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	ZNEO
Core Size	16-Bit
Speed	20MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	46
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 ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	68-LCC (J-Lead)
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z16f2810vh20ag

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.

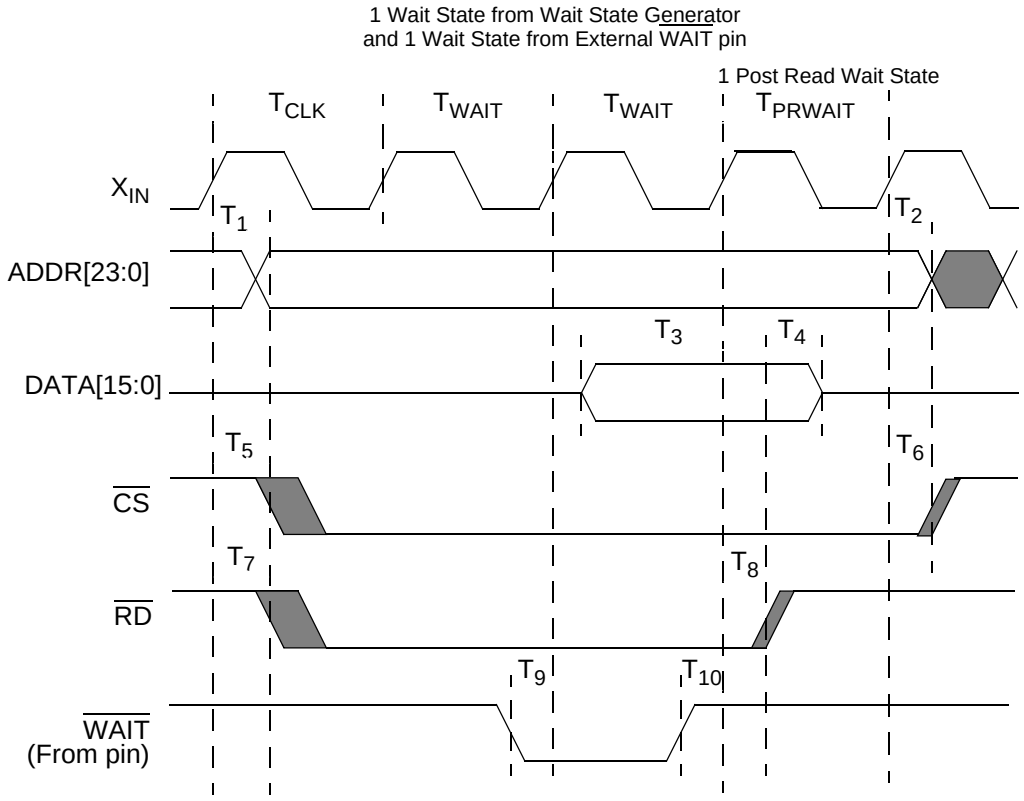

Figure 14. External Interface Timing for a Read Operation, 2 Wait States and 1 Post Read Wait State

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.

Stop Mode Recovery only affects the contents of the the Reset Status and Control Register (see page 62) and the Oscillator Control Register (see page 333). Stop Mode Recovery does not affect any other values in the register file, including the stack pointer, register pointer, flags, peripheral control registers and general-purpose RAM.

The ZNEO CPU fetches the Reset vector at program memory addresses 0004h-0007h and loads that value into the program counter. Program execution begins at the Reset vector address. Following Stop Mode Recovery, the Stop bit in the Reset Status and Control Register is set to 1. Table 20 lists the Stop Mode Recovery sources and resulting actions. The following text provides more detailed information about each of the Stop Mode Recovery sources.

Table 20. Stop Mode Recovery Sources and Resulting Action

Operating Mode	Stop Mode Recovery Source	Action
Stop Mode	WDT time-out when configured for Reset	Stop Mode Recovery
	WDT time-out when configured for System Exception	Stop Mode Recovery followed by WDT System Exception
	Data transition on any GPIO Port pin enabled as a Stop Mode Recovery source	Stop Mode Recovery

Stop Mode Recovery Using WDT Time-Out

If the WDT times out during Stop Mode, the device undergoes a Stop Mode Recovery sequence. In the Reset Status and Control Register, the WDT and Stop bits are set to 1. If the WDT is configured to generate a System Exception on time-out, the ZNEO CPU services the WDT System Exception following the normal Stop Mode Recovery sequence.

