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	Active
Core Processor	eZ8
Core Size	8-Bit
Speed	20MHz
Connectivity	I²C, IrDA, LINbus, UART/USART
Peripherals	Brown-out Detect/Reset, LED, LVD, POR, PWM, Temp Sensor, WDT
Number of I/O	17
Program Memory Size	24KB (24K 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 7x10b
Oscillator Type	Internal
Operating Temperature	0°C ~ 70°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/z8f2480hh020sg

Table of Contents

Revision History iii
List of Figures xv
List of Tables xviii
Chapter 1. Overview 1
1.1. Features 1
1.2. Part Selection Guide 2
1.3. Block Diagram 3
1.4. An Overview of the eZ8 CPU and its Peripherals 4
1.4.1. General-Purpose Input/Output 4
1.4.2. Flash Controller 4
1.4.3. Non-Volatile Data Storage 5
1.4.4. Internal Precision Oscillator 5
1.4.5. Crystal Oscillator 5
1.4.6. Secondary Oscillator 5
1.4.7. 10-Bit Analog-to-Digital Converter 5
1.4.8. Low-Power Operational Amplifier 5
1.4.9. Analog Comparator 5
1.4.10. Temperature Sensor 6
1.4.11. Low-Voltage Detector 6
1.4.12. Enhanced SPI 6
1.4.13. UART with LIN 6
1.4.14. Master/Slave I2C 6
1.4.15. Timers 6
1.4.16. Multi-Channel Timer 7
1.4.17. Interrupt Controller 7
1.4.18. Reset Controller 7
1.4.19. On-Chip Debugger 7
1.4.20. Direct LED Drive 7
1.5. Acronyms and Expansions 8
Chapter 2. Pin Description 10
2.1. Available Packages 10
2.2. Pin Configurations 10
2.3. Signal Descriptions 14
2.4. Pin Characteristics 17

Figure 56. Target OCD Connector Interface 296
Figure 57. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #1 of 2 296
Figure 58. Interfacing the On-Chip Debugger's DBG Pin with an RS-232 Interface, #2 of 2 297
Figure 59. OCD Data Format 298
Figure 60. Synchronous Operation 300
Figure 61. Start Bit Flow Control 301
Figure 62. Recommended 20MHz Crystal Oscillator Configuration 322
Figure 63. Connecting the On-Chip Oscillator to an External RC Network 323
Figure 64. Typical RC Oscillator Frequency as a Function of External Capacitance 324
Figure 65. Recommended 32kHz Crystal Oscillator Configuration 325
Figure 66. Op Code Map Cell Description 345
Figure 67. First Op Code Map 347
Figure 68. Second Op Code Map after 1FH 348
Figure 69. Typical Active Flash Mode Supply Current (1–20MHz) 353
Figure 70. Typical Active PRAM Mode Supply Current (1–20MHz) 354
Figure 71. Typical Active Flash Mode Supply Current (32–900kHz) 354
Figure 72. Typical Active PRAM Mode Supply Current (32–900kHz) 355
Figure 73. STOP Mode Current Consumption as a Function of V_{DD} with Temperature as a Parameter; all Peripherals Disabled 356
Figure 74. VDD Versus Maximum System Clock Frequency 357
Figure 75. Port Input Sample Timing 366
Figure 76. GPIO Port Output Timing 367
Figure 77. On-Chip Debugger Timing 368
Figure 78. UART Timing With CTS 369
Figure 79. UART Timing Without CTS 370

Chapter 3. Address Space

The eZ8 CPU can access the following three distinct address spaces:

- The Register File contains addresses for general-purpose registers, eZ8 CPU, peripherals and GPIO port control registers
- The Program Memory contains addresses for all memory locations having executable code and/or data
- The Data Memory contains addresses for all memory locations that contain data only

These three address spaces are covered briefly in the following sections. For more details about the eZ8 CPU and its address space, refer to the eZ8 CPU Core User Manual (UM0128), available for download at www.zilog.com.

