Welcome to E-XFL.COM

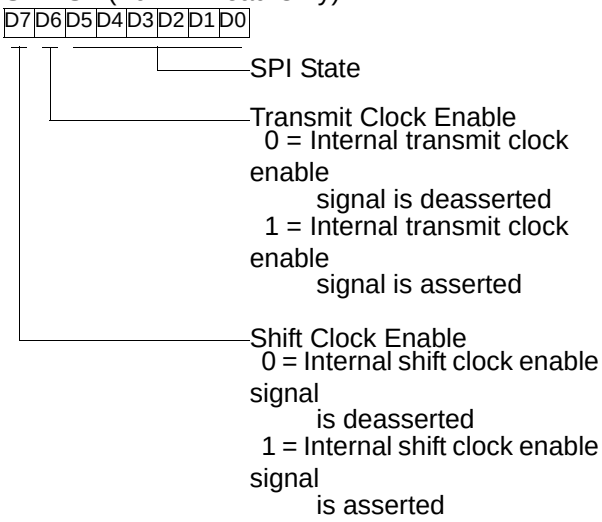
What is "Embedded - Microcontrollers"?

"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.

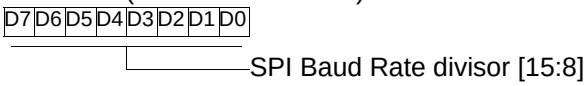
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	11
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 2x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	20-SSOP (0.209", 5.30mm Width)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z8f0821hh020ec

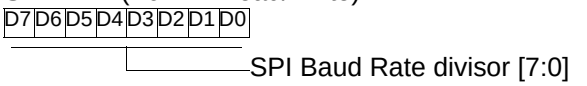
SPI Diagnostic State
SPIDST (F64H - Read Only)



SPI Baud Rate Generator High Byte
SPIBRH (F66H - Read/Write)



SPI Baud Rate Generator Low Byte
SPIBRL (F67H - Read/Write)



IRQ0 Enable Low Bit
IRQ0ENL (FC2H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

- D0: ADC IRQ Enable Hit Bit
- D1: SPI IRQ Enable Low Bit
- D2: I2C IRQ Enable Low Bit
- D3: UART 0 Transmitter IRQ
- D4: UART 0 Receiver IRQ Enable
- D5: Timer 0 IRQ Enable Low Bit
- D6: Timer 1 IRQ Enable Low Bit
- D7: Reserved

Interrupt Request 1
IRQ1 (FC3H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

- D0: Port A Pin Interrupt Request
  0 = IRQ from corresponding pin [7:0] is not pending
  1 = IRQ from corresponding pin [7:0] is awaiting service

IRQ1 Enable High Bit
IRQ1ENH (FC4H - Read/Write)

D7	D6	D5	D4	D3	D2	D1	D0

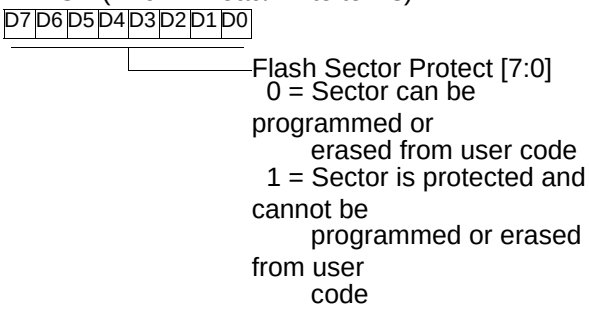
- D0: Port A Pin IRQ Enable High

IRQ1 Enable Low Bit
IRQ1ENL (FC5H - Read/Write)

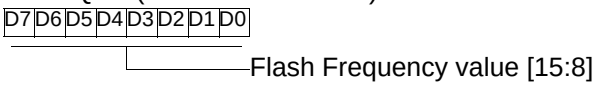
D7	D6	D5	D4	D3	D2	D1	D0

- D0: Port A Pin IRQ Enable Low

Flash Sector Protect
FPROT (FF9H - Read/Write to 1's)

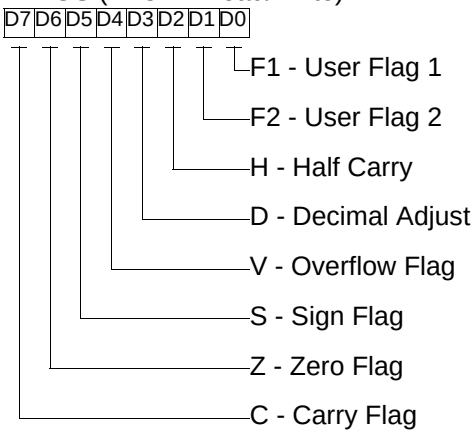


Flash Frequency High Byte
FFREQH (FFAH - Read/Write)