Stop Mode Recovery Using a GPIO Port Pin Transition

Each of the GPIO port pins is configured as a Stop Mode Recovery input source. If any GPIO pin enabled as a Stop Mode Recovery source, a change in the input pin value (from High to Low or from Low to High) initiates Stop Mode Recovery. The GPIO Stop Mode Recovery signals are filtered to reject pulses less than 10 ns (typical) in duration. In the Reset Status and Control Register, the Stop bit is set to 1.

Caution: Short pulses on the port pin initiates Stop Mode Recovery without initiating an interrupt (if enabled for that pin).

Bit	Description
[6:1]	**Reserved** These bits are reserved and must be programmed to 000000.
[0] DBGIMUX	**Debug Interrupt MUX** 0 = Select Port A0/D0 based on the Port A IRQ edge register as the interrupt source. 1 = Select the DBG as the interrupt source.

Port A IRQ MUX Register

The Port IRQ MUX Register, shown in Table 36, selects either Port A or Port D pins as interrupt sources.

Table 36. Port A IRQ MUX Register (PAIMUX)

Bits	7	6	5	4	3	2	1	0
Field	PAIMUX7	PAIMUX6	PAIMUX5	PAIMUX4	PAIMUX3	PAIMUX2	PAIMUX1	PAIMUX0
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
Addr	FF_E10E							

Bit	Description
PAIMUX[7:0]	**Port A/D Interrupt Source** 0 = Select Port A*x* as interrupt source. 1 = Select Port D*x* as interrupt source.

Port A IRQ Edge Register

The Port IRQ Edge Register, shown in Table 37, selects either positive or negative edge as the port pin interrupt sources.

Table 37. Port A IRQ Edge Register (PAIEDGE)

Bits	7	6	5	4	3	2	1	0
Field	PAIEDGE7	PAIEDGE6	PAIEDGE5	PAIEDGE4	PAIEDGE3	PAIEDGE2	PAIEDGE1	PAIEDGE0
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
Addr	FF_E10F							

Bit	Description
[7:0] PAIEDGE[7:0]	**Port A/D Interrupt Edge** 0 = Select Port A/D pin negedge as interrupt source. 1 = Select Port A/D pins posedge as interrupt source.

Timer Operating Modes

The timers are configured to operate in the following modes:

One-Shot Mode

In One-Shot 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. When the timer reaches the reload value, it generates an interrupt and the count value in the Timer High and Low byte registers is reset to `0001h`. The timer is automatically disabled and stops counting.

If the timer output alternate function is enabled, the timer output pin changes state for one system clock cycle (from Low to High then back to Low if TPOL = 0) at timer Reload. If the timer output is required to make a permanent state change on One-Shot time-out, first set the TPOL bit in the Timer Control 1 Register to the start value before beginning One-Shot Mode. Then, after starting the timer, set TPOL to the opposite value.

Observe the following steps to configure a timer for One-Shot Mode and initiate the count:

1. Write to the timer control registers to:
   - Disable the timer
   - Configure the timer for One-Shot Mode
   - Set the prescale value
   - Set the initial output level (High or Low) using the TPOL bit for the timer output alternate function
   - Set the INTERRUPT Mode
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. Enable the timer interrupt, if required and set the timer interrupt priority by writing to the relevant interrupt registers.
5. When using the timer output function, configure the associated GPIO port pin for the timer output alternate function.
6. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The timer period is calculated by the following equation (start value = 1):

$$\text{One-Shot Mode Time-Out Period (s)} = \frac{(\text{Reload Value} - \text{Start Value} + 1) \times \text{Prescale}}{\text{System Clock Frequency (Hz)}}$$

Table 56. Timer 0–2 High Byte Register (TxH)

Bits	7	6	5	4	3	2	1	0
Field	TH							
RESET	00h							
R/W	R/W							
Addr	FF–E300h, FF–E310h, FF–E320h							

Bit	Description
[7:0] TH	**Timer High and Low Byte** These two bytes, {TH[7:0], TL[7:0]}, contain the current 16-bit timer count value.

Table 57. Timer 0–2 Low Byte Register (TXL)

Bits	7	6	5	4	3	2	1	0
Field	TL							
RESET	01h							
R/W	R/W							
Addr	FF–E301h, FF–E311h, FF–E321h							

Bit	Description
[7:0] TL	**Timer High and Low Byte** These two bytes, {TH[7:0], TL[7:0]}, contain the current 16-bit timer count value.