3.1. Register File

The Register File address space in the Z8 Encore!® MCU is 4KB (4096 bytes). The Register File is composed of two sections: control registers and general-purpose registers. When instructions are executed, registers defined as sources are read and registers defined as destinations are written. The architecture of the eZ8 CPU allows all general-purpose registers to function as accumulators, address pointers, index registers, stack areas, or scratch pad memory.

The upper 256 bytes of the 4KB Register File address space are reserved for control of the eZ8 CPU, on-chip peripherals and the input/output ports. These registers are located at addresses F00H to FFFH. Some of the addresses within the 256 B control register sections are reserved (that is, unavailable). Reading from a reserved Register File address returns an undefined value. Zilog does not recommend writing to the reserved Register File addresses because doing so can produce unpredictable results.

The on-chip Register RAM always begins at address 000H in the Register File address space. The F1680 Series MCU contains 1KB or 2KB of on-chip Register RAM. Reading from Register File addresses outside the available RAM addresses (and not within the control register address space) returns an undefined value. Writing to these Register File addresses produces no effect.

In addition, the F1680 Series MCU contains 1KB of on-chip Program RAM. Normally it is used as Program RAM and is present in the Program Memory address space (see the Program Memory section on page 20). However, it can also be used as additional Register RAM present in the Register File address space 800H–BFFH (1KB Program RAM, 2KB Register RAM), or 400H–7FFH (1KB Program RAM, 1KB Register RAM), if you do not

Table 19. Port Alternate Function Mapping, 40-/44-Pin Parts[1,2] (Continued)

Port	Pin	Mnemonic	Alternate Function Description	Alternate Function Set Register AFS1
Port B	PB0	Reserved		AFS1[0]: 0
		ANA0/AMPOUT	ADC Analog Input/LPO Output	AFS1[0]: 1
	PB1	Reserved		AFS1[1]: 0
		ANA1/AMPINN	ADC Analog Input/LPO Input (N)	AFS1[1]: 1
	PB2	Reserved		AFS1[2]: 0
		ANA2/AMPINP	ADC Analog Input/LPO Input (P)	AFS1[2]: 1
	PB3	CLKIN	External Clock Input	AFS1[3]: 0
		ANA3	ADC Analog Input	AFS1[3]: 1
	PB4	Reserved		AFS1[4]: 0
		ANA7	ADC Analog Input	AFS1[4]: 1
	PB5	Reserved		AFS1[5]: 0
		VREF	Voltage Reference	AFS1[5]: 1

Notes:
1. Because there are at most two choices of alternate functions for some pins in Ports A–C, the Alternate Function Set Register (AFS2) is implemented but not used to select the function. The alternate function selection must also be enabled, as described in the Port A–E Alternate Function Subregisters section on page 61.
2. Because there is only one alternate function for each Port D and Port E pin, the Alternate Function Set registers are not implemented for Ports D and E. Enabling the alternate function selections automatically enables the associated alternate function, as described in the Port A–E Alternate Function Subregisters section on page 61.
3. This timer function is only available in the 44-pin package; its alternate functions are reserved in the 40-pin package.

7.11.12. Port A–E Output Data Register

The Port A–E Output Data Register, shown in Table 32, controls the output data to the pins.

Table 32. Port A–E 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	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	FD3H, FD7H, FDBH, FDFH, FE3H							

Bit	Description
[7:0] POUT	**Port Output Data** These bits contain the data to be driven to 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.

7.11.13. LED Drive Enable Register

The LED Drive Enable Register, shown in Table 33, activates the controlled current drive. The Port C pin must first be enabled by setting the Alternate Function Register to select the LED function.

Table 33. LED Drive Enable (LEDEN)

Bits	7	6	5	4	3	2	1	0
Field	LEDEN[7:0]							
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	F82H							

Bit	Description
[7:0] LEDEN	**LED Drive Enable** These bits determine which Port C pins are connected to an internal current sink. 0 = Tristate the Port C pin. 1 = Connect controlled current synch to Port C pin.

