Zilog - Z16F2811FI20EG Datasheet

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	ZNEO
Core Size	16-Bit
Speed	20MHz
Connectivity	EBI/EMI, I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	60
Program Memory Size	128KB (128K x 8)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2.7V ~ 3.6V
Data Converters	A/D 12x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 105°C (TA)
Mounting Type	Surface Mount
Package / Case	80-BQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z16f2811fi20eg

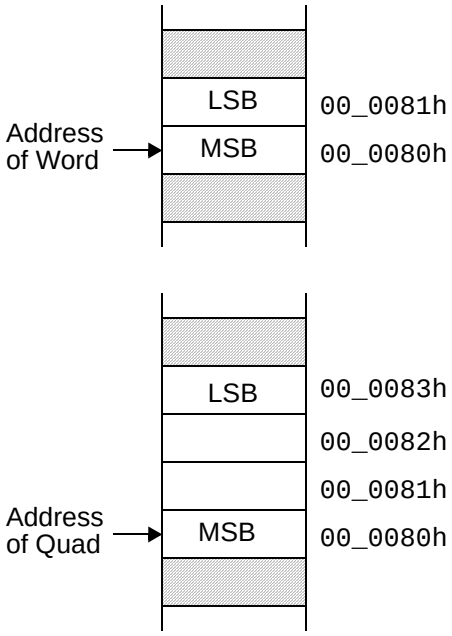


Figure 7. Endianness of Words and Quads

Bus Widths

The ZNEO CPU accesses 8-bit or 16-bit memories. The data buses of the internal nonvolatile memory and internal RAM are 16-bit wide. The internal peripherals are a mix of 8-bit and 16-bit peripherals. The external memory bus is configured as an 8-bit or 16-bit memory bus.

If a Word or Quad operation occurs on a 16-bit wide memory, the number of memory accesses depends on the alignment of the address. If the address is aligned on an even boundary, a Word operation takes one memory access and a Quad operation takes two memory accesses. If the address is on an odd boundary (unaligned), a Word operation takes two memory accesses and a Quad operation takes three memory accesses. Figure 8 displays the alignment Word and Quad operations on 16-bit memories.

Table 11 lists the External Chip Select Control Registers Low for $\overline{CS0}$ (EXTCS0L). This register sets the number of wait states for chip select 0. Waits are only added if the chip select is enabled. Chip Select 0 is enabled automatically in ROMLESS Mode.

Table 11. External Chip Select Control Registers Low for $\overline{CS0}$ (EXTCS0L)

Bits	7	6	5	4	3	2	1	0
Field	RESERVED		PR0WAIT		CS0WAIT			
RESET	0	0	1	1	1	1	1	1
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Addr	FF_(E073)H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:4] PR0WAIT[2:0]	**Post Read Wait Selection** 00 = 0 wait state. 01 = 1 wait state. 10 = 2 wait states. 11 = 3 wait states.
[3:0] CS0WAIT	**Chip Select 0 Wait Selection** 0000 = 0 wait state. 0001 = 1 wait state. 0010 = 2 wait states. 0011 = 3 wait states. 0100 = 4 wait states. 0101 = 5 wait states. 0110 = 6 wait states. 0111 = 7 wait states. 1000 = 8 wait states. 1001 = 9 wait states. 1010 = 10 wait state.s 1011 = 11 wait states. 1100 = 12 wait states. 1101 = 13 wait states. 1110 = 14 wait states. 1111 = 15 wait states.

Figure 14 and Table 16 provide timing information for the External Interface performing a read operation in Normal Mode with a post read wait state. The configuration is the same as in Figure 13, with the exception of the post read wait state.