Multi-Channel PWM Timer

The ZNEO® Z16F Series includes a Multi-Channel PWM optimized for motor control applications. The PWM includes the following features:

- Six independent PWM outputs or three complementary PWM output pairs
- Programmable deadband insertion for complementary output pairs
- Edge-aligned or center-aligned PWM signal generation
- PWM off-state is an option bit programmable
- PWM outputs driven to off-state on System Reset
- Asynchronous disabling of PWM outputs on system fault; outputs are forced to off-state
- Fault inputs generate pulse-by-pulse or hard shutdown
- 12-bit reload counter with 1, 2, 4 or 8 programmable clock prescaler
- High current source and sink on all PWM outputs
- PWM pairs used as general purpose inputs when outputs are disabled
- ADC synchronized with PWM period
- Synchronization for current-sense sample and hold
- Narrow pulse suppression with programmable threshold

Architecture

The PWM unit consists of a master timer to generate the modulator time base and six independent compare registers to set the PWM for each output. The six outputs are designed to provide control signals for inverter drive circuits. The outputs are grouped into pairs consisting of a high-side driver and a low-side driver output. The output pairs are programmable to operate independently or as complementary signals.

In complementary output mode, a programmable dead-time is inserted to ensure nonoverlapping signal transitions. The master count and compare values feed into modulator logic which generates the proper transitions in the output states. Output polarity and fault/off-state control logic allows programming of the default off-states which forces the outputs to a safe state in the event a fault in the motor drive is detected. Figure 21 displays the architecture of the PWM modulator.

c. Set or clear the CTSE bit to enable or disable control from the remote receiver through the $\overline{CTS}$ pin.

7. Execute an EI instruction to enable interrupts.

The LIN-UART is now configured for interrupt-driven data transmission. As the LIN-UART Transmit Data Register is empty, an interrupt is generated immediately. When the LIN-UART transmit interrupt is detected and there is transmit data ready to send, the associated interrupt service routine (ISR) performs the following operations:

1. If operating in Multiprocessor Mode, write the LIN-UART Control 1 Register to select the outgoing address bit by setting the multiprocessor bit transmitter (MPBT) if sending an address byte; clear it if sending a data byte.
2. Write the data byte to the LIN-UART Transmit Data Register. The transmitter automatically transfers the data to the transmit shift register and transmits the data.
3. Execute the IRET instruction to return from the interrupt service routine and waits for the Transmit Data Register to again become empty.

If a transmit interrupt occurs and there is no transmit data ready to send, the interrupt service routine executes the IRET instruction. When the application contains data to transmit, software sets the appropriate interrupt request bit in the interrupt controller to initiate a new transmit interrupt. Another alternative would be for software to write the data to the Transmit Data Register instead of invoking the ISR.

Receiving Data Using the Polled Method

Observe the following steps to configure the LIN-UART for polled data reception:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-UART pin functions by configuring the associated GPIO port pins for alternate function operation.
3. Write to the LIN-UART Control 1 Register to enable Multiprocessor Mode functions.
4. Write to the LIN-UART Control 0 Register to:
   a. Set the receive enable bit (REN) to enable the LIN-UART for data reception.
   b. Enable parity, if Multiprocessor Mode is not enabled and select either even or odd parity.
5. Check the RDA bit in the LIN-UART Status 0 Register to determine if the Receive Data Register contains a valid data byte (indicated by a 1). If RDA is set to 1 to indicate available data, continue to Step 6. If the Receive Data Register is empty (indicated by a 0), continue to monitor the RDA bit awaiting reception of the valid data.

6. Read data from the LIN-UART Receive Data Register. If operating in MULTIPROCESSOR (9-Bit) Mode, further actions are required depending on the Multiprocessor Mode bits MPMD[1:0].
7. Return to Step 5 to receive additional data.

Receiving Data Using the Interrupt-Driven Method

The LIN-UART receiver interrupt indicates the availability of new data (as well as error conditions). Observe the following steps to configure the LIN-UART receiver for interrupt-driven operation:

1. Write to the LIN-UART Baud Rate High and Low Byte registers to set the appropriate baud rate.
2. Enable the LIN-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 LIN-UART receiver interrupt and set the appropriate priority.
5. Clear the LIN-UART receiver interrupt in the applicable interrupt request register.
6. Write to the LIN-UART Control 1 Register to enable MULTIPROCESSOR (9-Bit) Mode functions:
   a. Set the Multiprocessor Mode select (MPEN) to enable Multiprocessor Mode.
   b. Set the Multiprocessor Mode bits, MPMD[1:0], to select the appropriate address matching scheme.
   c. Configure the LIN-UART to interrupt on received data and errors or errors only (interrupt on errors only is unlikely to be useful for ZNEO devices without a DMA block).
7. Write the device address to the Address Compare Register (automatic multiprocessor modes only).
8. Write to the LIN-UART Control 0 Register to:
   a. Set the receive enable bit (REN) to enable the LIN-UART for data reception
   b. Enable parity, if Multiprocessor Mode is not enabled and select either even- or odd-parity.
9. Execute an EI instruction to enable interrupts.

