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	76
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	100-LQFP
Supplier Device Package	-
Purchase URL	https://www.e-xfl.com/product-detail/zilog/z16f2811al20ag

ZNEO CPU Features

Zilog's ZNEO® CPU meets the continuing demand for faster and more code-efficient microcontrollers. The ZNEO CPU features are as follows:

- 16MB of program memory address space for object code and data with 8-bit or 16-bit data paths
- 8-bit, 16-bit and 32-bit ALU operations
- 24-bit stack with overflow protection
- Direct register-to-register architecture allows each memory address to function as an accumulator to improve execution time and decrease the required program memory
- New instructions improve execution efficiency for code developed using higher-level programming languages including 'C'
- Pipelined instructions: Fetch, Decode and Execute

For more information about the ZNEO CPU, refer to the ZNEO CPU Core User Manual (UM0188), available for download at www.zilog.com.

External Interface

The external interface allows seamless connection to external memory and peripherals. A 24-bit address bus and a selectable 8-bit or 16-bit data bus allows parallel access up to 16MB. The programmable nature of the external interface supports connection to various bus styles. More GPIO pins are utilized by controlling address and control signals bitwise.

Flash Controller

The Motor Control MCUs products contain 128 KB of internal Flash memory. The Flash controller programs and erases the Flash memory. ZNEO CPU accesses 16-bits at a time of internal Flash memory to improve the processor throughput. A sector protection scheme allows flexible protection of user code.

Random Access Memory

An internal RAM of 4 KB provides storage space for data, variables and stack operations. Like Flash memory, ZNEO CPU accesses 16-bits at a time of internal RAM to improve the processor performance.

ZNEO Peripheral Overview

The peripheral features of the ZNEO CPU are described in this section.

Signal Descriptions

Table 2 describes the ZNEO signals. To determine the signals available for the specific package styles, see the Pin Configurations section on page 7. Most of the signals described in Table 2 are multiplexed with GPIO pins. These signals are available as alternate functions on the GPIO pins. For more details about the GPIO alternate functions, see the General-Purpose Input/Output chapter on page 66.

Table 2. Signal Descriptions

Signal Mnemonic	I/O	Description
General-Purpose Input/Output Ports A–K		
PA[7:0]	I/O	Port A[7:0]: These pins are used for GPIO
PB[7:0]	I/O	Port B[7:0]: These pins are used for GPIO
PC[7:0]	I/O	Port C[7:0]: These pins are used for GPIO
PD[7:0]	I/O	Port D[7:0]: These pins are used for GPIO
PE[7:0]	I/O	Port E[7:0]: These pins are used for GPIO
PF[7:0]	I/O	Port F[7:0]: These pins are used for GPIO
PG[7:0]	I/O	Port G[7:0]: These pins are used for GPIO
PH[3:0]	I/O	Port H[3:0]: These pins are used for GPIO
PJ[7:0]	I/O	Port J[7:0]: These pins are used for GPIO
PK[7:0]	I/O	Port K[7:0]: These pins are used for GPIO
External Interface		
ADR[23:0]	O	Address bus: When the associated GPIO pins are configured for alternate function and the external interface is enabled, these pins function as output pin only. The address bus signals are driven to 0, when execution is out of internal program memory. The address bus alternate functions are individually enabled and disabled.
DATA[15:0]	I/O	Data bus: When the associated GPIO pins are configured for alternate function and the external interface is enabled, these pins functions as input/output. The data bus alternate functions are individually enabled and disabled. When write operation is not performed through the external interface, these signals are tri-stated. The data bus is enabled as either 8-bits (DATA[7:0] only) or 16-bits (DATA[15:0]).
$\overline{RD}$	O	Read output: This pin is the Read output signal from the external interface. Assertion of the $\overline{RD}$ signal indicates that the ZNEO CPU is performing a read operation from the external memory or peripheral.

Table 6. Register File Address Map (Continued)