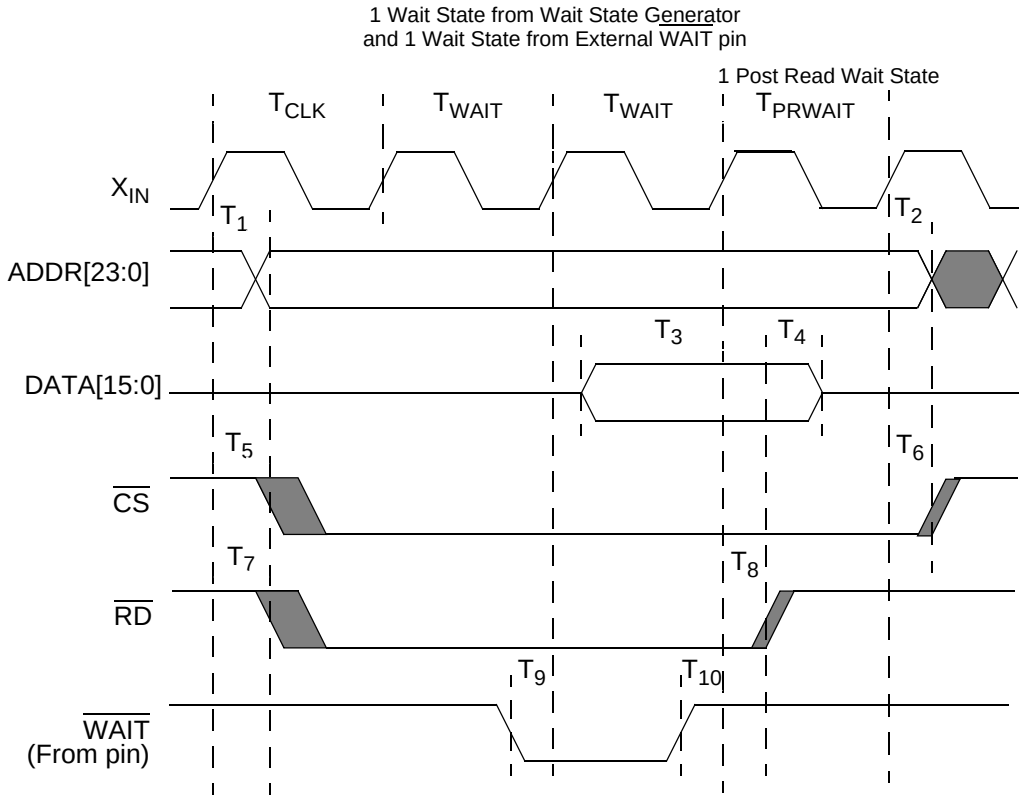


Figure 14. External Interface Timing for a Read Operation, 2 Wait States and 1 Post Read Wait State

External Interface Read Timing, ISA Mode

Figure 15 and Table 17 provide timing information for the external interface performing a read operation in ISA Mode. In Figure 15, it is assumed the wait state generator has been configured to provide 2 wait states during read operations. In Figure 15, it is also assumed that the chip select ($\overline{CS}$) signals have been configured for active Low operation. The Read signal ($\overline{RD}$) timing is shown for both NORMAL and ISA modes.

The most significant byte (MSB) of the four byte interrupt vector is not used. The vector is stored in the three least significant byte (LSB) of the vector, as shown in Table 40.

Table 40. Interrupt Vector Placement

Vector Byte	Data
0	Reserved
1	IRQ Vector[23:16]
2	IRQ Vector[15:8]
3	IRQ Vector[7:0]

Architecture

Figure 19 displays a block diagram of the interrupt controller.

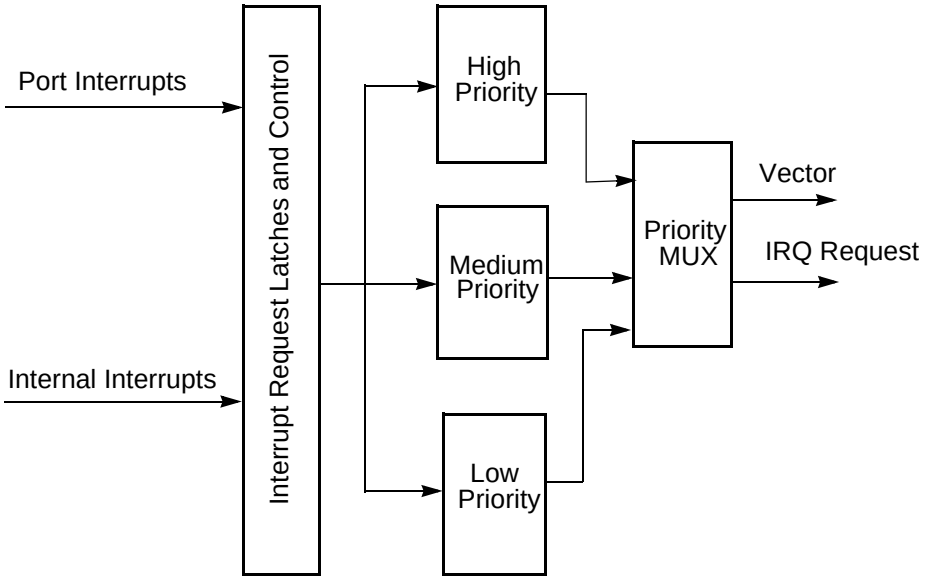


Figure 19. Interrupt Controller Block Diagram

Operation

Master Interrupt Enable

The master interrupt enable bit in the flag register globally enables or disables interrupts. This bit has been moved to the flag register (bit 0). Thus, anytime the register is loaded, it

PWM Fault Status Register (PWMFSTAT) is read to determine which fault source caused the interrupt.

When a fault is detected and the PWM outputs are disabled, modulator control of the PWM outputs are reenabled either by the software or by the fault input signal deasserting. Selection of the reenable method is made using the PWM Fault Control Register (PWMFCTL). Configuration of the fault modes and reenable methods allow pulse-by-pulse limiting and hard shutdown. When configured in Automatic Restart Mode, the PWM outputs are reengaged at beginning of the next PWM cycle (master timer value is equal to 0) if all fault signals are deasserted. In software controlled restart, all fault inputs must be deasserted and the fault flags must be cleared.