The LIN-UART is now configured for interrupt-driven data reception. When the LIN-UART receiver interrupt is detected, the associated ISR performs the following operations:

Transmitter Interrupts

The transmitter generates a single interrupt when the Transmit Data Register empty bit (TDRE) is set to 1. This indicates that the transmitter is ready to accept new data for transmission. The TDRE interrupt occurs when the transmitter is initially enabled and after the transmit shift register has shifted the first bit of a character out. At this point, the Transmit Data Register is written with the next character to send. This provides 7 bit periods of latency to load the Transmit Data Register before the transmit shift register completes shifting the current character. Writing to the LIN-UART Transmit Data Register clears the TDRE bit to 0.

Receiver Interrupts

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

- A data byte is received and is available in the LIN-UART Receive Data Register. This interrupt is disabled independent of the other receiver interrupt sources using the RDAIRQ bit (this feature is useful in devices, which support DMA). The received data interrupt occurs after the receive character is placed in the Receive Data Register. To avoid an overrun error, the software responds to this received data available condition before the next character is completely received.

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

- A break is received.
- A receive data overrun or LIN slave autobaud overrun error is detected.
- A data framing error is detected.
- A parity error is detected (physical layer error in LIN Mode).

LIN-UART Overrun Errors

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

When the valid data is read, the OE bit of the Status 0 Register is updated to indicate the overrun condition (and Break Detect, ifapplicable). 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 may not contain valid data and must be ignored. The BRKD bit indicates if the overrun is caused due to a break condition on the line. After reading the status

LIN-UART Control 0 Register

The LIN-UART Control 0 Register, shown in Table 89, configures the basic properties of the LIN-UART's transmit and receive operations.

Table 89. LIN-UART Control 0 Register (UxCTL0)

Bits	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
Addr	FF–E202h, FF–E212h							

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	**CTS 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 transmitted and expected on all received data. 1 = Odd parity is transmitted and expected on all received data.

Bit	Description (Continued)
[1:0] TxBreakLength	**TxBreakLength** Used in LIN Mode by the master to control the duration of the transmitted Break. 00 = 13 bit times 01 = 14 bit times 10 = 15 bit times 11 = 16 bit times

LIN-UART Address Compare Register

The LIN-UART Address Compare Register, shown in Table 93, stores the multi-node network address of the LIN-UART. When the MPMD[1] bit of LIN-UART Control Register 0 is set, all incoming address bytes are compared to the value stored in the Address Compare Register. Receive interrupts and RDA assertions occur only in the event of a match.

Table 93. LIN-UART Address Compare Register (UxADDR)

Bits	7	6	5	4	3	2	1	0
Field	COMP_ADDR							
RESET	00h							
R/W	R/W							
Addr	FF–E205h, FF–E215h							

Bit	Description
[7:0] COMP_ADDR	**Compare Address** This 8-bit value is compared to the incoming address bytes.

LIN-UART Baud Rate High and Low Byte Registers

The LIN-UART Baud Rate High and Low Byte registers, shown in Tables 94 and 95, combine to create a 16-bit baud rate divisor value (BRG[15:0]) which sets the data transmission rate (baud rate) of the LIN-UART.

Table 94. LIN-UART Baud Rate High Byte Register (UxBRH)

Bits	7	6	5	4	3	2	1	0
Field	BRH							
RESET	1							
R/W	R/W							
Addr	FF–E206h, FF–E216h							

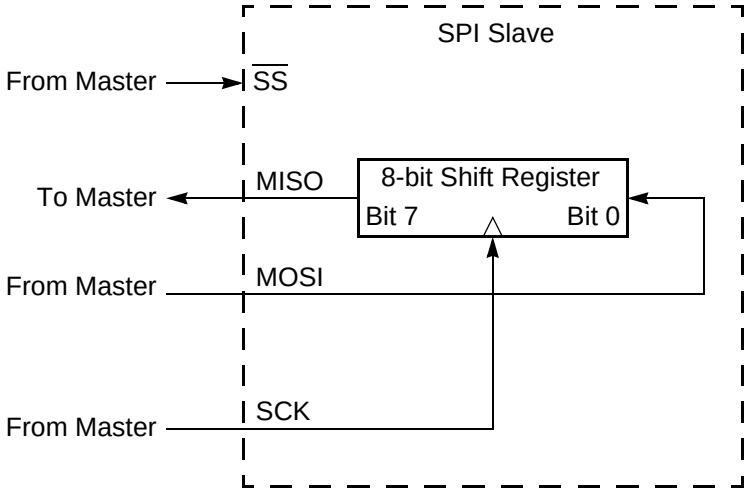

Figure 42. ESPI Configured as an SPI Slave

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.

