE76 | Randomized Lot ID Byte 5. |
| 77 | FE77 | Randomized Lot ID Byte 4. |
| 79 | FE79 | Randomized Lot ID Byte 3. |
| 7A | FE7A | Randomized Lot ID Byte 2. |
| 7C | FE7C | Randomized Lot ID Byte 1. |
| 7D | FE7D | Randomized Lot ID Byte 0 (least significant). |

## Byte Write

To write a byte to the NVDS array, the user code must first push the address, then the data byte onto the stack. The user code issues a CALL instruction to the address of the byte-write routine (0x10B3). At the return from the sub-routine, the write status byte resides in working register R0. The bit fields of this status byte are defined in Table 106. The contents of the status byte are undefined for write operations to illegal addresses. Also, user code must pop the address and data bytes off the stack.

The write routine uses 13 bytes of stack space in addition to the two bytes of address and data pushed by the user. Sufficient memory must be available for this stack usage.

Because of the Flash memory architecture, NVDS writes exhibit a nonuniform execution time. In general, a write takes 251µs (assuming a 20MHz system clock). Every 400 to 500 writes, however, a maintenance operation is necessary. In this rare occurrence, the write takes up to 61ms to complete. Slower system clock speeds result in proportionally higher execution times.

NVDS byte writes to invalid addresses (those exceeding the NVDS array size) have no effect. Illegal write operations have a 2µs execution time.

**Table 106. Write Status Byte**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | Reserved | | | | RCPY | PF | AWE | DWE |
| Default Value | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Description |
|---|---|
| [7:4] | **Reserved**<br>These bits are reserved and must be programmed to 0000. |
| [3]<br>RCPY | **Recopy Subroutine Executed**<br>A recopy subroutine was executed. These operations take significantly longer than a normal write operation. |
| [2]<br>PF | **Power Failure Indicator**<br>A power failure or system reset occurred during the most recent attempted write to the NVDS array. |
| [1]<br>AWE | **Address Write Error**<br>An address byte failure occurred during the most recent attempted write to the NVDS array. |
| [0]<br>DWE | **Data Write Error**<br>A data byte failure occurred during the most recent attempted write to the NVDS array. |

# On-Chip Debugger

The Z8 Encore! XP F082A Series devices contain an integrated On-Chip Debugger (OCD) that provides advanced debugging features including:

- Single pin interface
- Reading and writing of the register file
- Reading and writing of program and data memory
- Setting of breakpoints and watchpoints
- Executing eZ8 CPU instructions
- Debug pin sharing with general-purpose input-output function to maximize pins available to the user (8-pin product only)

## Architecture

The on-chip debugger consists of four primary functional blocks: transmitter, receiver, auto-baud detector/generator and debug controller. Figure 23 displays the architecture of the on-chip debugger.