Address (Hex)	Register Description	Mnemonic	Reset (Hex)	Page No
FF_E43A	DMA2 Source Address High	DMA2SARH	00	288
FF_E43B	DMA2 Source Address Low	DMA2SARL	00	288
FF_E43C	Reserved			
FF_E43D	DMA2 List Address Upper	DMA2LARU	00	289
FF_E43E	DMA2 List Address High	DMA2LARH	00	289
FF_E43F	DMA2 List Address Low	DMA2LARL	00	289
DMA Channel 3 Base Address = FF_E440				
FF_E440	DMA3 Control 0	DMA3CTL0	00	285
FF_E441	DMA3 Control 1	DMA3CTL1	00	285
FF_E442	DMA3 Transfer Length High	DMA3TXLNH	00	286
FF_E443	DMA3 Transfer Length Low	DMA3TXLNL	00	287
FF_E444	Reserved	—	—	—
FF_E445	DMA3 Destination Address Upper	DMA3DARU	00	287
FF_E446	DMA3 Destination Address High	DMA3DARH	00	287
FF_E447	DMA3 Destination Address Low	DMA3DARL	00	287
FF_E448	Reserved	—	—	—
FF_E449	DMA3 Source Address Upper	DMA3SARU	00	288
FF_E44A	DMA3 Source Address High	DMA3SARH	00	288
FF_E44B	DMA3 Source Address Low	DMA3SARL	00	288
FF_E44C	Reserved	—	—	—
FF_E44D	DMA3 List Address Upper	DMA3LARU	00	289
FF_E44E	DMA3 List Address High	DMA3LARH	00	289
FF_E44F	DMA3 List Address Low	DMA3LARL	00	289
Analog Block Base Address = FF_E500				
ADC Base Address = FF_E500				
FF_E500	ADC0 Control Register	ADC0CTL	00	246
FF_E501	Reserved	—	—	—
FF_E502	ADC0 Data High Byte Register	ADC0D_H	XX	247
FF_E503	ADC0 Data Low Bits Register	ADC0D_L	XX	248

XX = Undefined.

Table 12 shows the External Chip Select Control Registers Low for $\overline{CS1}$(EXTSC1L). This register sets the number of wait states for Chip Select 1. Waits are only added if the chip select is enabled.

Table 12. External Chip Select Control Registers Low for $\overline{CS1}$ (EXTCS1L)

Bits	7	6	5	4	3	2	1	0
Field	RESERVED		PR1WAIT		CS1WAIT			
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_(E075)H							

Bit	Description
[7:6]	**Reserved** These bits are reserved and must be programmed to 00.
[5:4] PR1WAIT[2:0]	**Post Read Wait Selection** 00 = 0 wait state. 01 = 1 wait state. 10 = 2 wait states. 11 = 3 wait states.
[3:0] CS1WAIT	**Chip Select 1 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 states. 1011 = 11 wait states. 1100 = 12 wait states. 1101 = 13 wait states. 1110 = 14 wait states. 1111 = 15 wait states.

GPIO Control Register Definitions

Port A-K Input Data Registers

Reading from the Port A-K Input Data registers, shown in Table 25, returns the sampled values from the corresponding port pins. The Port A-K Input Data registers are Read-only.

Table 25. Port A-K Input Data Registers (PxIN)

Bits	7	6	5	4	3	2	1	0
Field	PIN7	PIN6	PIN5	PIN4	PIN3	PIN2	PIN1	PIN0
RESET	X	X	X	X	X	X	X	X
R/W	R	R	R	R	R	R	R	R
Addr	FF_E100, FF_E110, FF_E120, FF_E130, FF_E140, FF_E150, FF_E160, FF_E170, FF_E180, FF_E190							

Bit	Description
[7:0] PIN[7:0]	**Port Input Data** Sampled data from the corresponding port pin input. 0 = Input data is logical 0 (Low). 1 = Input data is logical 1 (High).

5. Enable the timer interrupt, if required and set the timer interrupt priority by writing to the relevant interrupt registers.
6. Configure the associated GPIO port pin(s) for the timer output alternate function.
7. Write to the Timer Control 1 Register to enable the timer and initiate counting.

The PWM period is determined by the following equation:

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

If an initial starting value other than 0001h is loaded into the Timer High and Low Byte registers, use the One-Shot Mode equation to determine the first PWM time-out period.

If TPOL is set to 0, the ratio of the PWM output High time to the total period is determined by:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{Reload Value} - \text{PWM Value}}{\text{Reload Value}} \times 100$$

If TPOL is set to 1, the ratio of the PWM output High time to the total period is determined by:

$$\text{PWM Output High Time Ratio (\%)} = \frac{\text{PWM Value}}{\text{Reload Value}} \times 100$$

Capture Modes

There are three Capture modes which provide slightly different methods for recording the time or time interval between timer input events. These modes are Capture Mode, Capture Restart Mode and Capture Compare Mode. In all of the three modes, when the appropriate timer input transition (capture event) occurs, the timer counter value is captured and stored in the PWM High and Low Byte registers. The TPOL bit in the Timer Control 1 Register determines if the Capture occurs on a rising edge or a falling edge of the timer input signal. The TICONFIG bit determines whether interrupts are generated on capture events, reload events or both. The INCAP bit in Timer Control 0 Register clears to indicate an interrupt caused by a reload event and sets to indicate the timer interrupt is caused by an input capture event.

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. The initial value is determined by the TPOL bit.