ESPI State Register

The ESPI State Register, shown in Table 107, provides observability of the ESPI clock, data and internal state. Table 108 describes the ESPI State Machine values.

Table 107. ESPI State Register (ESPISTATE)

Bits	7	6	5	4	3	2	1	0
Field	SCKI	SDI	ESPISTATE					
RESET	0	0	0					
R/W	R	R	R					
Addr	FF_E265h							

Bit	Description
[7] SCKI	**Serial Clock Input** This bit reflects the state of the serial clock pin. 0 = The SCK input pin is Low 1 = The SCK input pin is High
[6] SDI	**Serial Data Input** This bit reflects the state of the serial data input (MOSI or MISO depending on the MMEN bit). 0 = The serial data input pin is Low. 1 = The serial data input pin is High.
[5:0] ESPISTATE	**ESPI State Machine** Indicates the current state of the internal ESPI State Machine. This information is intended for manufacturing test. The state values may change in future hardware revisions and are not intended to be used by a software driver. Table 108 defines the valid states.

Table 108. ESPISTATE Values and Description

ESPISTATE Value	Description
00_0000	Idle
00_0001	Slave Wait For SCK
00_0010	I2S Slave Mode start delay
00_0011	I2S Slave Mode start delay
01_0000	SPI Master Mode start delay
11_0001	I2S Master Mode start delay
11_0010	I2S Master Mode start delay
10_1110	Bit 7 Receive
10_1111	Bit 7 Transmit

Each interrupt source other than the baud rate generator interrupt has an associated bit in the I2CISTAT Register, which clears automatically when software reads the register or performs some other task such as reading or writing the data register.

Transmit Interrupts

Transmit interrupts (TDRE bit = 1 in I2CISTAT) occur under the following conditions:

- The Transmit Data Register is empty and the TXI bit = 1 in the I^2C Control Register
- The I^2C Controller is enabled, with any one of the following operations:
  - The first bit of a 10-bit address is shifted out
  - The first bit of the final byte of an address is shifted out and the RD bit is deasserted
  - The first bit of a data byte is shifted out

Writing to the I^2C Data Register always clears the TRDE bit to 0.

Receive Interrupts

Receive interrupts (RDRF bit = 1 in I2CISTAT) occur when a byte of data has been received by the I^2C Controller. The RDRF bit is cleared by reading from the I^2C Data Register. If the RDRF interrupt is not serviced prior to the completion of the next receive byte, the I^2C Controller holds SCL Low during the last data bit of the next byte until RDRF is cleared to prevent receive overruns. A receive interrupt does not occur when a Slave receives an address byte or for data bytes following a Slave address that did not match. An exception is if the interactive receive mode (IRM) bit is set in the I2CMODE Register in which case receive interrupts occur for all receive address and data bytes in Slave Mode.

Slave Address Match Interrupts

Slave address match interrupts (SAM bit = 1 in I2CISTAT) occur when the I^2C Controller is in Slave Mode and an address is received which matches the unique Slave address. The General Call Address (0000_0000) and STARTBYTE (0000_0001) are recognized if the GCE bit = 1 in the I2CMODE Register. Software verifies the RD bit in the I2CISTAT Register to determine if the transaction is a read or write transaction. The General Call Address and STARTBYTE addresses are also distinguished by the RD bit. The general call address (GCA) bit of the I2CISTAT Register indicates whether the address match occurred on the unique Slave address or the General Call/STARTBYTE address. The SAM bit clears automatically when the I2CISTAT Register is read.

If configured using the MODE[1:0] field of the I^2C Mode Register for 7-bit slave addressing, the most significant 7 bits of the first byte of the transaction are compared against the SLA[6:0] bits of the Slave Address Register. If configured for 10-bit slave addressing, the

Flash Read Protection

The user code within the Flash memory is protected from external access. Programming the Flash Read Protect option bit prevents reading of user code by the OCD or by using the Flash Controller Bypass Mode. For more information, see the Option Bits chapter on page 292 and the On-Chip Debugger chapter on page 298.