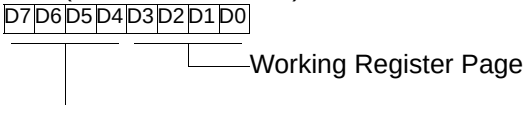


Flash Frequency Low Byte
FFREQL (FFBH - Read/Write)

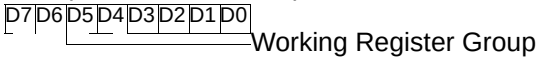
Flags
FLAGS (FFCH - Read/Write)



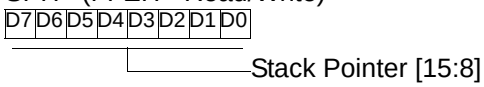
Register Pointer
RP (FFDH- Read/Write)



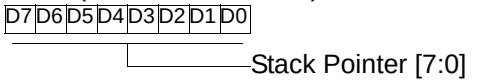
Register Pointer
RP (FFDH- Read/Write)



Stack Pointer High Byte
SPH (FFEH - Read/Write)



Stack Pointer Low Byte
SPL (FFFH - Read/Write)



Reset and Stop Mode Recovery

The Reset Controller within the Z8 Encore! XP® F0822 Series controls Reset and Stop Mode Recovery operation. In typical operation, the following events cause a Reset to occur:

- Power-On Reset (POR)
- Voltage Brownout
- WDT time-out (when configured through 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 Z8 Encore! XP F0822 Series device is in STOP mode, a Stop Mode Recovery is initiated by any of the following events:

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

Reset Types

Z8 Encore! XP F0822 Series provides two 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 Z8 Encore! XP F0822 Series device and the source of the Reset. Table 8 lists the types of Resets and their operating characteristics.

Table 8. Reset and Stop Mode Recovery Characteristics and Latency

	Reset Characteristics and Latency		
Reset Type	**Control Registers**	**eZ8 CPU**	**Reset Latency (Delay)**
System Reset	Reset (as applicable)	Reset	66 WDT Oscillator cycles + 16 System Clock cycles
Stop Mode Recovery	Unaffected, except WDT_CTL register	Reset	66 WDT Oscillator cycles + 16 System Clock cycles

Table 31. IRQ1 Enable and Priority Encoding

IRQ1ENH[*x*]	IRQ1ENL[*x*]	Priority	Description
0	0	Disabled	Disabled
0	1	Level 1	Low
1	0	Level 2	Nominal
1	1	Level 3	High

where *x* indicates the register bits from 0 through 7.

Table 32. IRQ1 Enable High Bit Register (IRQ1ENH)

BITS	7	6	5	4	3	2	1	0
FIELD	PA7ENH	PA6ENH	PA5ENH	PA4ENH	PA3ENH	PA2ENH	PA1ENH	PA0ENH
RESET	0							
R/W	R/W							
ADDR	FC4H							

PA*x*ENH—Port A Bit[*x*] Interrupt Request Enable High Bit

Table 33. IRQ1 Enable Low Bit Register (IRQ1ENL)

BITS	7	6	5	4	3	2	1	0
FIELD	PA7ENL	PA6ENL	PA5ENL	PA4ENL	PA3ENL	PA2ENL	PA1ENL	PA0ENL
RESET	0							
R/W	R/W							
ADDR	FC5H							

PA*x*ENL—Port A Bit[*x*] Interrupt Request Enable Low Bit

IRQ2 Enable High and Low Bit Registers

Table 34 describes the priority control for IRQ2. The IRQ2 Enable High and Low Bit Registers (Table 35 and Table 36) form a priority encoded enabling for interrupts in the Interrupt Request 2 register. Priority is generated by setting bits in each register.

- 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:

$$\text{ONE-SHOT Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value}) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

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.

Follow the steps below for configuring a timer for CONTINUOUS mode and initiating the count:

1. Write to the Timer Control Register to:
   - Disable the timer
   - Configure the timer for CONTINUOUS 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 (usually 0001H). This only affects the first pass in CONTINUOUS mode. After the first timer Reload in CONTINUOUS mode, counting always begins at the reset value of 0001H.
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.