Bit	Description (Continued)
[3] ADCTRIG	**ADC Trigger Enable** 0 = No ADC trigger pulses. 1 = ADC trigger enabled.
[2]	**Reserved** This bit is reserved and must be programmed to 0.
[1] READY	**Values Ready for Next Reload Event** 0 = PWM values (prescale, period and duty cycle) are not ready. Do not use values in holding registers at next PWM reload event. 1 = PWM values (prescale, period and duty cycle) are ready. Transfer all values from temporary holding registers to working registers at next PWM reload event.
[0] PWMEN	**PWM Enable** 0 = PWM is disabled and enabled PWM output pins are forced to default off-state. PWM master counter is stopped. Certain control registers may only be written in this state. 1 = PWM is enabled and PWM output pins are enabled as outputs.

PWM Control 1 Register

The PWM Control 1 (PWMCTL1) Register controls portions of PWM operation.

Table 71. PWM Control 1 Register (PWMCTL1)

Bits	7	6	5	4	3	2	1	0
Field	RLFREQ[1:0]		INDEN	POL45	POL23	POL10	PRES[1:0]	
RESET	00		0	0	0	0	00	
R/W	R/W		R/W	R/W	R/W	R/W	R/W	
Addr	FF_E381h							

Bit	Description
[7:6] RLFREQ[1:0]	**Reload Event Frequency** This bit field is buffered. Changes to the reload event frequency takes effect at the end of the current PWM period. Reads always return the bit values from the temporary holding register. 00 = PWM reload event occurs at the end of every PWM period. 01 = PWM reload event occurs once every two PWM periods. 10 = PWM reload event occurs once every four PWM periods. 11 = PWM reload event occurs once every eight PWM periods.
[5] INDEN	**Independent PWM Mode Enable** 0 = PWM outputs operate as three complementary pairs. 1 = PWM outputs operate as six independent channels.

Bit	Description (Continued)
[4] POL2	**PWM2 Polarity** 0 = Noninverted polarity for channel pair PWM2. 1 = Invert output polarity for channel pair PWM2.
[3] POL1	**PWM1 Polarity** 0 = Noninverted polarity for channel pair PWM1. 1 = Invert output polarity for channel pair PWM1.
[2] POL0	**PWM0 Polarity** 0 = Noninverted polarity for channel pair PWM0. 1 = Invert output polarity for channel pair PWM0.
[1:0] PRES	**PWM Prescaler** The prescaler divides down the PWM input clock (either the system clock or the PWMIN external input). This field is buffered. Changes to this field take effect at the next PWM reload event. Reads always return the values from the temporary holding register. 00 = Divide by 1. 01 = Divide by 2. 10 = Divide by 4. 11 = Divide by 8.

PWM Deadband Register

The PWM Deadband (PWMDB) Register, shown in Table 72, stores the 8-bit PWM deadband value. The deadband value determines the number of PWM input cycles to use for the deadband time for complementary PWM output pairs. When counting PWM input cycles, the PWM input signal is used directly (no prescaler). The minimum deadband value is 1. Maximum deadband time is programmed by setting a value of 00h.

Table 72. PWM Deadband Register (PWMDB)

Bits	7	6	5	4	3	2	1	0
Field	PWMDB[7:0]							
RESET	01h							
R/W	R/W							
Addr	FF_E382h							

Bit	Description
[7:0] PWMDB[7:0]	**PWM Deadband** Sets the PWM deadband period for which both PWM outputs of a complementary PWM output pair are deasserted.

Note: This register can only be written when PWMEN is cleared.