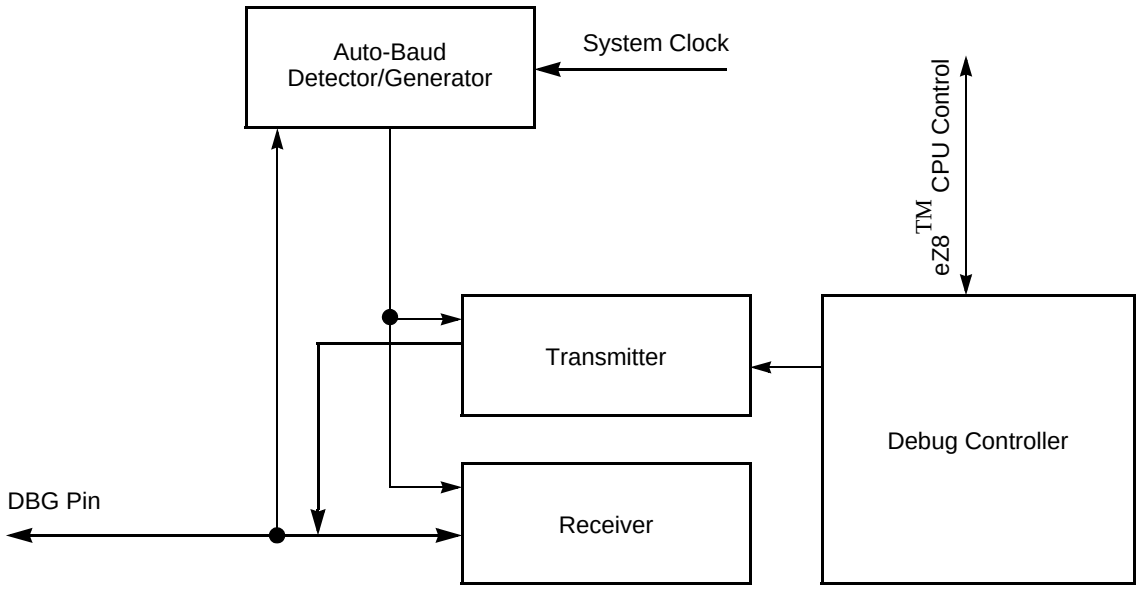


**Figure 23. On-Chip Debugger Block Diagram**

**Read Register (09H).** The Read Register command reads data from the Register File. Data can be read 1–256 bytes at a time (256 bytes can be read by setting size to 0). If the device is not in DEBUG Mode or if the Flash Read Protect Option bit is enabled, this command returns FFH for all the data values.

```
DBG ← 09H
DBG ← {4'h0,Register Address[11:8]
DBG ← Register Address[7:0]
DBG ← Size[7:0]
DBG ⟶ 1-256 data bytes
```

**Write Program Memory (0AH).** The Write Program Memory command writes data to Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be written 1–65536 bytes at a time (65536 bytes can be written by setting size to 0). The on-chip Flash Controller must be written to and unlocked for the programming operation to occur. If the Flash Controller is not unlocked, the data is discarded. If the device is not in DEBUG Mode or if the Flash Read Protect Option bit is enabled, the data is discarded.

```
DBG ← 0AH
DBG ← Program Memory Address[15:8]
DBG ← Program Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG ← 1-65536 data bytes
```

**Read Program Memory (0BH).** The Read Program Memory command reads data from Program Memory. This command is equivalent to the LDC and LDCI instructions. Data can be read 1–65536 bytes at a time (65536 bytes can be read by setting size to 0). If the device is not in DEBUG Mode or if the Flash Read Protect Option bit is enabled, this command returns FFH for the data.

```
DBG ← 0BH
DBG ← Program Memory Address[15:8]
DBG ← Program Memory Address[7:0]
DBG ← Size[15:8]
DBG ← Size[7:0]
DBG ⟶ 1-65536 data bytes
```

**Write Data Memory (0CH).** The Write Data Memory command writes data to Data Memory. This command is equivalent to the LDE and LDEI instructions. Data can be written 1–65536 bytes at a time (65536 bytes can be written by setting size to 0). If the device is not in DEBUG Mode or if the Flash Read Protect Option bit is enabled, the data is discarded.

```
DBG ← 0CH
DBG ← Data Memory Address[15:8]
DBG ← Data Memory Address[7:0]
```

| Bit | Description (Continued) |
|---|---|
| [5]<br>DBGACK | **Debug Acknowledge**<br>This bit enables the debug acknowledge feature. If this bit is set to 1, the OCD sends a Debug Acknowledge character (FFH) to the host when a Breakpoint occurs.<br>0 = Debug Acknowledge is disabled.<br>1 = Debug Acknowledge is enabled. |
| [4:1] | **Reserved**<br>These bits are reserved and must be programmed to 0000. |
| [0]<br>RST | **Reset**<br>Setting this bit to 1 resets the Z8F04xA family device. The device goes through a normal Power-On Reset sequence with the exception that the On-Chip Debugger is not reset. This bit is automatically cleared to 0 at the end of reset.<br>0 = No effect.<br>1 = Reset the Flash Read Protect Option Bit device. |

## OCD Status Register

The OCD Status Register reports status information about the current state of the debugger and the system.