The fault input pin is Schmitt-triggered. The input signal from the pin as well as the comparators pass though an analog filter to reject high-frequency noise.

The logic path from the fault sources to the PWM output is asynchronous ensuring that the fault inputs forces the PWM outputs to their off-state even if the system clock is stopped.

PWM Operation in CPU Halt Mode

When the ZNEO CPU is operating in Halt Mode, the PWM continues to operate if it is enabled. To minimize current in Halt Mode, the PWM must be disabled by clearing the PWMEN bit to 0.

PWM Operation in CPU Stop Mode

When the ZNEO CPU is operating in Stop Mode, the PWM is disabled as the system clock ceases to operate in Stop Mode. The PWM output remains in the same state as they were prior to entering the Stop Mode. In normal operation, the PWM outputs must be disabled by software prior to the CPU entering the Stop Mode. A fault condition detected in Stop Mode forces the PWM outputs to the predefined off-state.

Observing the State of PWM Output Channels

The logic value of the PWM outputs is sampled by reading the PWMIN Register. If a PWM channel pair is disabled (option bit is not set), the associated PWM outputs are forced to high impedance and are used as general purpose inputs.

PWM Control Register Definitions

The following sections describe the various PWM control registers.

2. Load the appropriate 16-bit count value into the LIN-UART Baud Rate High and Low Byte registers.
3. Enable the BRG timer function and associated interrupt by setting the BRGCTL bit in the LIN-UART Control 1 Register to 1. Enable the UART receive interrupt in the interrupt controller.

When configured as a general purpose timer, the BRG interrupt interval is calculated using the following equation:

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

Noise Filter

A noise filter circuit is included to filter noise on a digital input signal, such as UART receive data before the data is sampled by the block. This filter is a requirement for protocols in a noisy environment.

The noise filter includes following features:

- Synchronizes the receive input data to the system clock.
- Noise filter enable (NFEN)input selects whether the noise filter is bypassed (NFEN = 0) or included (NFEN = 1) in the receive data path.
- Noise filter control (NFCTL[2:0])input selects the width of the up/down saturating counter digital filter. The available widths range is from 4 to11 bits.
- The digital filter output has hysteresis.
- Provides an active low saturated state output (FiltSatB), used to indicate presence of noise.

Architecture

Figure 30 displays how the noise filter is integrated with the LIN-UART for use on a LIN network.

Bit	Description
[2] TDRE	**Transmitter Data Register Empty** This bit indicates that the Transmit Data Register is empty and ready for additional data. Writing to the Transmit Data Register resets this bit. 0 = Do not write to the Transmit Data Register. 1 = The Transmit Data Register is ready to receive an additional byte to be transmitted.
[1] TXE	**Transmitter Empty** This bit indicates that the transmit shift register is empty and character transmission is finished. 0 = Data is currently transmitting. 1 = Transmission is complete.
[0] CTS	**$\overline{\text{CTS}}$ Signal** When this bit is read it returns the $\overline{\text{CTS}}$ signal level. If LBEN = 1, the $\overline{\text{CTS}}$ input signal is replaced by the internal receive data Available signal to provide flow control in LOOPBACK Mode. CTS only affects transmission if the CTSE bit = 1.

Table 83. LIN-UART Status 0 Register, LIN Mode (UxSTAT0)

Bits	7	6	5	4	3	2	1	0
Field	RDA	PLE	OE	FE	BRKD	TDRE	TXE	ATB
RESET	0	0	0	0	0	1	1	0
R/W	R	R	R	R	R	R	R	R
Addr	FF–E201H, FF–E211H							

Bit	Description
[7] RDA	**Receive Data Available** This bit indicates that the Receive Data Register has received data. Reading the Receive Data Register clears this bit. 0 = The Receive Data Register is empty. 1 = There is a byte in the Receive Data Register.
[6] PLE	**Physical Layer Error** This bit indicates that transmit and receive data do not match when a LIN slave or master is transmitting. This type of instance can be caused by a fault in the physical layer or multiple devices driving the bus simultaneously. Reading the Status 0 Register or the Receive Data Register clears this bit. 0 = Transmit and receive data match. 1 = Transmit and receive data do not match.

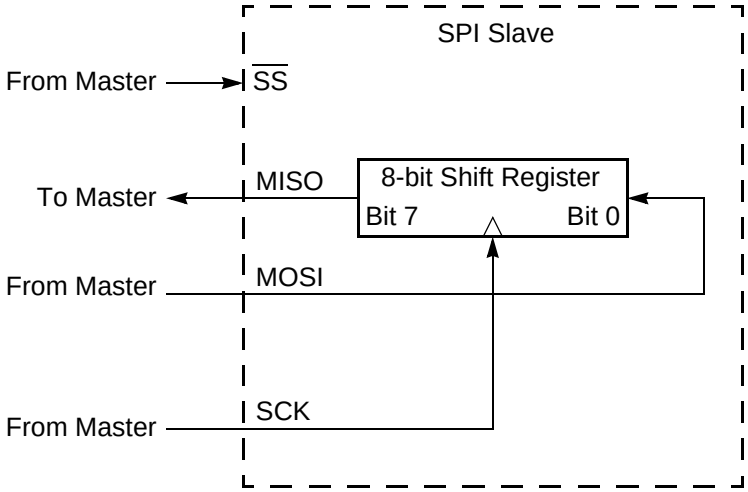


Figure 42. ESPI Configured as an SPI Slave

Error Detection

Error events detected by the ESPI block are described in this section. Error events generate an ESPI interrupt and set a bit in the ESPI Status Register. The error bits of the ESPI Status register are read/write 1 to clear.

