Welcome to E-XFL.COM

Embedded - Microcontrollers - Application Specific: Tailored Solutions for Precision and Performance

Embedded - Microcontrollers - Application Specific represents a category of microcontrollers designed with unique features and capabilities tailored to specific application needs. Unlike general-purpose microcontrollers, application-specific microcontrollers are optimized for particular tasks, offering enhanced performance, efficiency, and functionality to meet the demands of specialized applications.

What Are Embedded - Microcontrollers - Application Specific?

Application-specific microcontrollers are engineered to

Details	
Product Status	Obsolete
Applications	USB Microcontroller
Core Processor	M8B
Program Memory Type	OTP (3kB)
Controller Series	CY7C632xx
RAM Size	96 x 8
Interface	USB
Number of I/O	8
Voltage - Supply	3.5V ~ 5.5V
Operating Temperature	0°C ~ 70°C
Mounting Type	Through Hole
Package / Case	16-DIP (0.300", 7.62mm)
Supplier Device Package	16-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/infineon-technologies/cy7c63221a-pxc

TABLE OF CONTENTS

1.0 Features

- enCoRe™ USB - enhanced Component Reduction
  - Internal oscillator eliminates the need for an external crystal or resonator
  - Interface can auto-configure to operate as PS/2 or USB without the need for external components to switch between modes (no GPIO pins needed to manage dual mode capability)
  - Internal 3.3V regulator for USB pull-up resistor
  - Configurable GPIO for real-world interface without external components
- Flexible, cost-effective solution for applications that combine PS/2 and low-speed USB, such as mice, gamepads, joysticks, and many others
- USB Specification Compliance
  - Conforms to USB Specification, Version 2.0
  - Conforms to USB HID Specification, Version 1.1
  - Supports 1 low-speed USB device address
  - Supports 1 control endpoint and 1 data endpoint
  - Integrated USB transceiver
  - 3.3V regulated output for USB pull-up resistor
- 8-bit RISC microcontroller
  - Harvard architecture
  - 6-MHz external ceramic resonator or internal clock mode
  - 12-MHz internal CPU clock
  - Internal memory
  - 96 bytes of RAM
  - 3 Kbytes of EPROM
  - Interface can auto-configure to operate as PS/2 or USB
  - No external components for switching between PS/2 and USB modes
- I/O ports
  - Up to 10 versatile General Purpose I/O (GPIO) pins, individually configurable
  - High current drive on any GPIO pin: 50 mA/pin current sink
  - Each GPIO pin supports high-impedance inputs, internal pull-ups, open drain outputs, or traditional CMOS outputs
  - Maskable interrupts on all I/O pins
  - XTALIN, XTALOUT and VREG can be configured as additional input pins
- Internal low-power wake-up timer during suspend mode
  - Periodic wake-up with no external components
- Optional 6-MHz internal oscillator mode
  - Allows fast start-up from suspend mode
- Watchdog timer (WDT)
- Low-voltage Reset at 3.75V
- Internal brown-out reset for suspend mode
- Improved output drivers to reduce EMI
- Operating voltage from 4.0V to 5.5VDC
- Operating temperature from 0 to 70 degrees Celsius
- available in DIE form or 16-pin PDIP
- available in 18-pin SOIC, 18-pin PDIP
- Industry-standard programmer support

2.0 Functional Overview

2.1 *enCoRe* USB - The New USB Standard

Cypress has reinvented its leadership position in the low-speed USB market with a new family of innovative microcontrollers. Introducing...*enCoRe*™ USB—"enhanced Component Reduction." Cypress has leveraged its design expertise in USB solutions to create a new family of low-speed USB microcontrollers that enables peripheral developers to design new products with a minimum number of components. At the heart of the Cypress *enCoRe* USB technology is the breakthrough design of a crystal-less oscillator. By integrating the oscillator into the chip, an external crystal or resonator is no longer needed. We have also integrated other external components commonly found in low-speed USB applications such as pull-up resistors, wake-up circuitry, and a 3.3V regulator. All of this adds up to a lower system cost.