**Table 111. OCD Status Register (OCDSTAT)**

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| Field | DBG | HALT | FRPENB | Reserved | | | | |
| RESET | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| R/W | R | R | R | R | R | R | R | R |

| Bit | Description |
|---|---|
| [7]<br>DBG | **Debug Status**<br>0 = NORMAL Mode.<br>1 = DEBUG Mode. |
| [6]<br>HALT | **HALT Mode**<br>0 = Not in HALT Mode.<br>1 = In HALT Mode. |
| [5]<br>FRPENB | **Flash Read Protect Option Bit Enable**<br>0 = FRP bit enabled, that allows disabling of many OCD commands.<br>1 = FRP bit has no effect. |
| [4:0] | **Reserved**<br>These bits are reserved and must be programmed to 00000. |

# Internal Precision Oscillator

The internal precision oscillator (IPO) is designed for use without external components. You can either manually trim the oscillator for a nonstandard frequency or use the automatic factory-trimmed version to achieve a 5.53MHz frequency. IPO features include:

- On-chip RC oscillator that does not require external components
- Output frequency of either 5.53MHz or 32.8kHz (contains both a fast and a slow mode)
- Trimmed through Flash option bits with user override
- Elimination of crystals or ceramic resonators in applications where very high timing accuracy is not required

## Operation

An 8-bit trimming register, incorporated into the design, compensates for absolute variation of oscillator frequency. Once trimmed the oscillator frequency is stable and does not require subsequent calibration. Trimming is performed during manufacturing and is not necessary for you to repeat unless a frequency other than 5.53MHz (fast mode) or 32.8kHz (slow mode) is required. This trimming is done at +30ºC and a supply voltage of 3.3V, so accuracy of this operating point is optimal.

If not used, the IPO can be disabled by the Oscillator Control Register (see the Oscillator Control Register Definitions section on page 196).

By default, the oscillator frequency is set by the factory trim value stored in the write-protected Flash information page. However, the user code can override these trim values as described in the Trim Bit Address Space section on page 165.

Select one of two frequencies for the oscillator (5.53MHz and 32.8kHz) using the OSCSEL bits in the the Oscillator Control chapter on page 193.

- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

Tables 120 through 127 list the instructions belonging to each group and the number of operands required for each instruction. Some instructions appear in more than one table as these instruction can be considered as a subset of more than one category. Within these tables, the source operand is identified as *src*, the destination operand is *dst* and a condition code is *cc*.

**Table 120. Arithmetic Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ADC | dst, src | Add with Carry |
| ADCX | dst, src | Add with Carry using Extended Addressing |
| ADD | dst, src | Add |
| ADDX | dst, src | Add using Extended Addressing |
| CP | dst, src | Compare |
| CPC | dst, src | Compare with Carry |
| CPCX | dst, src | Compare with Carry using Extended Addressing |
| CPX | dst, src | Compare using Extended Addressing |
| DA | dst | Decimal Adjust |
| DEC | dst | Decrement |
| DECW | dst | Decrement Word |
| INC | dst | Increment |
| INCW | dst | Increment Word |
| MULT | dst | Multiply |
| SBC | dst, src | Subtract with Carry |
| SBCX | dst, src | Subtract with Carry using Extended Addressing |
| SUB | dst, src | Subtract |
| SUBX | dst, src | Subtract using Extended Addressing |

**Table 121. Bit Manipulation Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| BCLR | bit, dst | Bit Clear |
| BIT | p, bit, dst | Bit Set or Clear |
| BSET | bit, dst | Bit Set |
| BSWAP | dst | Bit Swap |
| CCF | — | Complement Carry Flag |
| RCF | — | Reset Carry Flag |
| SCF | — | Set Carry Flag |
| TCM | dst, src | Test Complement Under Mask |
| TCMX | dst, src | Test Complement Under Mask using Extended Addressing |
| TM | dst, src | Test Under Mask |
| TMX | dst, src | Test Under Mask using Extended Addressing |

**Table 122. Block Transfer Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| LDCI | dst, src | Load Constant to/from Program Memory and Auto-Increment Addresses |
| LDEI | dst, src | Load External Data to/from Data Memory and Auto-Increment Addresses |

**Table 123. CPU Control Instructions**

| Mnemonic | Operands | Instruction |
|---|---|---|
| ATM | — | Atomic Execution |
| CCF | — | Complement Carry Flag |
| DI | — | Disable Interrupts |
| EI | — | Enable Interrupts |
| HALT | — | Halt Mode |
| NOP | — | No Operation |

**Table 128. eZ8 CPU Instruction Summary (Continued)**

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

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

# Opcode Maps

A description of the opcode map data and the abbreviations are provided in Figure 30. Figures 31 and 32 display the eZ8 CPU instructions. Table 129 lists Opcode Map abbreviations.