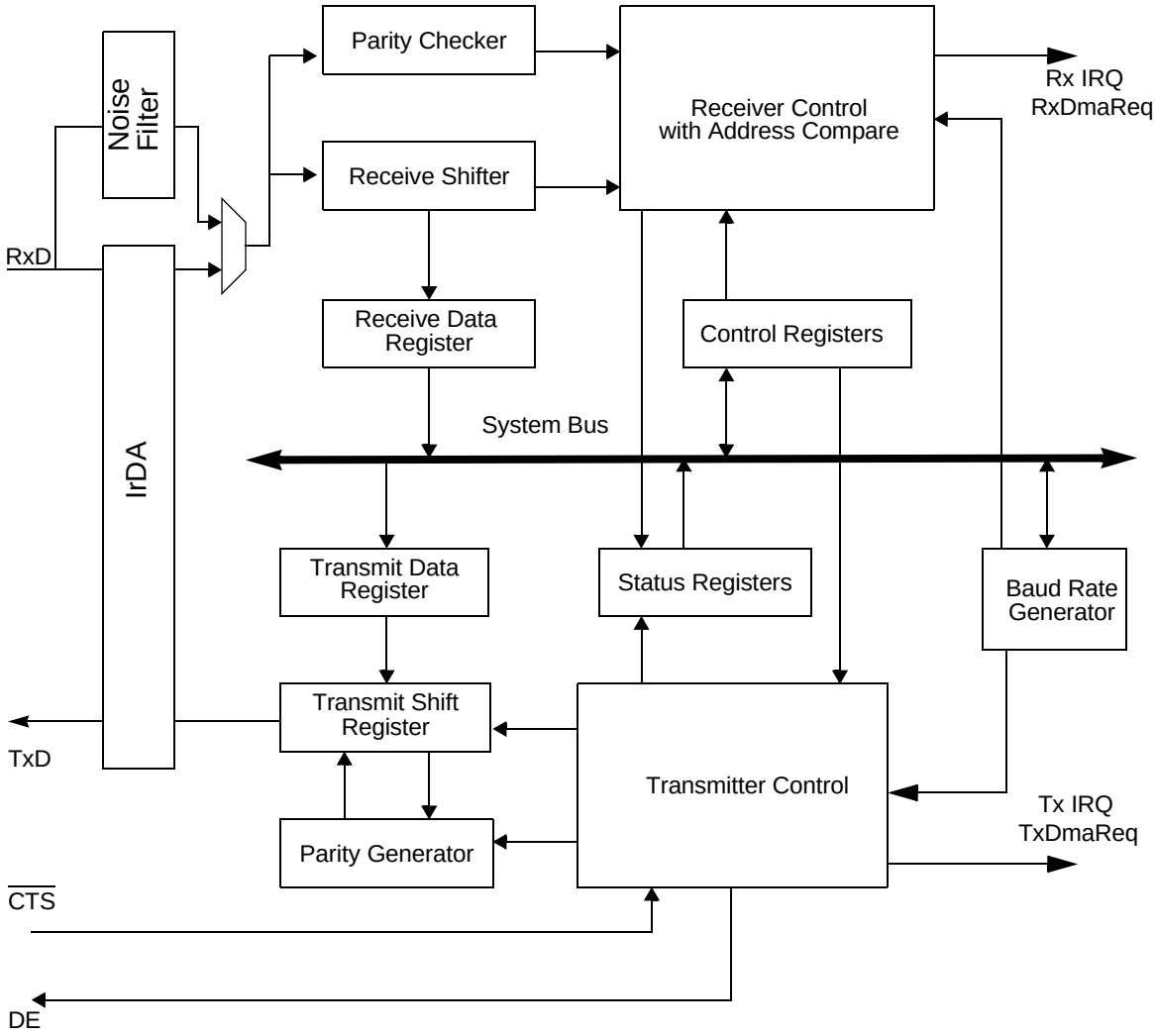

Figure 24. LIN-UART Block Diagram

Operation

Data Format for Standard UART Modes

The LIN-UART always transmits and receives data in an 8-bit data format, with the first bit being the least-significant bit. An even- or odd-parity bit or multiprocessor address/ data bit is optionally added to the data stream. Each character begins with an active Low start bit and ends with either 1 or 2 active High stop bits. Figures 25 and 26 display the

obtained by reading the RxBreakLength field in the mode status register. After a Wake-up message is detected, the LIN-UART is placed (by software) into either Lin Master or Lin Slave Wait for Break states as appropriate. If the break duration exceeds fifteen bit times, the RxBreakLength field contains the value Fh.

Lin Sleep state is selected by software setting LinState[1:0] = 00. The decision to move from an active state to sleep state is based on the LIN messages as interpreted by the software.

LIN Slave Operation

LIN Slave Mode is selected by setting the bits LMST = 0, LSLV = 1, ABEN = 1 or 0 and LinState[1:0] = 01b (Wait for Break state). The LIN slave detects the start of a new message by the Break which appears to the slave as a break of at least 11 bit times in duration. The LIN-UART detects the Break and generates an interrupt to the CPU. The duration of the Break is observable in the RxBreakLength field of the mode status register. A Break of less than 11 bit times in duration does not generate a break interrupt when the LIN-UART is in a Wait for Break state. If the Break duration exceeds 15 bit times, the RxBreakLength field contains the value Fh.

Following the Break the LIN-UART hardware automatically transits to the autobaud state, where it autobauds by timing the duration of the first 8 bit times of the Synch character as defined in the standard. At the end of the autobaud period, the duration measured by the BRG counter (auto baud period divided by 8) is automatically transferred to the baud reload high and low registers if the ABEN bit of the LIN Control Register is set. If the BRG counter overflows before reaching the start of bit 7 in the autobaud sequence the autobaud overrun error interrupt occurs, the OE bit in the Status 0 Register is set and the baud reload registers are not updated. To autobaud within 2% of the master's baud rate, the slave system clock must be minimum 100 times the baud rate. To avoid an autobaud overrun error, the system clock must not be greater than 2^{19} times the baud rate (16 bit counter following 3-bit prescaler when counting the 8 bit times of the autobaud sequence).