Transmit Underrun

A transmit underrun error occurs for a master with SSMD = 10 or 11 when a character transfer completes and TDRE = 1. In these modes when a transmit underrun occurs the transfer is aborted (SCK will halt and SSV will be deasserted). For a master in SPI Mode (SSMD = 00), a transmit underrun is not signaled because SCK will pause and wait for the data register to be written.

In Slave Mode, a transmit underrun error occurs if TDRE = 1 at the start of a transfer. When a transmit underrun occurs in Slave Mode, ESPI transmits a character of all 1s.

A transmit underrun sets the TUND bit in the ESPI Status Register to 1. Writing 1 to TUND clears this error flag.

Mode Fault (Multi-Master Collision)

A mode fault indicates when more than one master is trying to communicate at the same time (a multi-master collision) in SPI Mode. The mode fault is detected when the enabled master's $\overline{SS}$ input pin is asserted. For this to happen the control and mode registers must be configured with MMEN = 1, SSIO = 0 ($\overline{SS}$ is an input) and $\overline{SS}$ input = 0. A mode fault sets the COL bit in the ESPI Status Register to 1. Writing a 1 to COL clears this error flag.

2. Software asserts the TXI bit of the I^2C Control Register to enable Transmit interrupts.
3. The I^2C interrupt asserts because the I^2C Data Register is empty.
4. Software responds to the TDRE interrupt by writing the first Slave address byte (11110xx0). The least-significant bit must be 0 for the write operation.
5. Software asserts the Start bit of the I^2C Control Register.
6. The I^2C Controller sends the Start condition to the I^2C Slave.
7. The I^2C Controller loads the I^2C Shift Register with the contents of the I^2C Data Register.
8. When one bit of address is shifted out by the SDA signal, the Transmit interrupt asserts.
9. Software responds by writing the second byte of address into the contents of the I^2C Data Register.
10. The I^2C Controller shifts the rest of the first byte of address and write bit out the SDA signal.
11. The I^2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL. The I^2C Controller sets the ACK bit in the I^2C Status Register.

    If the slave does not acknowledge the first address byte, the I^2C Controller sets the NCKI bit in the I^2C Status Register, sets the ACKV bit and clears the ACK bit in the I^2C State Register. Software responds to the Not Acknowledge interrupt by setting the Stop bit and clearing the TXI bit. The I^2C Controller flushes the second address byte from the data register, sends the Stop condition on the bus and clears the Stop and NCKI bits. The transaction is complete; ignore the following steps.
12. The I^2C Controller loads the I^2C Shift Register with the contents of the I^2C Data Register (2nd address byte).
13. The I^2C Controller shifts the second address byte out the SDA signal. When the first bit is sent, the Transmit interrupt asserts.
14. Software responds by writing the data to be written out to the I^2C Control Register.
15. The I^2C Controller shifts out the rest of the second byte of Slave address (or ensuing data bytes if looping) by the SDA signal.
16. The I^2C Slave sends an acknowledge by pulling the SDA signal Low during the next High period of SCL. The I^2C Controller sets the ACK bit in the I^2C Status Register.

    If the slave does not acknowledge, return to the second paragraph of Step 11.
17. The I^2C Controller shifts the data out by the SDA signal. After the first bit is sent, the Transmit interrupt asserts.
18. If more bytes remain to be sent, return to step 14.

Table 120. I2CSTATE_L (Continued)

State I2CSTATE_H	Sub-State I2CSTATE_L	Sub-State Name	State Description
0101	0000	Master Start	Initiating a new transaction.
	0001	Master Restart	Master is ending one transaction and starting a new one without letting the bus go nonactive.
1000–1111	0111	send/receive bit 7	Sending/Receiving most significant bit.
	0110	send/receive bit 6	
	0101	send/receive bit 5	
	0100	send/receive bit 4	
	0011	send/receive bit 3	
	0010	send/receive bit 2	
	0001	send/receive bit 1	
	0000	send/receive bit 0	Sending/Receiving least significant bit
	1000	send/receive Acknowl-edge	Sending/Receiving Acknowledge