The family is comprised of 8-bit RISC One Time Programmable (OTP) microcontrollers. The instruction set has been optimized specifically for USB and PS/2 operations, although the microcontrollers can be used for a variety of other embedded applications.

The features up to 10 general-purpose I/O (GPIO) pins to support USB, PS/2 and other applications. The I/O pins are grouped into two ports (Port 0 to 1) where each pin can be individually configured as inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs with programmable drive strength of up to 50 mA output drive. Additionally, each I/O pin can be used to generate a GPIO interrupt to the microcontroller.

The microcontrollers feature an internal oscillator. With the presence of USB traffic, the internal oscillator can be set to precisely tune to USB timing requirements (6 MHz ±1.5%). This clock generator has been optimized to reduce clock-related noise emissions (EMI), and provides the 6-MHz and 12-MHz clocks that remain internal to the microcontroller. When using the internal oscillator, XTALIN and XTALOUT can be configured as additional input pins that can be read on port 2. Optionally, an external 6-MHz ceramic resonator can be used to provide a higher precision reference if needed.

The is offered with 3 Kbytes of EPROM to minimize cost, and has 96 bytes of data RAM for stack space, user variables, and USB endpoint FIFOs.

The family includes low-voltage reset logic, a watchdog timer, a vectored interrupt controller, and a 12-bit free-running timer. The low-voltage reset (LVR) logic detects when power is applied to the device, resets the logic to a known state, and begins executing instructions at EPROM address 0x0000. LVR will also reset the part when V_{CC} drops below the operating voltage range. The watchdog timer can be used to ensure the firmware never gets stalled for more than approximately 8 ms.

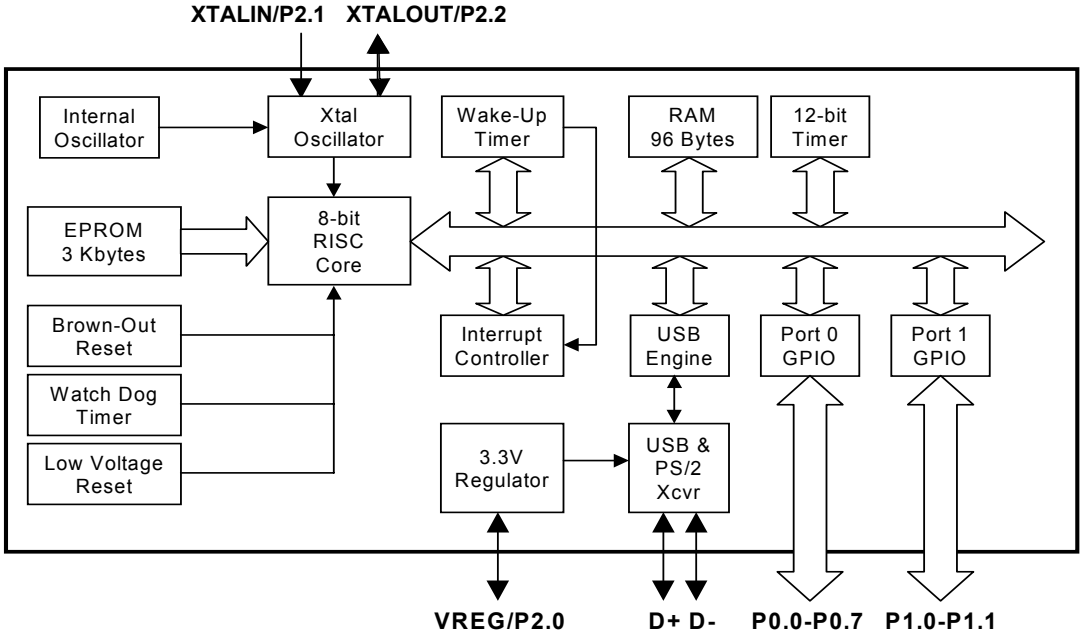
The microcontroller supports 7 maskable interrupts in the vectored interrupt controller. Interrupt sources include the USB Bus-Reset, the 128-µs and 1.024-ms outputs from the free-running timer, two USB endpoints, an internal wake-up timer and the GPIO port. The timers bits cause periodic interrupts when enabled. The USB endpoints interrupt after USB transactions complete on the bus. The GPIO port has a level of masking to select which GPIO inputs can cause a GPIO interrupt. For additional flexibility, the input transition polarity that causes an interrupt is programmable for each GPIO pin. The interrupt polarity can be either rising or falling edge.