Following the Synch character, the LIN-UART hardware transits to the active state, where the Identifier character is received and the characters of the response section of the message are sent or received. The slave remains in the active state until a Break is received or the software forces a state change. When it is in active State (autobaud has completed), a Break of 10 or more bit times is recognized and a transition to the autobaud state is caused.

LIN-UART Interrupts

The LIN-UART features separate interrupts for the transmitter and receiver. In addition, when the LIN-UART primary functionality is disabled, the BRG also functions as a basic timer with interrupt capability.

byte indicating an overrun error, the Receive Data Register must be read again to clear the error bits in the LIN-UART Status 0 Register.

In LIN Mode, an overrun error is signaled for receive data overruns as described above and in the LIN Slave, if the BRG counter overflows during the autobaud sequence (the ATB bit will also be set in this case). There is no data associated with the autobaud overflow interrupt, however the Receive Data Register must be read to clear the OE bit. In this case software must write `10b` to the LinState field, forcing the LIN slave back to Wait for Break state.

LIN-UART Data and Error Handling Procedure

Figure 29 displays the recommended procedure for use in LIN-UART receiver interrupt service routines.

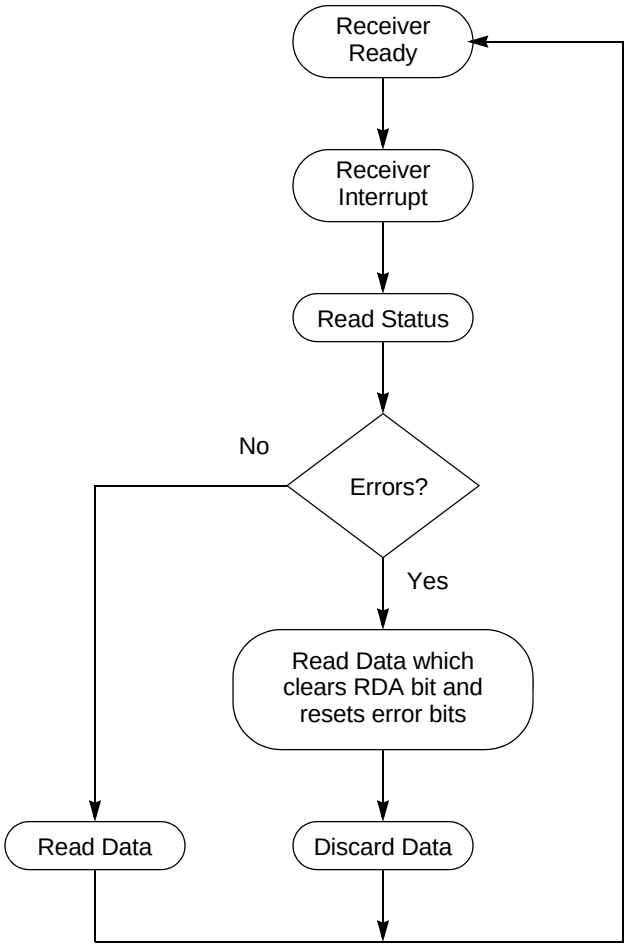

Figure 29. LIN-UART Receiver Interrupt Service Routine Flow

Table 95. LIN-UART Baud Rate Low Byte Register (UxBRL)

Bits	7	6	5	4	3	2	1	0
Field	BRL							
RESET	1							
R/W	R/W							
Addr	FF–E207h, FF–E217h							

The LIN-UART data rate is calculated using the following equation for standard UART modes. For LIN protocol, the baud rate registers must be programmed with the baud period rather than 1/16 baud period.

Note: The UART must be disabled when updating the baud rate registers because High and Low registers must be written independently.