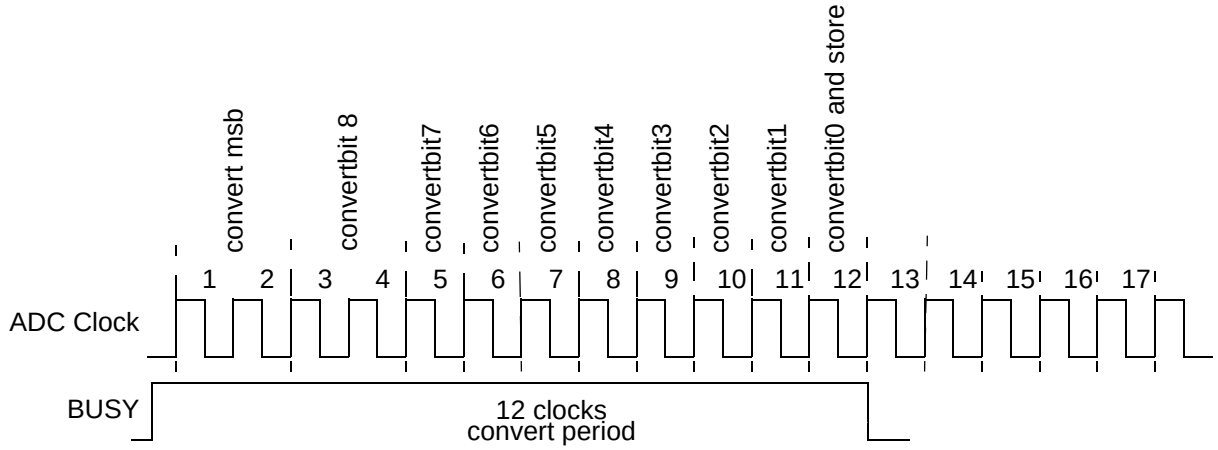


Figure 54. ADC Convert Timing

ADC Interrupts

The ADC generates an interrupt request when a conversion has been completed. An interrupt request pending when the ADC is disabled is not automatically cleared.

ADC0 Timer 0 Capture

The Timer 0 count is captured for every ADC0 conversion. The information is used to determine the zero crossing of back EMF in motor control applications. The capture of the Timer 0 count occurs when the programmed sample time is complete for every conversion and stored in the ADC timer capture register (ADCTCAP).

ADC Convert on Read

The ADC is set up to automatically convert the next channel input after reading the results of the current conversion. The conversions continue up to the channel listed in the ADC0MAX Register and then start over at the initial channel. The initial channel to convert is written to the control register, ADC0CTL, prior to starting the convert on Read process. Once started, the conversions continue to loop from the initial channel to Max channel until the convert on Read bit, CVTRD0, is cleared or the data is not read from the data registers.

Flash Sector Protect Register

The Flash Sector Protect Register, shown in Table 142, protects Flash memory sectors from being programmed or erased from user code. User code can only write bits in this register to 1 (bits cannot be cleared to 0 by user code).

Table 142. Flash Sector Protect Register (FSECT)

Bits	7	6	5	4	3	2	1	0
Field	SECT7	SECT6	SECT5	SECT4	SECT3	SECT2	SECT1	SECT0
RESET	0	0	0	0	0	0	0	0
R/W	R/W1	R/W1	R/W1	R/W1	R/W1	R/W1	R/W1	R/W1
Addr	FF_E062h							

Note: R/W1 = Register is accessible for read operations. Register is written to 1 only (via user code).

Bit	Description
[7:0] SECT*n*	**Sector Protect** 0 = Sector *n* is programmed or erased from user code. 1 = Sector *n* is protected and cannot be programmed or erased from user code.

Note: User code write bits from 0 to 1 only.

Bit	Description (Continued)
[3:0] REQSEL	**DMA Request Selection by Channel**
DMA0	**DMA 0 Request Select** 0000 = Continuous (i.e., Memory to Memory) 0001 = Timer 0 0010 = Timer 1 0011 = Timer 2 0100 = UART0 RXD 0101 = UART0 TXD 0110 = UART1 RXD 0111 = UART1 TXD 1000 = I2C RX 1001 = I2C TX 1010 = SPI RX 1011 = SPI TX 1100 = ADC0 1101 = Reserved 1110 = Reserved 1111 = $\overline{\text{DMA0REQ}}$ Pin
DMA1	**DMA 1 Request Select** 0000 = Continuous (i.e., Memory to Memory) 0001 = Timer 0 0010 = Timer 1 0011 = Timer 2 0100 = UART0 RXD 0101 = UART0 TXD 0110 = UART1 RXD 0111 = UART1 TXD 1000 = I2C RX 1001 = I2C TX 1010 = SPI RX 1011 = SPI TX 1100 = ADC0 1101 = Reserved 1110 = Reserved 1111 = $\overline{\text{DMA1REQ}}$ Pin

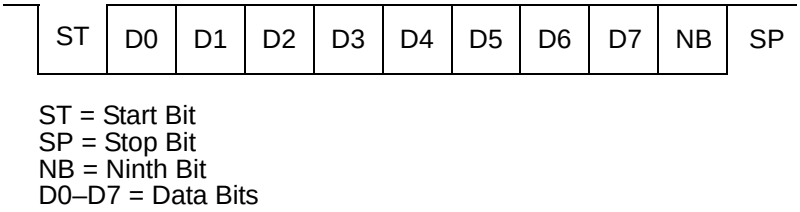


Figure 67. 9-Bit Mode

Start Bit Flow Control

If flow control is required, start bit flow control is used. Start bit flow control requires the receiving device send the start bit. The transmitter waits for the start bit, then transmit its data following the start bit.