Table 48. IRQ2 Enable Low Bit Register (IRQ2ENL)

Bits	7	6	5	4	3	2	1	0
Field	Reserved	MCTENL	U1RENL	U1TENL	C3ENL	C2ENL	C1ENL	C0ENL
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	FC8H							

Bit	Description
[7]	Reserved; must be 0.
[6] MCTENL	Multi-Channel Timer Interrupt Request Enable Low Bit (MCTENL)
[5] U1RENL	UART1 Receive Interrupt Request Enable Low Bit (U1RENL)
[4] U1TENL	UART1 Transmit Interrupt Request Enable Low Bit (U1TENL)
[3] C3ENL	Port C3 Interrupt Request Enable Low Bit (C3ENL)
[2] C2ENL	Port C2 Interrupt Request Enable Low Bit (C2ENL)
[1] C1ENL	Port C1 Interrupt Request Enable Low Bit (C1ENL)
[0] C0ENL	Port C0 Interrupt Request Enable Low Bit (C0ENL)

3. Write to the Timer Control 0 Register to set the timer interrupt configuration field TICONFIG.
4. Write to the Timer High and Low Byte registers to set the starting count value. This value only affects the first pass in COUNTER Mode. After the first timer reload in COUNTER Mode, counting always begins at the reset value of 0001H. Generally, in COUNTER Mode the Timer High and Low Byte registers must be written with the value 0001H.
5. Write to the Timer Reload High and Low Byte registers to set the reload value.
6. If required, 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 Input alternate function.
8. When using the Timer Output function, configure the associated GPIO port pin for the Timer Output alternate function.
9. Write to the Timer Control 1 Register to enable the timer.

In COUNTER Mode, the number of Timer Input transitions since the timer start is calculated using the following equation:

COUNTER Mode Timer Input Transitions = Current Count Value - Start Value

9.2.3.5. COMPARATOR COUNTER Mode

In COMPARATOR COUNTER Mode, the timer counts output transitions from an analog comparator output. The assignment of a comparator to a timer is based on the TIMTRG bits in the CMP0 and CMP1 registers. The TPOL bit in the Timer Control 1 Register selects whether the count occurs on the rising edge or the falling edge of the comparator output signal. In COMPARATOR COUNTER Mode, the prescaler is disabled.

! Caution: The frequency of the comparator output signal must not exceed one-fourth the timer clock frequency.

Upon reaching the reload value stored in the Timer Reload High and Low Byte registers, 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 High to Low) at timer reload.

Observe the following steps to configure a timer for COMPARATOR COUNTER Mode and initiate the count:

9.3.5. Timer 0–2 Control Registers

The Timer Control registers are described in Tables 63 through 65.

9.3.5.1. Timer 0–2 Control 0 Register

The Timer 0–2 Control 0 (TxCTL0) register together with TxCTL1 register determines the timer operating mode. It also includes a programmable PWM deadband delay, two bits to configure timer interrupt definition and a status bit to identify if the last timer interrupt is due to an input capture event.

Table 63. Timer 0–2 Control 0 Register (TxCTL0)

Bit	7	6	5	4	3	2	1	0
Field	TMODE[3]	TICONFIG		CSC	PWMD			INPCAP
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	F06H, F0EH, F16H							

Bit	Description
[7] TMODE[3]	**Timer Mode High Bit** This bit, along with the TMODE[2:0] field in the TxCTL1 Register, determines the operating mode of the timer. This bit is the most significant bit of the timer mode selection value. For more details, see the description of the Timer 0–2 Control 1 Register (TxCTL1) on page 113.
[6:5] TICONFIG	**Timer Interrupt Configuration** This field configures timer interrupt definition. 0x = Timer Interrupt occurs on all defined Reload, Compare and Input Events. 10 = Timer Interrupt only on defined Input Capture/Deassertion Events. 11 = Timer Interrupt only on defined Reload/Compare Events.
[4] CSC	**Cascade Timers** 0 = Timer Input signal comes from the pin. 1 = For Timer 0, Input signal is connected to Timer 2 output. For Timer 1, Input signal is connected to Timer 0 output. For Timer 2, Input signal is connected to Timer 1 output.