TH and TL—Timer High and Low Bytes
These 2 bytes, {TMRH[7:0], TMRL[7:0]}, contain the current 16-bit timer count value.

Timer Reload High and Low Byte Registers

The Timer 0–1 Reload High and Low Byte (TxRH and TxRL) Registers (Table 41) store a 16-bit reload value, {TRH[7:0], TRL[7:0]}. Values written to the Timer Reload High Byte register are stored in a temporary holding register. When a write to the Timer Reload Low Byte Register occurs, the temporary holding register value is written to the Timer High Byte Register. This operation allows simultaneous updates of the 16-bit Timer Reload value.

In COMPARE mode, the Timer Reload High and Low Byte Registers store the 16-bit Compare value.

Table 41. Timer 0–1 Reload High Byte Register (TxRH)

BITS	7	6	5	4	3	2	1	0
FIELD	TRH							
RESET	1							
R/W	R/W							
ADDR	F02H, F0AH							

Table 42. Timer 0–1 Reload Low Byte Register (TxRL)

BITS	7	6	5	4	3	2	1	0
FIELD	TRL							
RESET	1							
R/W	R/W							
ADDR	F03H, F0BH							

TRH and TRL—Timer Reload Register High and Low
These two bytes form the 16-bit Reload value, {TRH[7:0], TRL[7:0]}. This value sets the maximum count value which initiates a timer reload to 0001H. In COMPARE mode, these two bytes form the 16-bit Compare value.

Timer 0–1 PWM High and Low Byte Registers

The Timer 0–1 PWM High and Low Byte (TxPWMH and TxPWML) registers (Table 43 and Table 44) are used for Pulse-Width Modulator (PWM) operations. These registers also store the Capture values for the CAPTURE and CAPTURE/COMPARE modes.

STOP—Stop Mode Recovery Indicator
If this bit is set to 1, a Stop Mode Recovery occurred. If the STOP and WDT bits are both set to 1, the Stop Mode Recovery occurred due to a WDT time-out. If the STOP bit is 1 and the WDT bit is 0, the Stop Mode Recovery was not caused by a WDT time-out. This bit is reset by a POR or a WDT time-out that occurred while not in STOP mode. Reading this register also resets this bit.

WDT—Watchdog Timer Time-Out Indicator
If this bit is set to 1, a WDT time-out occurred. A POR resets this pin. A Stop Mode Recovery due a change in an input pin also resets this bit. Reading this register resets this bit.

EXT—External Reset Indicator
If this bit is set to 1, a Reset initiated by the external $\overline{\text{RESET}}$ pin occurred. A POR or a Stop Mode Recovery from a change in an input pin resets this bit. Reading this register resets this bit.

Reserved
These bits are reserved and must be 0.

Watchdog Timer Reload Upper, High and Low Byte Registers

The Watchdog Timer Reload Upper, High and Low Byte (WDTU, WDTH, WDTL) Registers (Table 49 through Table 51) form the 24-bit reload value that is loaded into the WDT, when a WDT instruction executes. The 24-bit reload value is {WDTU[7:0], WDTH[7:0], WDTL[7:0]}. Writing to these registers sets the required Reload Value. Reading from these registers returns the current WDT count value.

! Caution: *The 24-bit WDT Reload Value must not be set to a value less than* 000004H.

Table 49. Watchdog Timer Reload Upper Byte Register (WDTU)

BITS	7	6	5	4	3	2	1	0
FIELD	WDTU							
RESET	1							
R/W	R/W*							
ADDR	FF1H							
R/W*—Read returns the current WDT count value. Write sets the desired Reload Value.								

WDTU—WDT Reload Upper Byte
Most significant byte (MSB), Bits[23:16], of the 24-bit WDT reload value.

5. Check the TDRE bit in the UART Status 0 Register to determine if the Transmit Data Register is empty (indicated by a 1). If empty, continue to step 6. If the Transmit Data Register is full (indicated by a 0), continue to monitor the TDRE bit until the Transmit Data Register becomes available to receive new data.
6. Write the UART Control 1 Register to select the outgoing address bit:
   – Set the Multiprocessor Bit Transmitter (MPBT) if sending an address byte, clear it if sending a data byte.
7. Write data byte to the UART Transmit Data Register. The transmitter automatically transfers data to the Transmit Shift Register and then transmits the data.
8. If required, and multiprocessor mode is enabled, make any changes to the Multiprocessor Bit Transmitter (MPBT) value.
9. To transmit additional bytes, return to step 5.