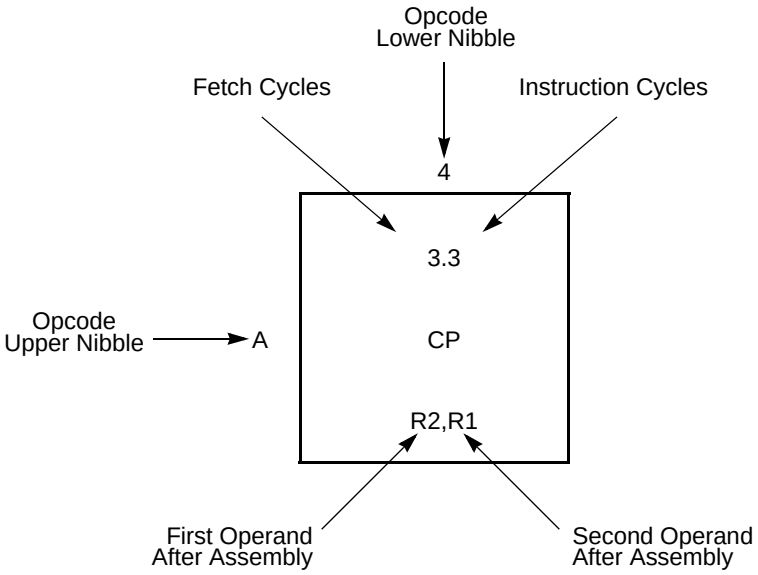


**Figure 30. Opcode Map Cell Description**

## UART Timing

Figure 37 and Table 146 provide timing information for UART pins for the case where CTS is used for flow control. The CTS to DE assertion delay (T1) assumes the Transmit Data Register has been loaded with data prior to CTS assertion.

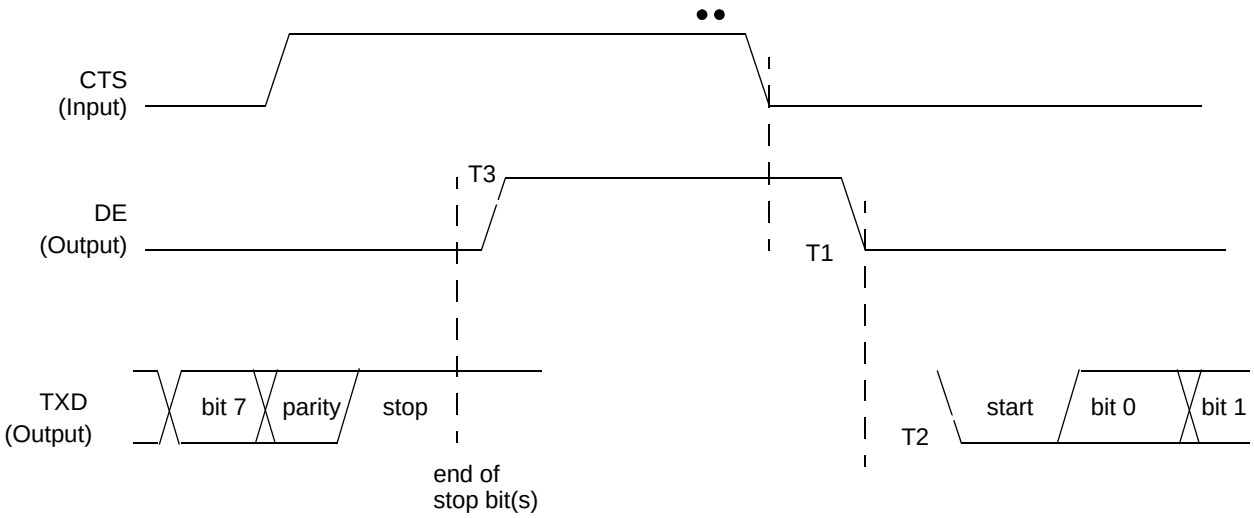


**Figure 37. UART Timing With CTS**

**Table 146. UART Timing With CTS**

| | | Delay (ns) | |
|---|---|---|---|
| **Parameter** | **Abbreviation** | **Minimum** | **Maximum** |
| **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 |