10.7.2. Multi-Channel Timer High and Low Byte Registers

The High and Low Byte (MCTH and MCTL) registers, shown in Table 70, contain the current 16-bit Multi-Channel Timer count value.

Zilog does not recommend writing to the Multi-Channel Timer High and Low Byte registers while the Multi-Channel Timer is enabled. If either or both of the Multi-Channel Timer High or Low Byte registers are written during counting, the 8-bit written value is placed in the counter (High and/or Low byte) at the next system clock edge. The counter continues counting from the new value.

Table 70. Multi-Channel Timer High and Low Byte Registers (MCTH, MCTL)

Bit	7	6	5	4	3	2	1	0
Field	MCTH							
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	FA0H							

Bit	7	6	5	4	3	2	1	0
Field	MCTL							
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	FA1H							

Bit	Description
[7:0] MCTH, MCTL	**Multi-Channel Timer High and Low Byte** These bytes contain the current 16-bit Multi-Channel Timer count value, {MCTH[7:0], MCTL[7:0]}.

When the Multi-Channel Timer is enabled, a read from MCTH causes the value in MCTL to be stored in a temporary holding register. A read from MCTL returns this temporary register when the Multi-Channel Timer is enabled. When the Multi-Channel Timer is disabled, reads from MCTL read the register directory. The Multi-Channel Timer High and Low Byte registers are not reset when TEN = 0.

10.7.3. Multi-Channel Timer Reload High and Low Byte Registers

The Multi-Channel Timer Reload High and Low Byte (MCTRH and MCTRL) registerss, shown in Table 71, store a 16-bit reload value, {MCTRH[7:0], MCTRL[7:0]}. When TEN = 0, writes to this address update the register on the next clock cycle. When TEN = 1,

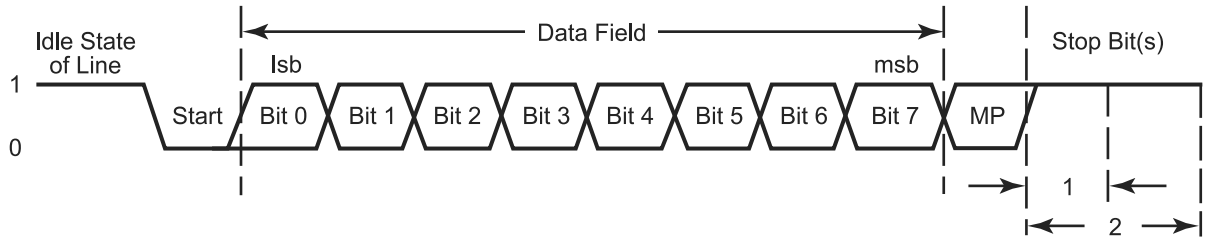

Figure 23. LIN-UART Asynchronous MULTIPROCESSOR Mode Data Format

In MULTIPROCESSOR (9-bit) Mode, the Parity bit location (9^{th} bit) becomes the Multiprocessor control bit. The LIN-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 LIN-UART Address Compare register holds the network address of the device.

12.1.9.1. MULTIPROCESSOR Mode Receive Interrupts

When MULTIPROCESSOR (9-bit) Mode is enabled, the LIN-UART processes only frames addressed to it. To determine whether a frame of data is addressed to the LIN-UART can be made in hardware, software or a combination of the two, depending on the multiprocessor configuration bits. In general, the address compare feature reduces the load on the CPU, because it is not required to access the LIN-UART when it receives data directed to other devices on the multinode 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 LIN-UART Control 1 Register. For all MULTIPROCESSOR Modes, bit MPEN of the LIN-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 checks the address byte which triggered the interrupt. If it matches the LIN-UART address, the software clears MPMD[0]. At this point, each new incoming byte interrupts the CPU. The software determines the end of the frame and checks for it by reading the MPRX bit of the LIN-UART Status 1 Register for each incoming byte. If MPRX = 1, a new frame begins. If the address of this new frame is different from the LIN-UART's address, then MPMD[0] must be set to 1 by software, causing the LIN-UART

In LIN mode, the interrupts defined for normal UART operation still apply with the following changes:

- A Parity Error (the PE bit in the Status 0 Register) is redefined as the Physical Layer Error (PLE) bit. The PLE bit indicates that receive data does not match transmit data when the LIN-UART is transmitting. This definition applies to both Master and Slave operating modes.
- The Break Detect interrupt (BRKD bit in Status 0 Register) indicates when a Break is detected by the slave (break condition for at least 11 bit times). Software can use this interrupt to start a timer checking for message frame time-out. The duration of the break can be read in the RxBreakLength[3:0] field of the Mode Select and Status Register.
- The Break Detect interrupt (BRKD bit in Status 0 Register) indicates when a wake-up message has been received, if the LIN-UART is in LIN SLEEP state.
- In LIN SLAVE Mode, if the BRG counter overflows while measuring the autobaud period (from the start bit to the beginning of bit 7 of the autobaud character), an Overrun Error is indicated (OE bit in the Status 0 Register). In this case, software sets the LINSTATE field back to `10b`, where the slave ignores the current message and waits for the next break. The Baud Reload High and Low registers are not updated by hardware if this autobaud error occurs. The OE bit is also set if a data overrun error occurs.

12.1.10.1. LIN System Clock Requirements

The LIN Master provides the timing reference for the LIN network and is required to have a clock source with a tolerance of ±0.5%. A slave with autobaud capability is required to have a baud clock matching the master oscillator within ±14%. The slave nodes autobaud to lock onto the master timing reference with an accuracy of ±2%. If a slave does not contain autobaud capability, it must include a baud clock which deviates from the masters by not more than ±1.5%. These accuracy requirements must include the effects such as voltage and temperature drift during operation.

Before sending/receiving messages, the Baud Reload High/Low registers must be initialized. Unlike standard UART modes, the Baud Reload High/Low registers must be loaded with the baud interval rather than 1/16 of the baud interval.

In order to autobaud with the required accuracy, the LIN SLAVE system clock must be at least 100 times the baud rate.

12.1.10.2. LIN Mode Initialization and Operation

The LIN protocol mode is selected by setting either the LIN Master (LMST) or LIN SLAVE (LSLV) and optionally (for the LIN SLAVE) the Autobaud Enable (ABEN) bits in the LIN Control Register. To access the LIN Control Register, the Mode Select (MSEL) field of the LIN-UART Mode Select/Status Register must be equal to `010B`. The LIN-

- Noise Filter Control (NFCTL[2:0]) input selects the width of the up/down saturating counter digital filter; the available width ranges from 4 to 11 bits
- The digital filter output features hysteresis
- Provides an active low *Saturated State* output (FiltSatB) which is used as an indication of the presence of noise

12.2.1. Architecture

Figure 25 displays how the noise filter is integrated with the LIN-UART on a LIN network.