Transmitting Data Using Interrupt-Driven Method

The UART Transmitter interrupt indicates the availability of the Transmit Data Register to accept new data for transmission. Follow the below steps to configure the UART for interrupt-driven data transmission:

1. Write to the UART Baud Rate High and Low Byte Registers to set the required baud rate.
2. Enable the UART pin functions by configuring the associated GPIO Port pins for alternate function operation.
3. Execute a DI instruction to disable interrupts.
4. Write to the Interrupt Control Registers to enable the UART Transmitter interrupt and set the required priority.
5. If MULTIPROCESSOR mode is required, write to the UART Control 1 Register to enable Multiprocessor (9-bit) mode functions:
   – Set the Multiprocessor Mode Select (MPEN) to enable MULTIPROCESSOR mode.
6. Write to the UART Control 0 Register to:
   – Set the transmit enable (TEN) bit to enable the UART for data transmission
   – Enable parity, if required, and if MULTIPROCESSOR mode is not enabled, and select either even or odd parity.
   – Set or clear the CTSE bit to enable or disable control from the remote receiver through the $\overline{\text{CTS}}$ pin.
7. Execute an EI instruction to enable interrupts.

Receiver Interrupts

The receiver generates an interrupt when any of the following occurs:

- A data byte is received and is available in the UART Receive Data Register. This interrupt can be disabled independent of the other receiver interrupt sources. The received data interrupt occurs once the receive character is received and placed in the Receive Data Register. Software must respond to this received data available condition before the next character is completely received to avoid an overrun error. 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
- An overrun is detected
- A data framing error is detected

UART Overrun Errors

When an overrun error condition occurs the UART prevents overwriting of the valid data currently in the Receive Data Register. The break detect and overrun status bits are not displayed until the valid data is read.

After the valid data has been read, the UART 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 should be ignored. The BRKD bit indicates if the overrun was 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 is the UART Status 0 Register. Updates to the Receive Data Register occur only when the next data word is received.

UART Data and Error Handling Procedure

Figure16 on page 99 displays the recommended procedure for UART receiver ISRs.

Baud Rate Generator Interrupts

If the BRG interrupt enable is set, the UART Receiver interrupt asserts when the UART Baud Rate Generator reloads. This action allows the BRG to function as an additional counter if the UART functionality is not employed.

For a given UART data rate, the integer baud rate divisor value is calculated using the following equation:

$$\text{UART Baud Rate Divisor Value (BRG)} = \text{Round}\left(\frac{\text{System Clock Frequency (Hz)}}{16 \times \text{UART Data Rate (bits/s)}}\right)$$

The baud rate error relative to the desired baud rate is calculated using the following equation:

$$\text{UART Baud Rate Error (\%)} = 100 \times \left(\frac{\text{Actual Data Rate} - \text{Desired Data Rate}}{\text{Desired Data Rate}}\right)$$

For reliable communication, the UART baud rate error must never exceed 5 percent. Table 61 provides information on data rate errors for popular baud rates and commonly used crystal oscillator frequencies.

Table 61. UART Baud Rates

10.0 MHz System Clock				5.5296 MHz System Clock			
Desired Rate (kHz)	**BRG Divisor (Decimal)**	**Actual Rate (kHz)**	**Error (%)**	**Desired Rate (kHz)**	**BRG Divisor (Decimal)**	**Actual Rate (kHz)**	**Error (%)**
1250.0	N/A	N/A	N/A	1250.0	N/A	N/A	N/A
625.0	1	625.0	0.00	625.0	N/A	N/A	N/A
250.0	3	208.33	-16.67	250.0	1	345.6	38.24
115.2	5	125.0	8.51	115.2	3	115.2	0.00
57.6	11	56.8	-1.36	57.6	6	57.6	0.00
38.4	16	39.1	1.73	38.4	9	38.4	0.00
19.2	33	18.9	0.16	19.2	18	19.2	0.00
9.60	65	9.62	0.16	9.60	36	9.60	0.00
4.80	130	4.81	0.16	4.80	72	4.80	0.00
2.40	260	2.40	-0.03	2.40	144	2.40	0.00
1.20	521	1.20	-0.03	1.20	288	1.20	0.00
0.60	1042	0.60	-0.03	0.60	576	0.60	0.00
0.30	2083	0.30	0.2	0.30	1152	0.30	0.00

SPI Status Register

The SPI Status Register indicates the current state of the SPI. All bits revert to their reset state if the SPIEN bit in the SPICTL Register equals 0.