The free-running 12-bit timer clocked at 1 MHz provides two interrupt sources as noted above (128 µs and 1.024 ms). The timer can be used to measure the duration of an event under firmware control by reading the timer at the start and end of an event, and subtracting the two values.

The CY7C63221/31A includes an integrated USB serial interface engine (SIE). The hardware supports one USB device address with two endpoints. The SIE allows the USB host to communicate with the function integrated into the microcontroller. A 3.3V regulated output pin provides a pull-up source for the external USB resistor on the D– pin. When using an external voltage regulator VREG can be configured as an input pin that can be read on port 2 (P2.0).

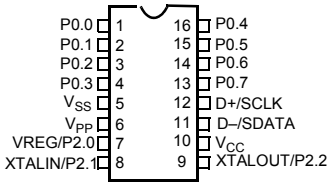
The USB D+ and D– USB pins can alternately be used as PS/2 SCLK and SDATA signals, so that products can be designed to respond to either USB or PS/2 modes of operation. PS/2 operation is supported with internal pull-up resistors on SCLK and SDATA, the ability to disable the regulator output pin, and an interrupt to signal the start of PS/2 activity. No external components are necessary for dual USB and PS/2 systems, and no GPIO pins need to be dedicated to switching between modes. Slow edge rates operate in both modes to reduce EMI.

3.0 Logic Block Diagram



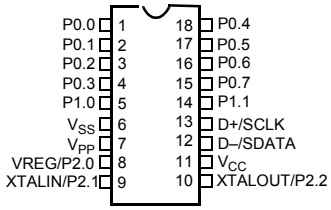
4.0 Pin Configurations

CY7C63221A
16-pin PDIP

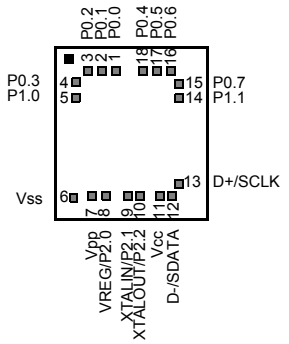


(Top View)

CY7C63231A
18-pin SOIC/PDIP



CY7C63221A-XC/XWC
DIE



5.0 Pin Assignments

Name	I/O	16-Pin	CY7C63231A/ CY7C63221A-XC 18-Pin/Pad	Description
D–/SDATA, D+/SCLK	I/O	11 12	12 13	USB differential data lines (D– and D+), or PS/2 clock and data signals (SDATA and SCLK)
P0[7:0]	I/O	1, 2, 3, 4, 13, 14, 15, 16	1, 2, 3, 4, 15, 16, 17, 18	GPIO Port 0 capable of sinking up to 50 mA/pin, or sinking controlled low or high programmable current. Can also source 2 mA current, provide a resistive pull-up, or serve as a high-impedance input.
P1[1:0]	I/O	NA	5,14	IO Port 1 capable of sinking up to 50 mA/pin, or sinking controlled low or high programmable current. Can also source 2 mA current, provide a resistive pull-up, or serve as a high-impedance input.

The call subroutine (CALL) instruction stores the program counter and flags on the program stack and increments the PSP by two.