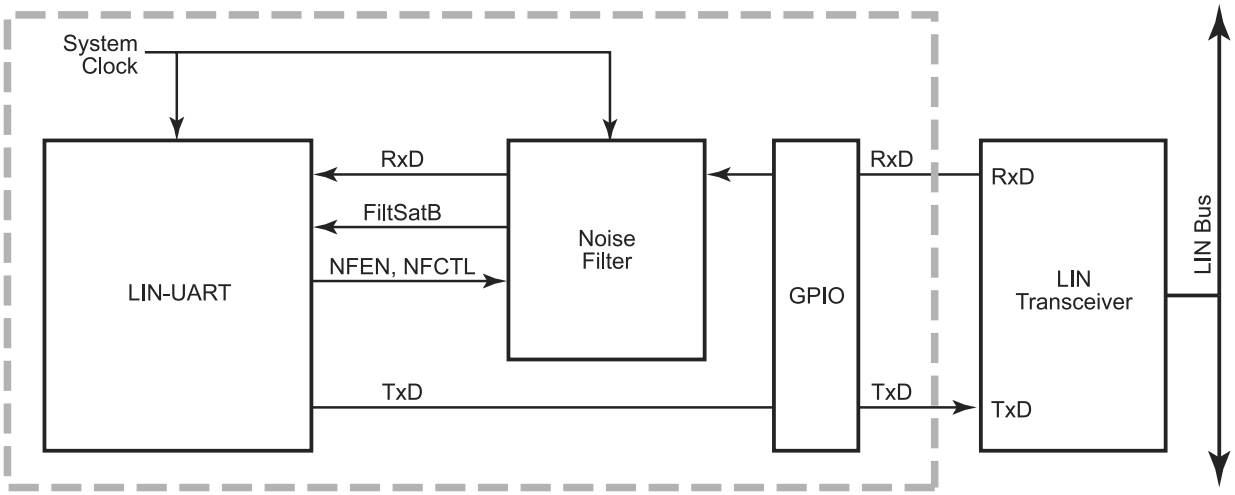

Figure 25. Noise Filter System Block Diagram

12.2.2. Operation

Figure 26 displays the operation of the noise filter both with and without noise. The noise filter in this example is a 2-bit up/down counter which saturates at `00b` and `11b`. A 2-bit counter is shown 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 `01b` to `00b`; and switches from 0 to 1, when the counter counts up from `10b` to `11b`. The noise filter delays the receive data by three System Clock cycles.

The FiltSatB signal is checked when the filtered RxD is sampled in the center of the bit time. The presence of noise (FiltSatB = 1 at center of bit time) does not mean that the sampled data is incorrect, but just that the filter is not in its ‘saturated’ state of all 1s or all 0s. If FiltSatB = 1 then RxD is sampled during a receive character, the NE bit in the

12.3.5. LIN-UART Control 0 Register

The LIN-UART Control 0 Register, shown in Table 89, configures the basic properties of LIN-UART's transmit and receive operations. A more detailed discussion of each bit follows the table.

Table 89. LIN-UART Control 0 Register (U0CTL0 = F42H)

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, F4AH							

Note: R/W = Read/Write.

Bit	Description
[7] TEN	**Transmit Enable** This bit enables or disables the transmitter. The enable is also controlled by the $\overline{CTS}$ signal and the CTSE bit. If the $\overline{CTS}$ signal is Low and the CTSE bit is 1, the transmitter is enabled. 0 = Transmitter disabled. 1 = Transmitter enabled.
[6] REN	**Receive Enable** This bit enables or disables the receiver. 0 = Receiver disabled. 1 = Receiver enabled.
[5] CTSE	**Clear To Send Enable** 0 = The $\overline{CTS}$ signal has no effect on the transmitter. 1 = The LIN-UART recognizes the $\overline{CTS}$ signal as an enable control for the transmitter.
[4] PEN	**Parity Enable** This bit enables or disables parity. Even or odd is determined by the PSEL bit. 0 = Parity is disabled. This bit is overridden by the MPEN bit. 1 = The transmitter sends data with an additional parity bit and the receiver receives an additional parity bit.
[3] PSEL	**Parity Select** 0 = Even parity is sent as an additional parity bit for the transmitter/receiver. 1 = Odd parity is sent as an additional parity bit for the transmitter/receiver.

SPI BRG Interrupt Interval (s) = System Clock Period (s) × BRG[15:0]

16.4. ESPI Control Register Definitions

This section defines the features of the following ESPI Control registers.

ESPI Data Register: see page 213

ESPI Transmit Data Command and Receive Data Buffer Control Register: see page 214

ESPI Control Register: see page 215

ESPI Mode Register: see page 217

ESPI Status Register: see page 219

ESPI State Register: see page 220

ESPI Baud Rate High and Low Byte Registers: see page 221

16.4.1. ESPI Data Register

The ESPI Data Register, shown in Table 109, addresses both the outgoing Transmit Data Register and the incoming Receive Data Register. Reads from the ESPI Data Register return the contents of the Receive Data Register. The Receive Data Register is updated with the contents of the Shift Register at the end of each transfer. Writes to the ESPI Data Register load the Transmit Data Register unless TDRE = 0. Data is shifted out starting with bit 7. The last bit received resides in bit position 0.