Table 65. SPI Status Register (SPISTAT)

BITS	7	6	5	4	3	2	1	0
FIELD	IRQ	OVR	COL	ABT	Reserved		TXST	SLAS
RESET	0							1
R/W	R/W*				R			
ADDR	F62H							
R/W* = Read access. Write a 1 to clear the bit to 0.								

IRQ—Interrupt Request
If SPIEN = 1, this bit is set if the STR bit in the SPICTL Register is set, or upon completion of an SPI Master or Slave transaction. This bit does not set if SPIEN = 0 and the SPI Baud Rate Generator is used as a timer to generate the SPI interrupt.
0 = No SPI interrupt request pending.
1 = SPI interrupt request is pending.

OVR—Overrun
0 = An overrun error has not occurred.
1 = An overrun error has been detected.

COL—Collision
0 = A multi-master collision (mode fault) has not occurred.
1 = A multi-master collision (mode fault) has been detected.

ABT—SLAVE mode transaction abort
This bit is set if the SPI is configured in SLAVE mode, a transaction is occurring and $\overline{SS}$ deasserts before all bits of a character have been transferred as defined by the NUMBITS field of the SPIMODE Register. The IRQ bit also sets, indicating the transaction has completed.
0 = A SLAVE mode transaction abort has not occurred.
1 = A SLAVE mode transaction abort has been detected.

Reserved—Must be 0

TXST—Transmit Status
0 = No data transmission currently in progress.
1 = Data transmission currently in progress.

SLAS—Slave Select
If SPI enabled as a Slave
0 = $\overline{SS}$ input pin is asserted (Low)
1 = $\overline{SS}$ input is not asserted (High).
If SPI enabled as a Master, this bit is not applicable.

BRH = SPI Baud Rate High Byte
Most significant byte, BRG[15:8], of the SPI Baud Rate Generator's reload value.

Table 69. SPI Baud Rate Low Byte Register (SPIBRL)

BITS	7	6	5	4	3	2	1	0
FIELD	BRL							
RESET	1							
R/W	R/W							
ADDR	F67H							

BRL = SPI Baud Rate Low Byte
Least significant byte, BRG[7:0], of the SPI Baud Rate Generator's reload value.

Operation

The I²C Controller operates in MASTER mode to transmit and receive data. Only a single master is supported. Arbitration between two masters must be accomplished in software. I²C supports the following operations:

- Master transmits to a 7-bit Slave
- Master transmits to a 10-bit Slave
- 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 becomes 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. BRG 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 ISR, the NCKI bit should always be checked prior to servicing transmit or receive interrupt conditions because it indicates the transaction is being terminated.

Follow the steps below to setup the Flash Sector Protect Register from user code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write 5EH to the Flash Control Register to select the Flash Sector Protect Register.
3. Read and/or write the Flash Sector Protect Register which is now at Register File address FF9H.
4. Write 00H to the Flash Control Register to return the Flash Controller to its reset state.

Flash Write Protection Option Bit

The Flash Write Protect option bit can block all program and erase operations from user code. For more information, see Option Bits on page 163.

Byte Programming

When the Flash Controller is unlocked, writes to Flash Memory from user code programs a byte into the Flash if the address is located in the unlocked page. An erased Flash byte contains all 1s (FFH). The programming operation is used to change bits from 1 to 0. To change a Flash bit (or multiple bits) from zero to one requires a Page Erase or Mass Erase operation.

Byte Programming is accomplished using the eZ8 CPU's LDC or LDCI instructions. Refer to *eZ8 CPU Core User Manual (UM0128)* for a description of the LDC and LDCI instructions.

While the Flash Controller programs the Flash memory, the eZ8 CPU idles but the system clock and on-chip peripherals continue to operate. Interrupts that occur when a Programming operation is in progress are serviced once the Programming operation is complete. To exit Programming mode and lock the Flash Controller, write 00H to the Flash Control Register.

User code cannot program Flash Memory on a page that is located in a protected sector. When user code writes memory locations, only addresses located in the unlocked page are programmed. Memory writes outside of the unlocked page are ignored.

! Caution: *Each memory location must not be programmed more than twice before an erase occurs.*

Follow the steps below to program the Flash from user code:

1. Write 00H to the Flash Control Register to reset the Flash Controller.
2. Write the page of memory to be programmed to the Page Select Register.
3. Write the first unlock command 73H to the Flash Control Register.
4. Write the second unlock command 8CH to the Flash Control Register.

Reserved
These Option Bits are reserved for future use and must always be 1.