Flash Write/Erase Protection

The ZNEO Z16F Series provides several levels of protection against accidental program and erasure of the Flash memory contents. This protection is provided by the Flash Controller unlock mechanism, the Flash Sector Protect register and the Flash Write Protect option bit.

Flash Controller Unlock Mechanism

At Reset, the Flash Controller locks to prevent accidental program or erasure of the Flash memory. To program or erase the Flash memory, the Flash controller must be unlocked. After unlocking the Flash Controller, the Flash is programmed or erased. Any value written by user code to the Flash Command Register or Flash Page Select Register out of sequence locks the Flash Controller.

Observe the following steps to unlock the Flash Controller from user code:

1. Write the page to be programmed or erased to the Flash Page Select Register.
2. Write the first unlock command 73h to the Flash Command Register.
3. Write the second unlock command 8Ch to the Flash Command Register.

Flash Sector Protection

The Flash Sector Protect register is configured to prevent sectors from being programmed or erased. After a sector is protected, it cannot be unprotected by user code. The Flash Sector Protect register is cleared after reset and any previously written protection values will be lost. User code must write this register in their initialization routine if they want to enable sector protection.

When user code writes the Flash Sector Protect register, bits are set to 1 only. Thus, sectors are protected, but not unprotected, using register write operations.

Flash Write Protection Option Bit

The Flash Write Protect option bit is enabled to block all program and erase operations from user code. For more detail, see the Option Bits chapter on page 292.