The return from subroutine (RET) instruction restores the program counter, but not the flags, from program stack and decrements the PSP by two.

6.5 8-bit Data Stack Pointer (DSP)

The data stack pointer (DSP) supports PUSH and POP instructions that use the data stack for temporary storage. A PUSH instruction will pre-decrement the DSP, then write data to the memory location addressed by the DSP. A POP instruction will read data from the memory location addressed by the DSP, then post-increment the DSP.

During a reset, the Data Stack Pointer will be set to zero. A PUSH instruction when DSP equals zero will write data at the top of the data RAM (address 0xFF). This would write data to the memory area reserved for a FIFO for USB endpoint 0. In non-USB applications, this works fine and is not a problem.

For USB applications, the firmware should set the DSP to an appropriate location to avoid a memory conflict with RAM dedicated to USB FIFOs. Since there are only 80 bytes of RAM available (except Endpoint FIFOs) the DSP should be set between 0x00 and 0x4Fh. The memory requirements for the USB endpoints are shown in Section 8.2. For example, assembly instructions to set the DSP to 20h (giving 32 bytes for program and data stack combined) are shown below:

```
MOV A,20h     ; Move 20 hex into Accumulator (must be D8h or less to avoid USB FIFOs)
SWAP A,DSP  ; swap accumulator value into DSP register
```

6.6 Address Modes

The microcontroller supports three addressing modes for instructions that require data operands: data, direct, and indexed.

6.6.1 Data

The “Data” address mode refers to a data operand that is actually a constant encoded in the instruction. As an example, consider the instruction that loads A with the constant 0x30:

- MOV A, 30h

This instruction will require two bytes of code where the first byte identifies the “MOV A” instruction with a data operand as the second byte. The second byte of the instruction will be the constant “0xE8h”. A constant may be referred to by name if a prior “EQU” statement assigns the constant value to the name. For example, the following code is equivalent to the example shown above:

- DSPINIT: EQU 30h
- MOV A,DSPINIT

6.6.2 Direct

“Direct” address mode is used when the data operand is a variable stored in SRAM. In that case, the one byte address of the variable is encoded in the instruction. As an example, consider an instruction that loads A with the contents of memory address location 0x10h:

- MOV A, [10h]

In normal usage, variable names are assigned to variable addresses using “EQU” statements to improve the readability of the assembler source code. As an example, the following code is equivalent to the example shown above:

- buttons: EQU 10h
- MOV A,[buttons]

6.6.3 Indexed

“Indexed” address mode allows the firmware to manipulate arrays of data stored in SRAM. The address of the data operand is the sum of a constant encoded in the instruction and the contents of the “X” register. In normal usage, the constant will be the “base” address of an array of data and the X register will contain an index that indicates which element of the array is actually addressed:

- array: EQU 10h
- MOV X,3
- MOV A,[x+array]

This would have the effect of loading A with the fourth element of the SRAM “array” that begins at address 0x10h. The fourth element would be at address 0x13h.