The following information applies only to the Flash versions of the F0822 Series devices:

FWP—Flash Write Protect
These two Option Bits combine to provide three levels of Program Memory protection:

FWP	Description
0	Programming, Page Erase, and Mass Erase using User Code is disabled. Mass Erase is available through the OCD.
1	Programming and Page Erase are enabled for all of Flash Program Memory.

Flash Memory Address 0001H

Table 90. Options Bits at Flash Memory Address 0001H

BITS	7	6	5	4	3	2	1	0
FIELD	Reserved							
RESET	U							
R/W	R/W							
ADDR	Program Memory 0001H							

Note: U = Unchanged by Reset. R/W = Read/Write.

Reserved
These Option Bits are reserved for future use and must always be 1. This setting is the default for unprogrammed (erased) Flash.

RPEN—Read Protect Option Bit Enabled
0 = The Read Protect Option Bit is disabled (1).
1 = The Read Protect Option Bit is enabled (0), disabling many OCD commands.

Reserved. Must be 0

Figure 55 and Table 112 provide timing information for UART pins for the case where the Clear To Send input signal ($\overline{CTS}$) is not used for flow control. In this example, it is assumed that the Driver Enable polarity has been configured to be Active Low and is represented here by $\overline{DE}$. $\overline{DE}$ asserts after the UART Transmit Data Register has been written. $\overline{DE}$ remains asserted for multiple characters as long as the Transmit Data Register is written with the next character before the current character has completed.

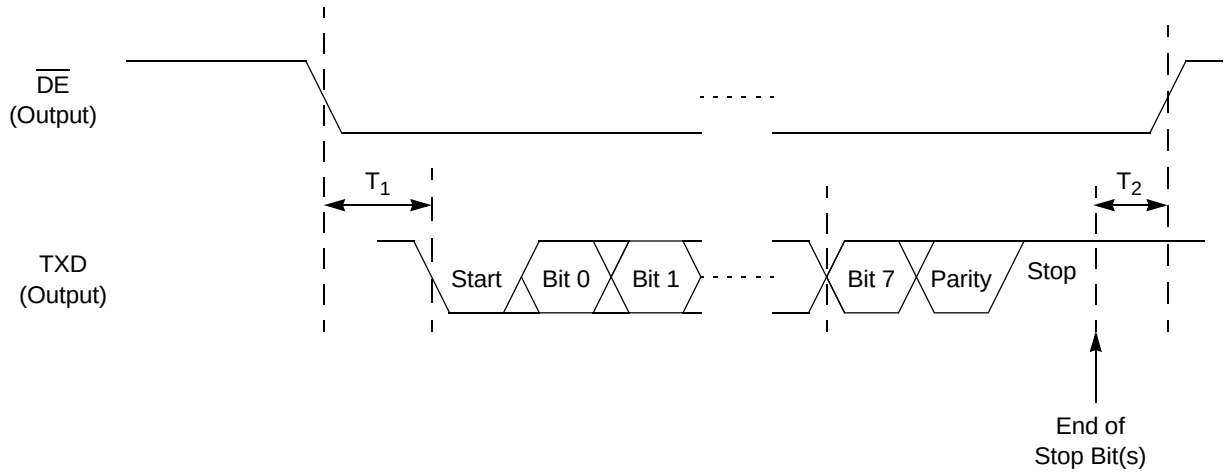


Figure 55. UART Timing without $\overline{CTS}$

Table 112. UART Timing without $\overline{CTS}$

		Delay (ns)	
Parameter	**Abbreviation**	**Minimum**	**Maximum**
T_1	$\overline{DE}$ Assertion to TXD Falling Edge (Start) Delay	1 Bit period	1 Bit period + 1 * XIN period
T_2	End of Stop Bit(s) to $\overline{DE}$ Deassertion Delay	1 * XIN period	2 * XIN period

Table 127. Opcode Map Abbreviations

Abbreviation	Description	Abbreviation	Description
b	Bit position	IRR	Indirect Register Pair
cc	Condition code	p	Polarity (0 or 1)
X	8-bit signed index or displacement	r	4-bit Working Register
DA	Destination address	R	8-bit register
ER	Extended Addressing register	r1, R1, Ir1, Irr1, IR1, rr1, RR1, IRR1, ER1	Destination address
IM	Immediate data value	r2, R2, Ir2, Irr2, IR2, rr2, RR2, IRR2, ER2	Source address
Ir	Indirect Working Register	RA	Relative
IR	Indirect register	rr	Working Register Pair
Irr	Indirect Working Register Pair	RR	Register Pair