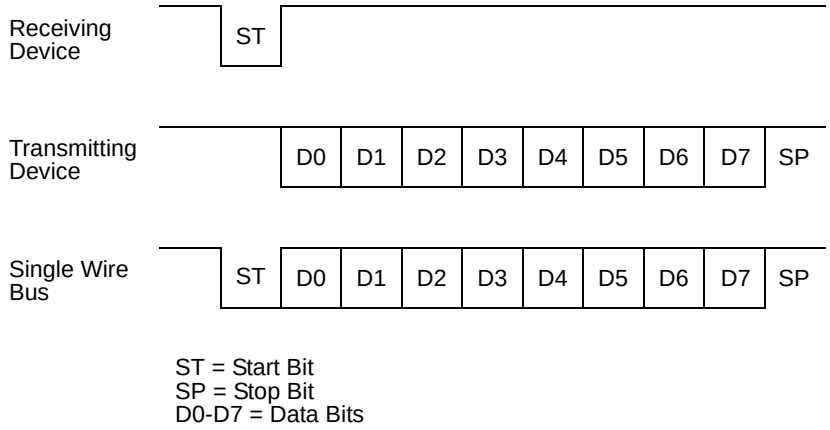


Figure 68. Start Bit Flow Control

If the standard serial port of a PC is used, transmit flow control is enabled on the ZNEO Z16F Series device. The PC sends the start bit when receiving data by transmitting the character FFh. Because the FFh character is also received from a nonresponsive device, space parity (parity bit always zero) must be enabled and used as an acknowledge bit.

Initialization

The OCD ignores any data received until it receives the read revision command 00h. After the read revision command is received, the remaining debug commands are issued. The packet CRC is not sent for the first read revision command issued during initialization.

On-Chip Debugger Commands

The hardware OCD supports several commands for controlling the device. In the following list of commands, data sent from the host to the OCD is identified by:

```
DBG <-- Data
```

Data sent from the OCD back to the host is identified by:

```
DBG --> Data
```

Multiple bytes transmitted are represented with double arrows, either <<or >>.

Read Revision. The Read Revision command returns the revision identifier.

```
DBG <-- 0000_0000
DBG --> RevID[15:8]
DBG --> RevID[7:0]
DBG --> CRC[0:7]
```

Read Status Register. The Read Status Register command returns the contents of the OCDSTAT Register.

```
DBG <-- 0000_0001
DBG --> status[7:0]
DBG --> CRC[0:7]
```

Read Control Register. The Read Control register command returns the contents of the OCDCTL Register.

```
DBG <-- 0000_0010
DBG --> OCDCTL[7:0]
DBG --> CRC[0:7]
```

Write Control Register. The Write Control register command writes data to the OCDCTL Register.

```
DBG <-- 0000_0011
DBG <-- OCDCTL[7:0]
DBG --> CRC[0:7]
```

Read Registers. The Read registers command returns the contents of CPU registers R15 through R0.

```
DBG <-- 0000_0100
DBG ->> regdata[31:24]
DBG ->> regdata[23:16]
DBG ->> regdata[15:8]
DBG ->> regdata[7:0]
DBG --> CRC[0:7]
```

Write Registers. The Write registers command writes data to CPU registers R15 through R0.

```
DBG <-- 0000_0101
DBG <<- regdata[31:24]
```

Bit	Description (Continued)
[23:16] ADDR[23:16]	**Breakpoint Address** The address to match when generating a breakpoint.
[15:0] ADDR[15:0]	

Trace Control Register

The Trace Control Register (TRACECTL), shown in Table 179, is used to enable the Trace operation. It also selects the size of the trace buffer.

Table 179. Trace Control Register (TRACECTL)

Bits	7	6	5	4	3	2	1	0
Field	TRACEEN	Reserved				TRACESEL		
RESET	0	0	0	0	0	000		
R/W	R/W	R	R	R	R	R/W		
Addr	FF_E013							

Bit	Description
[7] TRACEEN	**Trace Enable** 0 = Trace is disabled. 1 = Traces is enabled
[6:3]	**Reserved** This bit is reserved and must be programmed to 0000.
[2:0] TRACESEL	**Trace Size Select** 000 – 128 Bytes (16 Events) 001 – 256 Bytes (32 Events) 010 – 512 Bytes (64 Events) 011 – 1024 Bytes (128 Events) 100 – 2048 Bytes (256 Events) 101 – 4096 Bytes (512 Events) 110 – 8192 Bytes (1024 Events) 111 – 16384 Bytes (2048 Events)