8.0 Memory Organization

8.1 Program Memory Organization

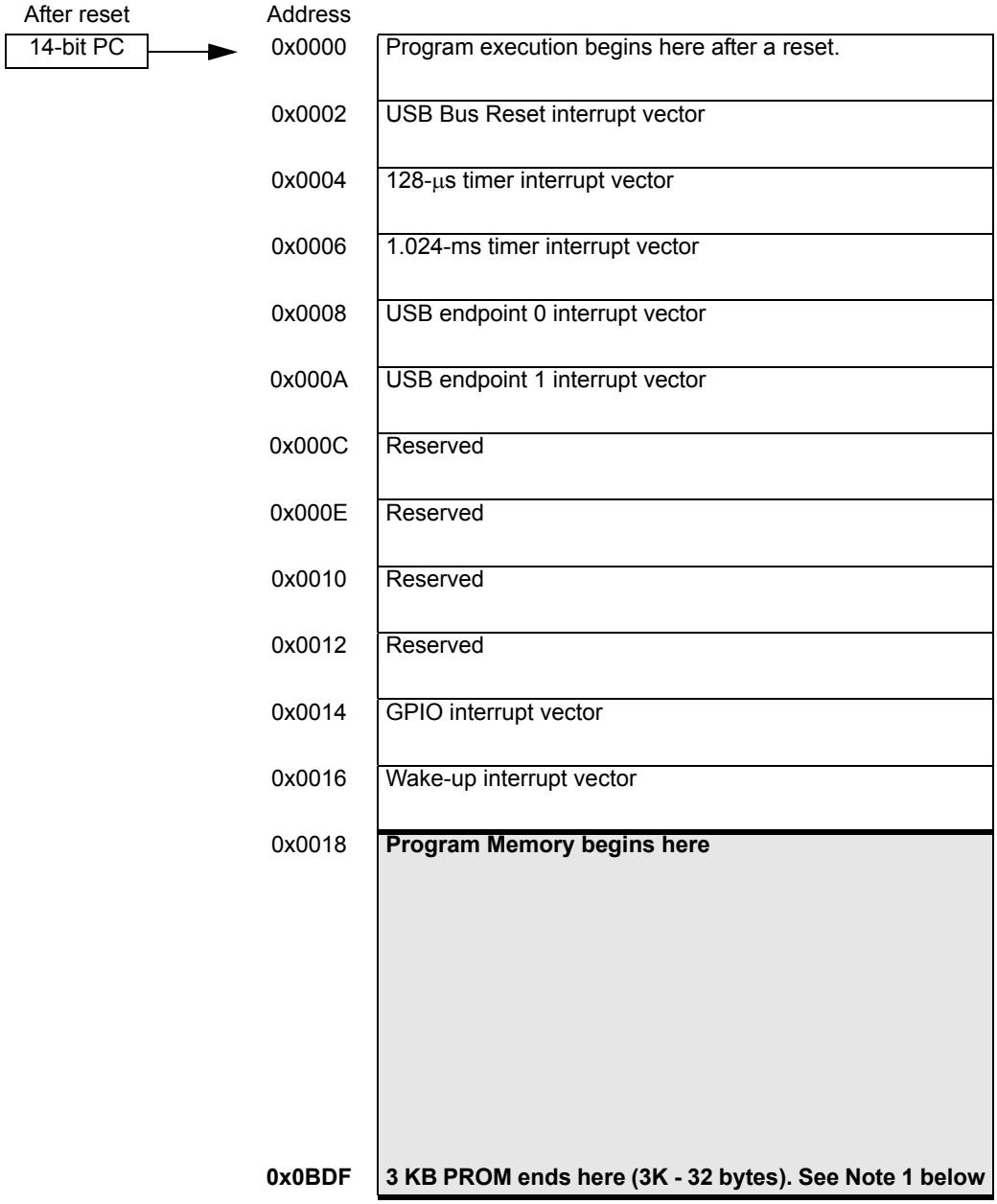
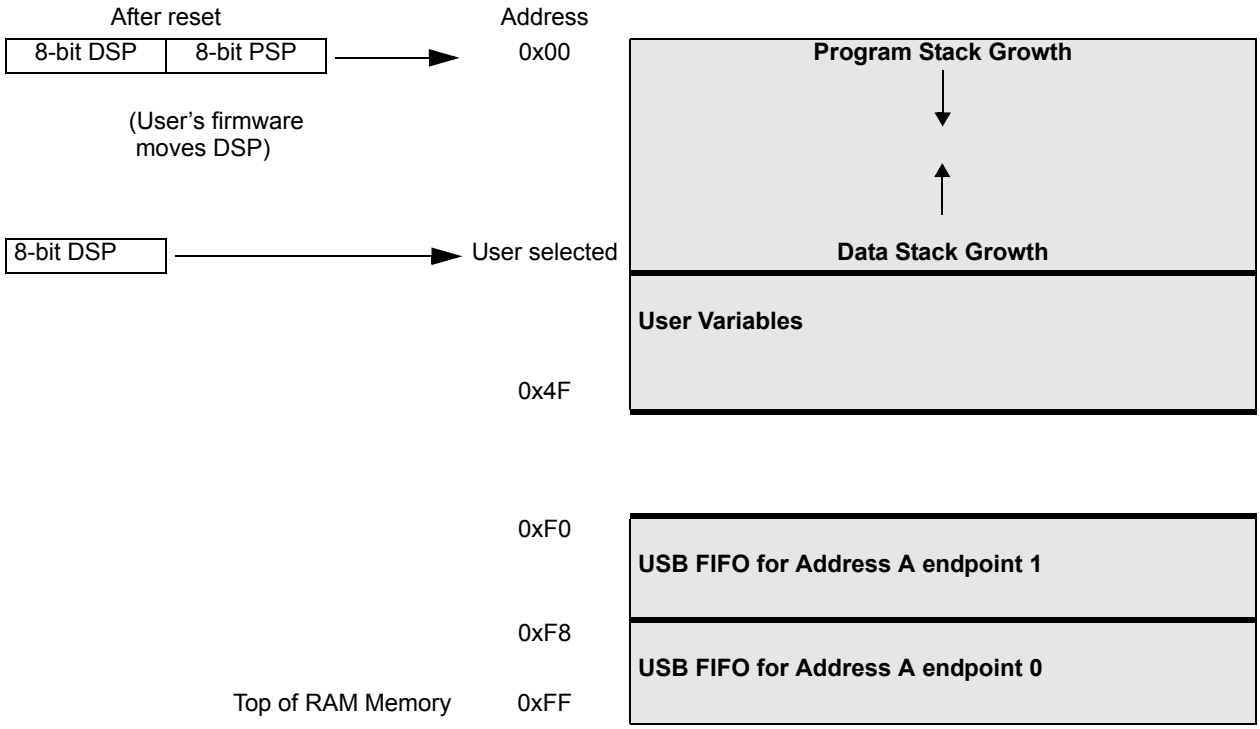