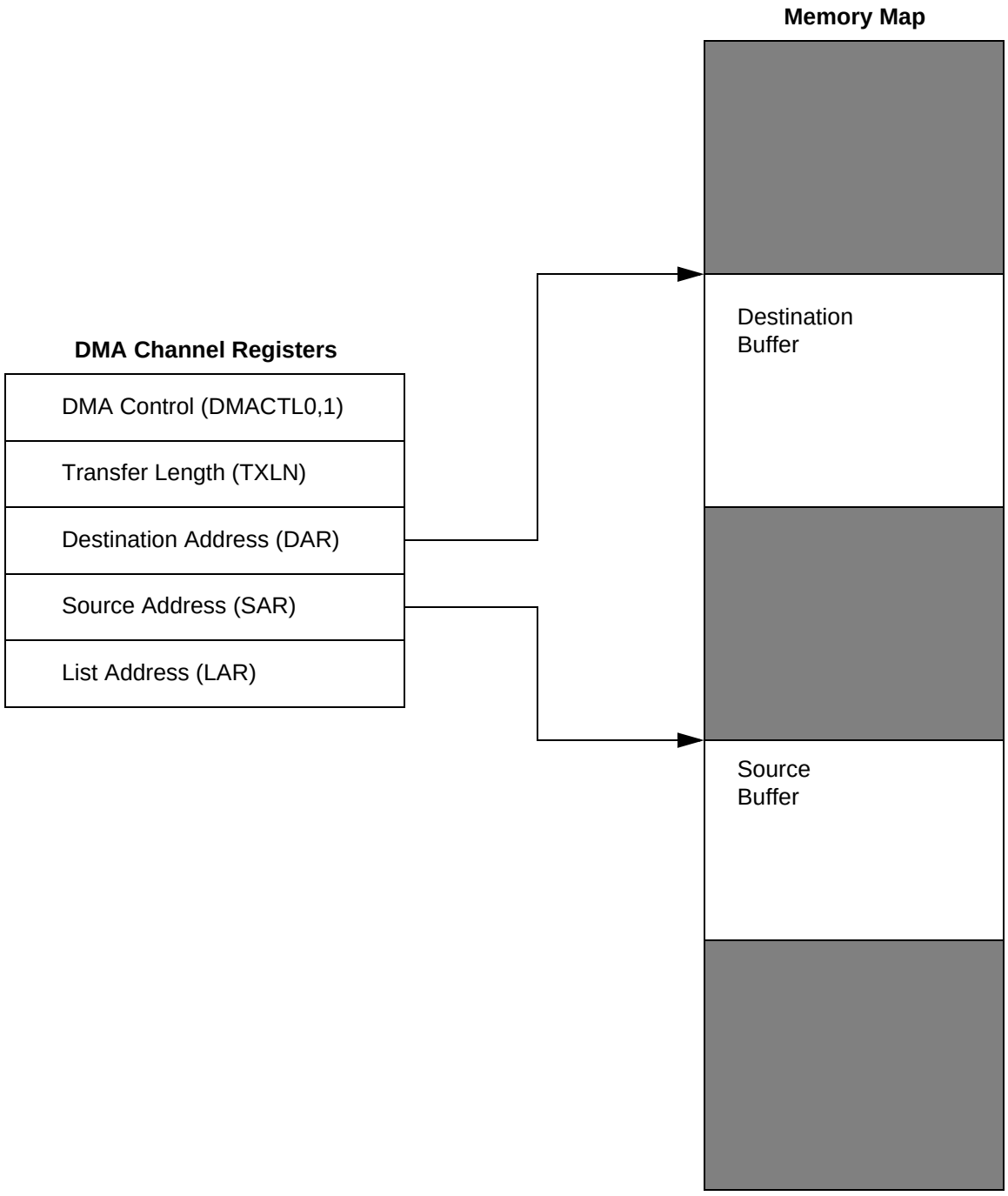

Figure 58. Direct DMA Diagram

External DMA Signals

Two external pins are associated with each DMA Channel capable of external transfers (Channel 3 does not have external DMA capability). They are active Low DMAxREQ and DMAxACK signals. DMAxACK signals are outputs and DMAxREQ are inputs. DMAxREQ must be asserted for a minimum of one system clock period to generate one DMA transfer. DMAxREQ is left asserted for multiple transactions and deasserted after DMAxACK asserts for the last appropriate transfer.

DMA Timing

External DMA Transfer

Figure 60 shows the read and write timing of external DMA transfers.