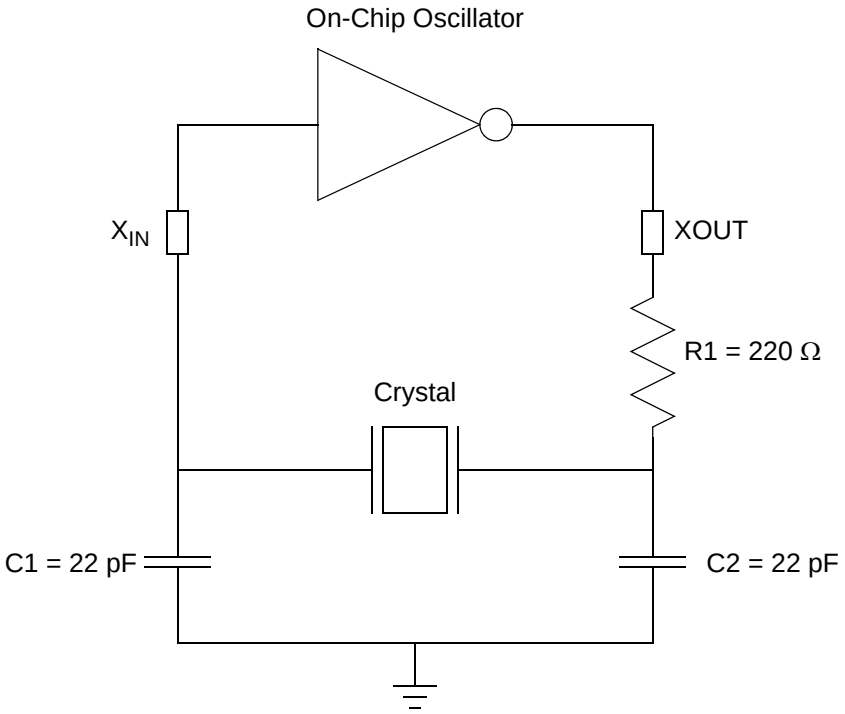


Figure 70. Recommended 20 MHz Crystal Oscillator Configuration

Table 181. Recommended Crystal Oscillator Specifications (20 MHz Operation)

Parameter	Value	Units	Comments
Frequency	20	MHz	
Resonance	Parallel		
Mode	Fundamental		
Series Resistance (R_S)	25	Ω	Maximum
Load Capacitance (C_L)	20	pF	Maximum
Shunt Capacitance (C_0)	7	pF	Maximum
Drive Level	1	mW	Maximum

Oscillator Control

The ZNEO® Z16F Series uses three possible user-selectable clocking schemes:

- Trimmable internal precision oscillator
- On-chip oscillator using off-chip crystal/resonator or external clock driver
- On-chip low-precision Watchdog Timer oscillator

In addition, ZNEO Z16F Series contain clock failure detection and recovery circuitry, allowing continued operation despite a failure of the primary oscillator.

The on-chip system clock frequency is reduced through a clock divider allowing reduced dynamic power dissipation. The FLASH is powered down during portions of the clock period when running slower than 10 MHz.

Operation

This section explains the logic used to select the system clock, divide down the system clock and handle oscillator failures. A description of the specific operation of each oscillator is outlined elsewhere in this document. See the Watchdog Timer chapter on page 238, the Internal Precision Oscillator chapter on page 336 and the On-Chip Oscillator chapter on page 327.

System Clock Selection

The oscillator control block selects from the available clocks. Table 182 details each clock source and its usage.

General Purpose I/O Port Input Data Sample Timing

Figure 76 displays timing of the GPIO port input sampling. The input value on a GPIO port pin is sampled on the rising edge of the system clock. The port value is then available to the ZNEO CPU on the second rising clock edge following the change of the port value. Table 195 lists the GPIO port input timing.