The LIN-UART data rate is calculated using the following equation for standard UART operation:

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

The LIN-UART data rate is calculated using the following equation for LIN Mode UART operation:

$$\text{UART Data Rate (bits/s)} = \frac{\text{System Clock Frequency (Hz)}}{\text{UART Baud Rate Divisor Value}}$$

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for standard UART operation:

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

For a given LIN-UART data rate, the integer baud rate divisor value is calculated using the following equation for LIN Mode UART operation:

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

S	Slave Address 1st Byte	W=0	A	Slave Address 2nd Byte	A	Data	A	Data	A/$\overline{A}$	P/S

Figure 49. Data Transfer Format, Slave Receive Transaction with 10-Bit Addressing

1. Software configures the controller for operation as a Slave in 10-Bit Addressing Mode as follows.
   a. Initialize the MODE field in the I2CMODE Register for either Slave Only Mode or Master/Slave Mode with 10-Bit Addressing.
   b. Optionally set the GCE bit.
   c. Initialize the SLA[7:0] bits in the I2CSLVAD Register and the SLA[9:8] bits in the I2CMODE Register.
   d. Set IEN = 1 in the I2CCTL Register. Set NAK = 0 in the I^2C Control Register.
   e. Program the Baud Rate High and Low Byte registers for the I^2C baud rate.
2. The Master initiates a transfer by sending the first address byte. The I^2C Controller recognizes the start of a 10-bit address with a match to SLA[9:8] and detects the R/$\overline{W}$ bit = 0 (write from Master to Slave). The I^2C Controller acknowledges, indicating that it is available to accept the transaction.
3. The Master sends the second address byte. The Slave Mode I^2C Controller detects an address match between the second address byte and SLA[7:0]. The SAM bit in the I2CISTAT Register is set = 1, causing an interrupt. The RD bit is set = 0, indicating a write to the Slave. The I^2C Controller Acknowledges, indicating it is available to accept the data.
4. Software responds to the interrupt by reading the I2CISTAT Register, which clears the SAM bit. When RD = 0, no immediate action is taken by software 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.
5. The Master detects the Acknowledge and sends the first byte of data.
6. The I^2C controller receives the first byte and responds with Acknowledge or Not Acknowledge, depending on the state of the NAK bit in the I2CCTL Register. The I^2C controller generates the receive data interrupt by setting the RDRF bit in the I2CISTAT Register.
7. Software responds by reading the I2CISTAT Register, finding the RDRF bit = 1 and then reading the I2CDATA Register, which clears the RDRF bit. If software accepts only one more data byte, it sets the NAK bit in the I2CCTL Register.
8. The Master and Slave loops on steps 5–7 until the Master detects a Not Acknowledge instruction or runs out of data to send.

Bit	Description (Continued)
[2:1] SLA[9:8]	**Slave Address Bits 9 and 8** Initialize with the appropriate Slave address value when using 10-bit Slave addressing. These bits are ignored when using 7-bit Slave addressing.
[0] DIAG	**Diagnostic Mode** Selects read back value of the Baud Rate Reload and State registers. 0 = Reading the Baud Rate registers returns the Baud Rate register values. Reading the State Register returns I2C Controller state information. 1 = Reading the Baud Rate registers returns the current value of the baud rate counter. Reading the State Register returns additional state information.

I2C Slave Address Register

The I2C Slave Address Register, shown in Table 122, provides control over the lower order address bits used in 7-bit and 10-bit Slave address recognition.

Table 122. I2C Slave Address Register (I2CSLVAD)

Bits	7	6	5	4	3	2	1	0
Field	SLA[7:0]							
RESET	00h							
R/W	R/W							
Addr	FF–E247h							

Bit	Description
[7:0] SLA[7:0]	**Slave Address Bits 7–0** Initialize with the appropriate Slave address value. When using 7 bit Slave addressing, SLA[9:7] are ignored.