With the ESPI configured as a Master, writing a data byte to this register initiates the data transmission. With the ESPI configured as a Slave, writing a data byte to this register loads the Shift Register in preparation for the next data transfer with the external Master. In either the MASTER or SLAVE modes, if TDRE = 0, writes to this register are ignored.

When the character length is less than 8 bits (as set by the NUMBITS field in the ESPI Mode Register), the transmit character must be left justified in the ESPI Data Register. A received character of less than 8 bits is right justified (last bit received is in bit position 0). For example, if the ESPI is configured for 4-bit characters, the transmit characters must be written to ESPIDATA[7:4] and the received characters are read from ESPIDATA[3:0].

Bit	Description (Continued)
[4:2] NUMBITS[2:0]	**Number of Data Bits Per Character to Transfer** This field contains the number of bits to shift for each character transfer. For information about valid bit positions when the character length is less than 8 bits, see the description of the ESPI Data Register on page 213. 000 = 8 bits 001 = 1 bit 010 = 2 bits 011 = 3 bits 100 = 4 bits 101 = 5 bits 110 = 6 bits 111 = 7 bits
[1] SSIO	**Slave Select I/O** This bit controls the direction of the $\overline{SS}$ pin. In single MASTER Mode, SSIO is set to 1 unless a separate GPIO pin is being used to provide the $\overline{SS}$ output function. In the SPI Slave or multi-Master configuration, SSIO is set to 0. 0 = $\overline{SS}$ pin configured as an input (SPI SLAVE and MULTI-MASTER modes). 1 = $\overline{SS}$ pin configured as an output (SPI SINGLE MASTER Mode).
[0] SSPO	**Slave Select Polarity** This bit controls the polarity of the $\overline{SS}$ pin. 0 = $\overline{SS}$ is active Low. (SSV = 1 corresponds to $\overline{SS}$ = 0). 1 = $\overline{SS}$ is active High. (SSV = 1 corresponds to $\overline{SS}$ = 1).

Table 115 defines the valid ESPI states.

Table 115. ESPISTATE Values

ESPISTATE Value	Description
00_0000	Idle
00_0001	Slave Wait For SCK
01_0001	Master Ready
10_1110	Bit 7 Receive
10_1111	Bit 7 Transmit
10_1100	Bit 6 Receive
10_1101	Bit 6 Transmit
10_1010	Bit 5 Receive
10_1011	Bit 5 Transmit
10_1000	Bit 4 Receive
10_1001	Bit 4 Transmit
10_0110	Bit 3 Receive
10_0111	Bit 3 Transmit
10_0100	Bit 2 Receive
10_0101	Bit 2 Transmit
10_0010	Bit 1 Receive
10_0011	Bit 1 Transmit
10_0000	Bit 0 Receive
10_0001	Bit 0 Transmit

16.4.7. ESPI Baud Rate High and Low Byte Registers

The ESPI Baud Rate High and Low Byte registers, shown in Tables 116 and 117, combine to form a 16-bit reload value, BRG[15:0], for the ESPI Baud Rate Generator. The ESPI baud rate is calculated using the following equation:

$$\text{SPI Baud Rate (bits § s)} = \frac{\text{System Clock Frequency (Hz)}}{2 \times \text{BRG[15:0]}}$$

The minimum baud rate is obtained by setting BRG[15:0] to 0000H for a clock divisor value of (2 x 65536 = 131072).

S Start
W Write
A Acknowledge
$\overline{A}$ Not Acknowledge
P Stop

17.2.6.5. Slave Receive Transaction with 7-Bit Address

The data transfer format for writing data from a Master to a Slave in 7-bit address mode is displayed in Figure 47. The procedure that follows describes the I^2C Master/Slave Controller operating as a slave in 7-bit addressing mode and receiving data from the bus master.