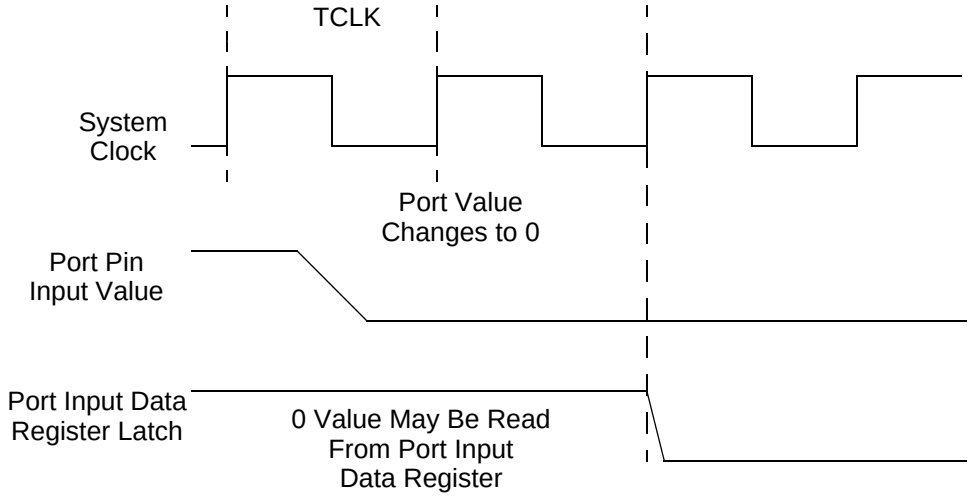


Figure 76. Port Input Sample Timing

Table 195. GPIO Port Input Timing

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
T_{SMR}	GPIO Port Pin Pulse Width to ensure Stop Mode Recovery (for GPIO Port Pins enabled as SMR sources)	1µs	

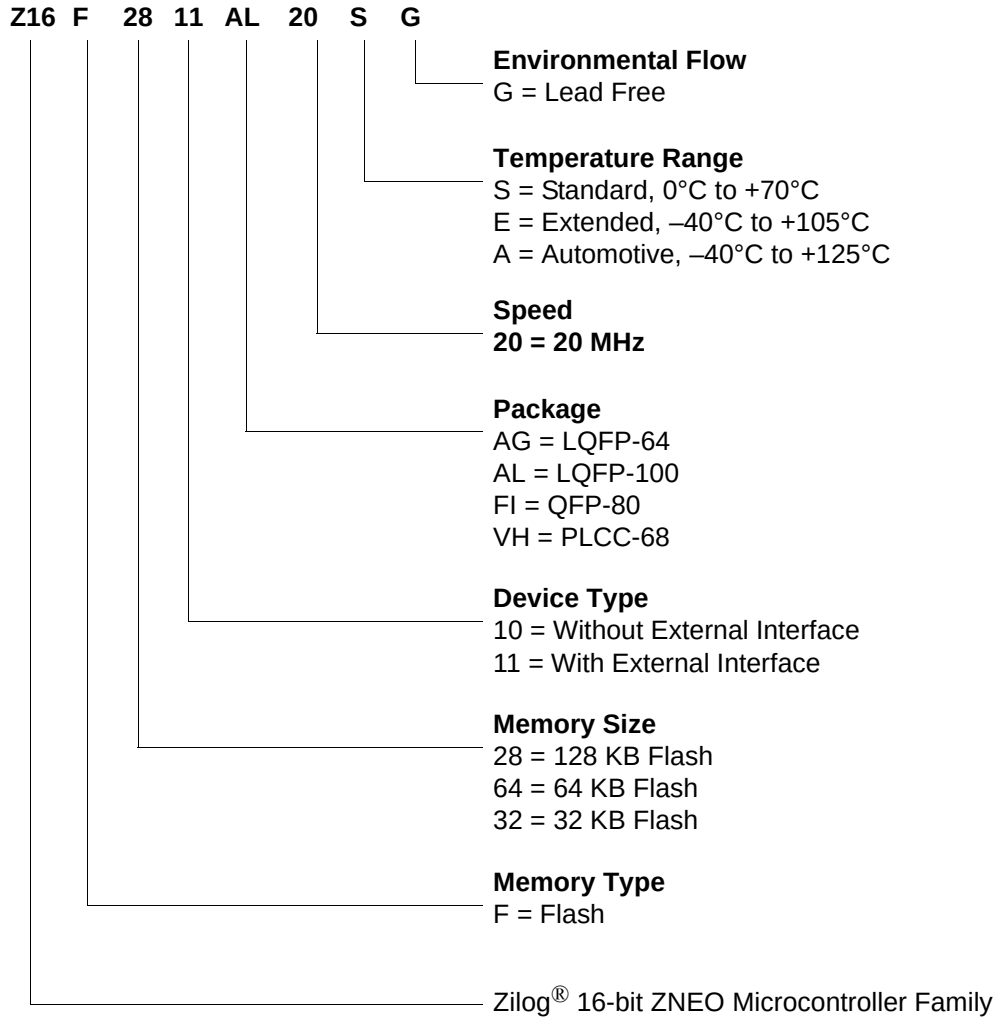
On-Chip Debugger Timing

Table 196 provides timing information for the DBG pin. The DBG pin timing specifications assume a 4µs maximum rise and fall time.

Table 196. On-Chip Debugger Timing

		Delay (ns)	
Parameter	**Description**	**Min**	**Max**
DBG	Debug frequency.	—	System Clock/4

Part Number Suffix Designations



Note: Packages are not available for all memory sizes. See the Ordering Information section on page 356 for available packages.

Precharacterization Product

The product represented by this document is newly introduced and Zilog® has not completed the full characterization of the product. The document states what Zilog knows about this product at this time, but additional features or nonconformance with some aspects of the document might be found, either by Zilog or its customers in the course of further application and characterization work. In addition, Zilog cautions that delivery might be uncertain at times, due to start-up yield issues. For more information, please visit www.zilog.com.