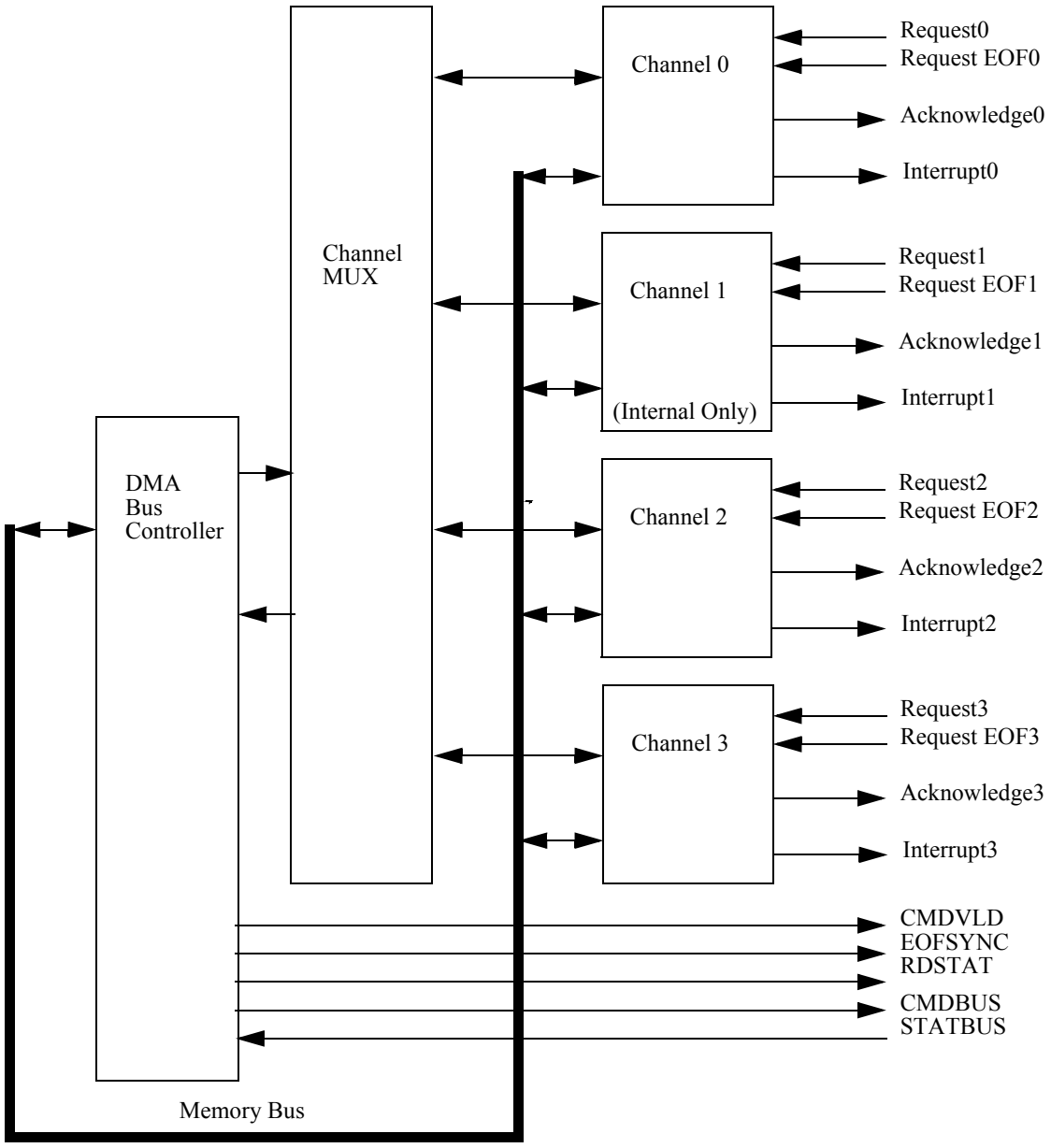

Figure 56. DMA Block Diagram

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

Read PC. The Read Program Counter command returns the contents of the program counter.

```
DBG <-- 0000_0110
DBG --> 00h
DBG --> PC[23:16]
DBG --> PC[15:8]
DBG --> PC[7:0]
DBG --> CRC[0:7]
```

Write PC. The Write Program Counter command writes data to the program counter.

```
DBG <-- 0000_0111
DBG <-- 00h
DBG <-- PC[23:16]
DBG <-- PC[15:8]
DBG <-- PC[7:0]
DBG --> CRC[0:7]
```

Read Flags. The Read Flags command returns the contents of the CPU flags.

```
DBG <-- 0000_1000
DBG --> 00h
DBG --> flags[7:0]
DBG --> CRC[0:7]
```

Write Instruction. The Write Instruction command writes one word of Op Code to the CPU.

```
DBG <-- 0000_1001
DBG <-- opcode[15:8]
DBG <-- opcode[7:0]
DBG --> CRC[0:7]
```

Read Register. The Read Register command returns the contents of a single CPU register.

```
DBG <-- {0100,regno[3:0]}
DBG --> regdata[31:24]
DBG --> regdata[23:16]
DBG --> regdata[15:8]
DBG --> regdata[7:0]
DBG --> CRC[0:7]
```

Write Register. The Write Register command writes data to a single CPU register.

```
DBG <-- {0101,regno[3:0]}
DBG <-- regdata[31:24]
DBG <-- regdata[23:16]
DBG <-- regdata[15:8]
DBG <-- regdata[7:0]
DBG --> CRC[0:7]
```