S	Slave Address	W=0	A	Data	A	Data	A	Data	A/$\overline{A}$	P/S

Figure 47. Data Transfer Format—Slave Receive Transaction with 7-Bit Address

1. The software configures the controller for operation as a slave in 7-bit addressing mode, as follows:
   a. Initialize the MODE field in the I^2C Mode Register for either SLAVE ONLY mode or MASTER/SLAVE Mode with 7-bit addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[6:0] bits in the I^2C Slave Address Register.
   d. Set IEN = 1 in the I^2C Control Register. Set NAK = 0 in the I^2C Control Register.
2. The bus master initiates a transfer, sending the address byte. In SLAVE Mode, the I^2C controller recognizes its own address and detects that R/$\overline{W}$ bit = 0 (written from the master to the slave). The I^2C controller acknowledges indicating it is available to accept the transaction. The SAM bit in the I2CISTAT Register is set to 1, causing an interrupt. The RD bit in the I2CISTAT Register is cleared to 0, indicating a Write to the slave. The I^2C controller holds the SCL signal Low waiting for the software to load the first data byte.
3. The software responds to the interrupt by reading the I2CISTAT Register (which clears the SAM bit). After seeing the SAM bit to 1, the software checks the RD bit. Because RD = 0, no immediate action is required until the first byte of data is received. If software is only able to accept a single byte, it sets the NAK bit in the I2CCTL Register at this time.
4. The Master detects the Acknowledge and sends the byte of data.

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 177 contains additional symbols that are used throughout the eZ8 CPU Instruction Summary section on page 336.

Table 177. Additional Symbols

Symbol	Definition
dst	Destination Operand
src	Source Operand
@	Indirect Address Prefix
SP	Stack Pointer
PC	Program Counter
FLAGS	Flags Register
RP	Register Pointer
#	Immediate Operand Prefix
B	Binary Number Suffix
%	Hexadecimal Number Prefix
H	Hexadecimal Number Suffix

Assignment of a value is indicated by an arrow. For example, the statement:

dst ← dst + src

indicates that the source data is added to the destination data and the result is stored in the destination location.

27.4. eZ8 CPU Instruction Classes

eZ8 CPU instructions is divided functionally into the following groups:

- Arithmetic
- Bit Manipulation
- Block Transfer
- CPU Control
- Load
- Logical
- Program Control
- Rotate and Shift

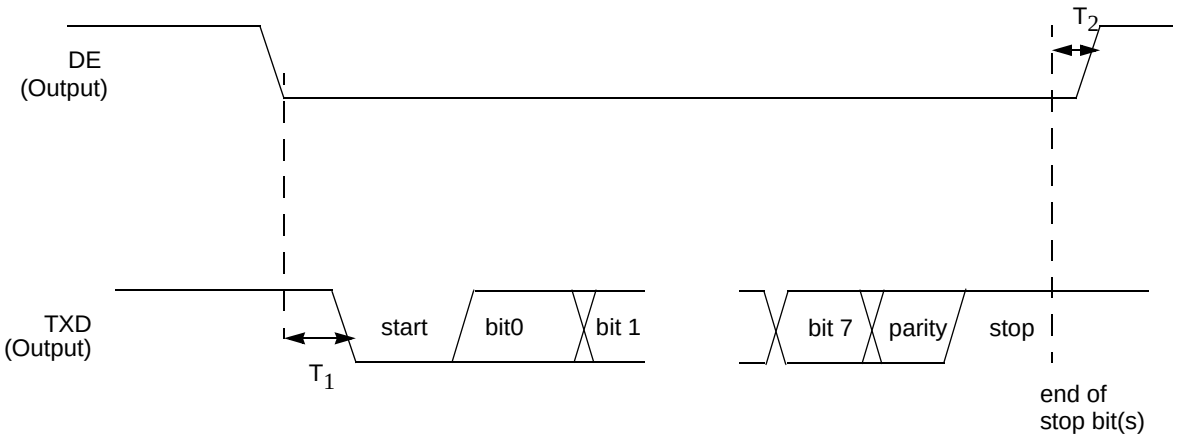

Figure 79. UART Timing Without CTS

Table 208. UART Timing Without CTS

		Delay (ns)	
Parameter	**Abbreviation**	**Min**	**Max**
UART			
T_1	DE assertion to TXD falling edge (start bit) delay	1 * X_{IN} period	1 bit time
T_2	End of stop bit(s) to DE deassertion delay (Tx Data Register is empty)	± 5	