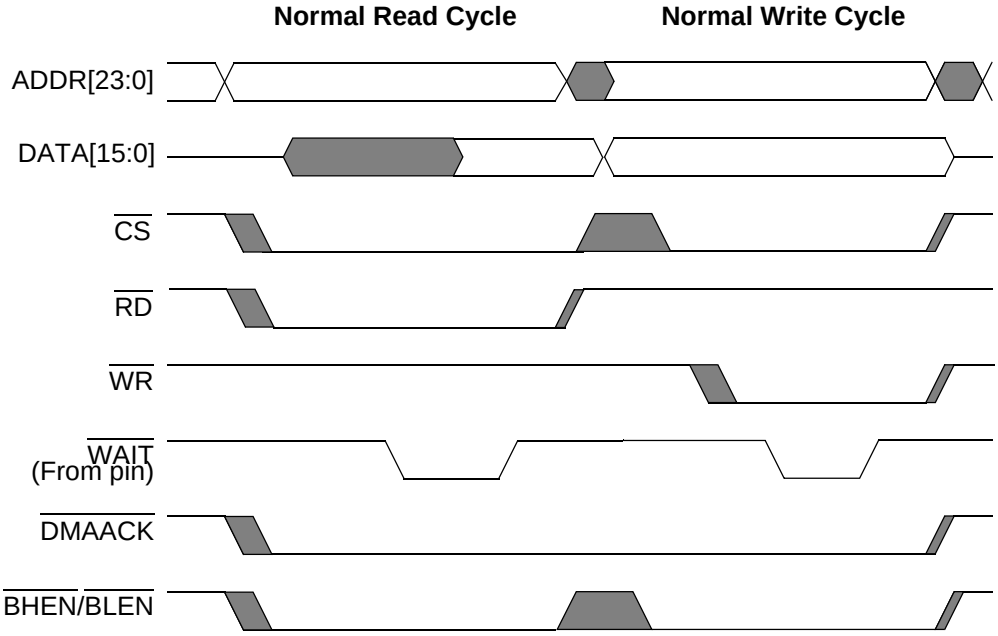

Figure 60. External DMA Transfer

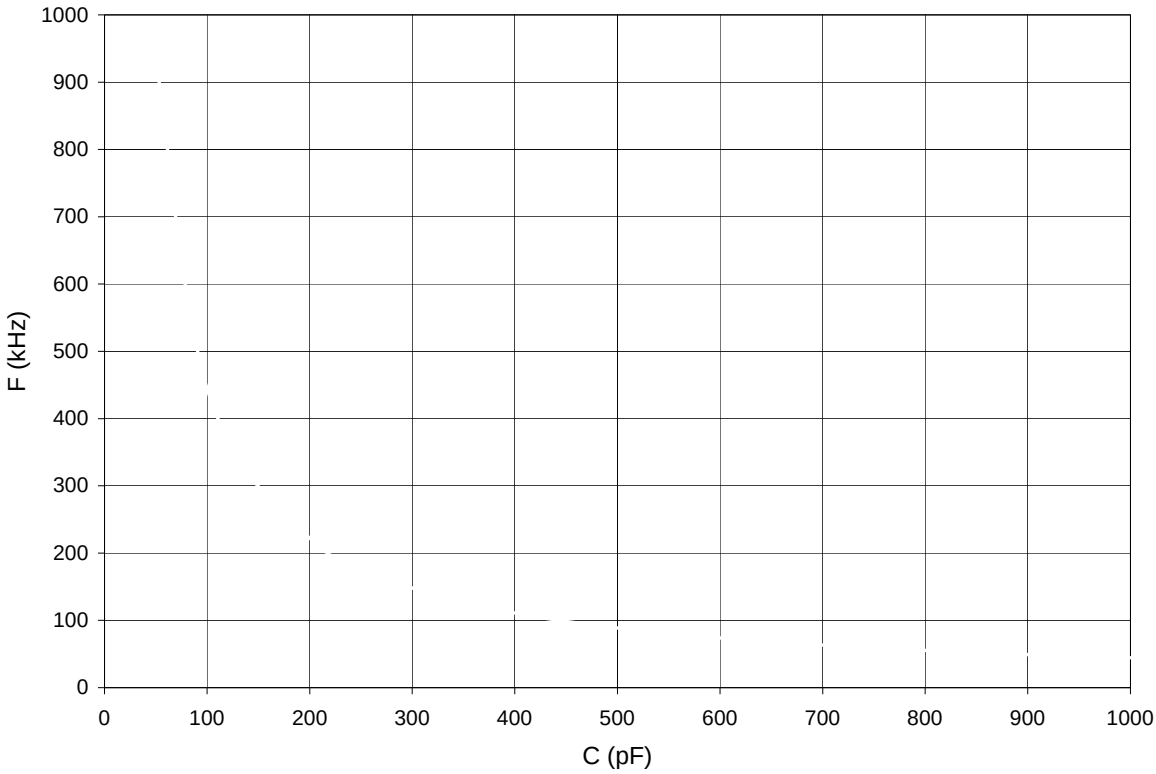

Figure 72. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 15 kΩ Resistor

Figure 73 displays the typical current consumption while operating at 3.3 V at 30 ºC versus the system clock frequency.

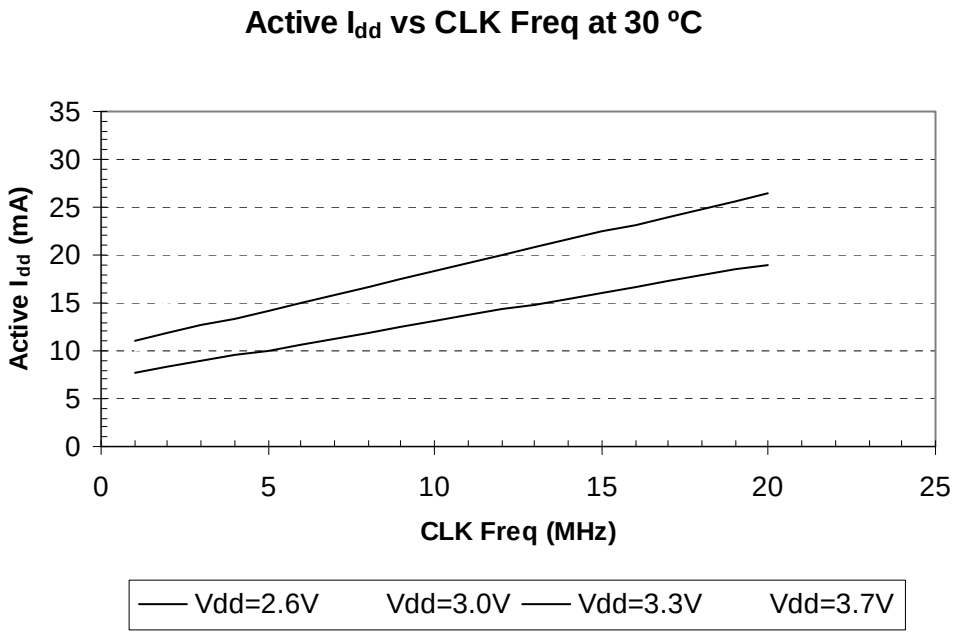

Figure 73. Typical I_{DD} Versus System Clock Frequency