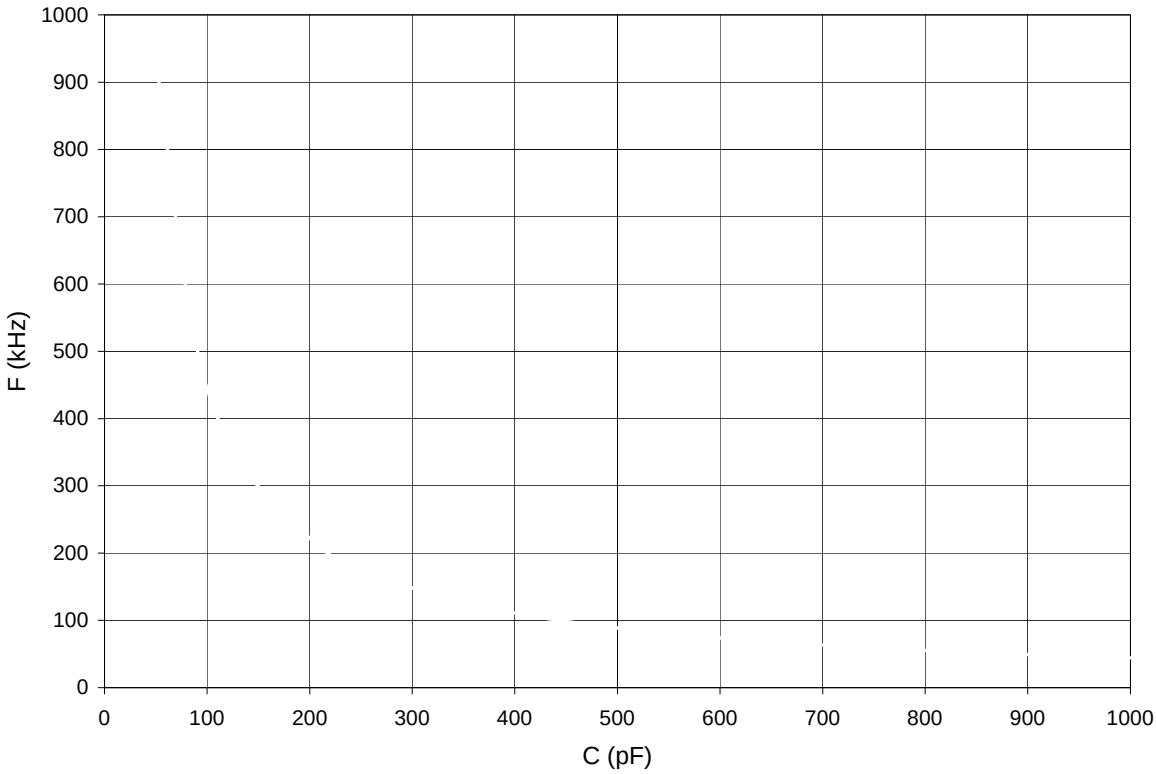

Figure 72. Typical RC Oscillator Frequency as a Function of the External Capacitance with a 15 kΩ Resistor

Figure 74 displays typical current consumption while operating at 3.3 V at 30ºC in Halt Mode versus the system clock frequency.

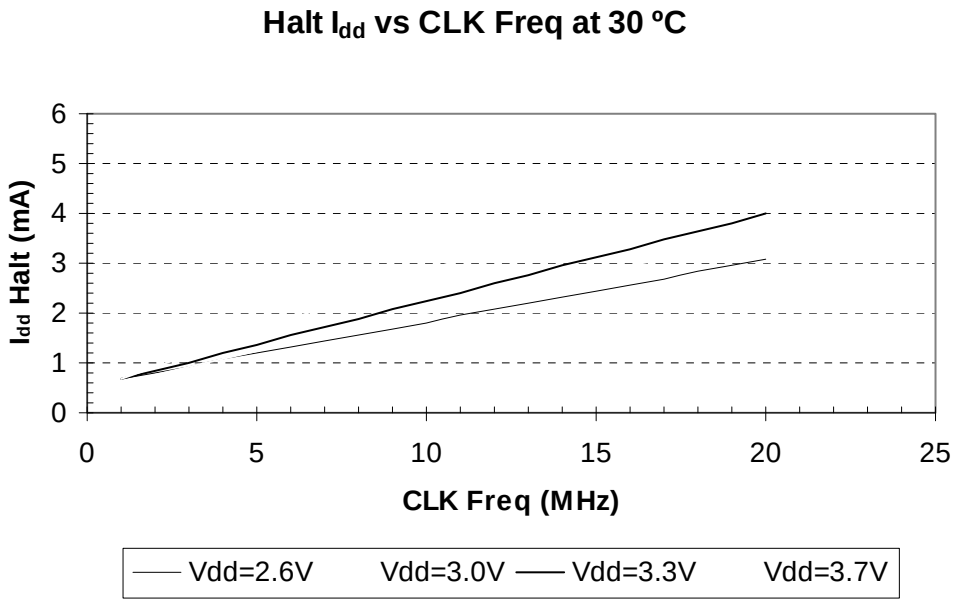

Figure 74. Typical Halt Mode I_{DD} Versus System Clock Frequency