Figure 8-1. Program Memory Space with Interrupt Vector Table

Note:
1. The upper 32 bytes of the 3K PROM are reserved. Therefore, user's program must not over-write this space.

8.2 Data Memory Organization

The microcontroller provides 96 bytes of data RAM. In normal usage, the SRAM is partitioned into four areas: program stack, data stack, user variables and USB endpoint FIFOs as shown below:



9.0 Clocking

The chip can be clocked from either the internal on-chip clock, or from an oscillator based on an external resonator/crystal, as shown in *Figure 9-1*. No additional capacitance is included on chip at the XTALIN/OUT pins. Operation is controlled by the Clock Configuration Register, *Figure 9-2*.

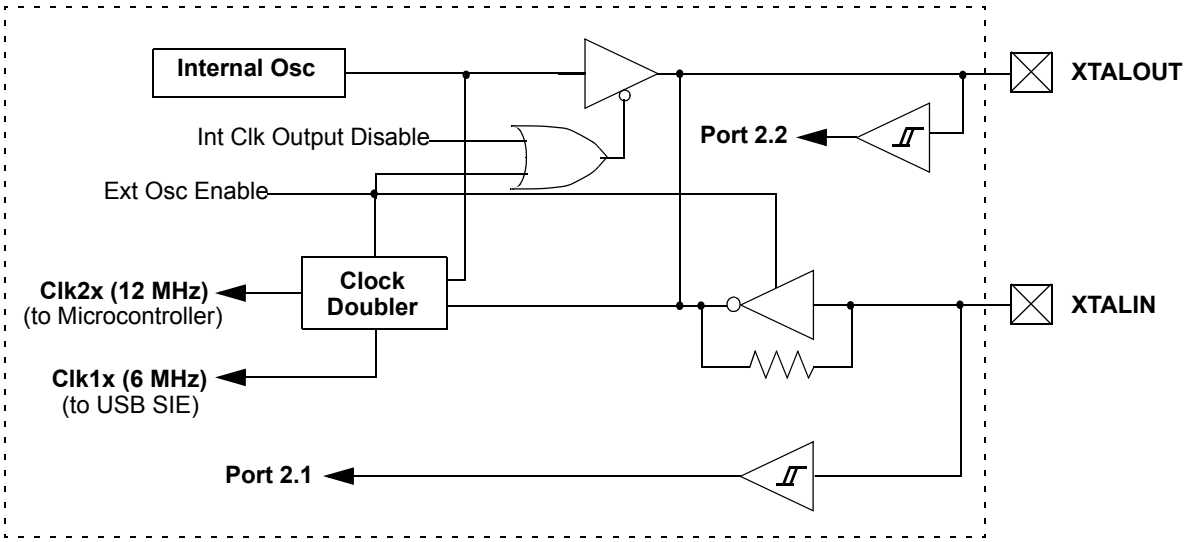


Figure 9-1. Clock Oscillator On-chip Circuit

Bit #	7	6	5	4	3	2	1	0
Bit Name	Ext. Clock Resume Delay	Wake-up Timer Adjust Bit [2:0]			Low-voltage Reset Disable	Precision USB Clocking Enable	Internal Clock Output Disable	External Oscillator Enable
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 9-2. Clock Configuration Register (Address 0xF8)

Bit 7: Ext. Clock Resume Delay

External Clock Resume Delay bit selects the delay time when switching to the external oscillator from the internal oscillator mode, or when waking from suspend mode with the external oscillator enabled.

1 = 4 ms delay.

0 = 128 μs delay.

The delay gives the oscillator time to start up. The shorter time is adequate for operation with ceramic resonators, while the longer time is preferred for start-up with a crystal. (These times **do not include** an initial oscillator start-up time which depends on the resonating element. This time is typically 50–100 μs for ceramic resonators and 1–10 ms for crystals). Note that this bit only selects the delay time for the external clock mode. When waking from suspend mode with the internal oscillator (Bit 0 is LOW), the delay time is only 8 μs in addition to a delay of approximately 1 μs for the oscillator to start.

Bit [6:4]: Wake-up Timer Adjust Bit [2:0]

The Wake-up Timer Adjust Bits are used to adjust the Wake-up timer period.

If the Wake-up interrupt is enabled in the Global Interrupt Enable Register, the microcontroller will generate wake-up interrupts periodically. The frequency of these periodical wake-up interrupts is adjusted by setting the Wake-up Timer Adjust Bit [2:0], as described in Section 11.2. One common use of the wake-up interrupts is to generate periodical wake-up events during suspend mode to check for changes, such as looking for movement in a mouse, while maintaining a low average power.

Bit 3: Low-voltage Reset Disable

When V_{CC} drops below V_{LVR} (see Section 23.0 for the value of V_{LVR}) and the Low-voltage Reset circuit is enabled, the microcontroller enters a partial suspend state for a period of t_{START} (see Section 24.0 for the value of t_{START}). Program

Start-up times for the external oscillator depend on the resonating device. Ceramic-resonator-based oscillators typically start in less than 100 μs, while crystal-based oscillators take longer, typically 1 to 10 ms. Board capacitance should be minimized on the XTALIN and XTALOUT pins by keeping the traces as short as possible.

An external 6-MHz clock can be applied to the XTALIN pin if the XTALOUT pin is left open.

10.0 Reset

The USB Controller supports three types of resets. The effects of the reset are listed below. The reset types are:

1. Low-voltage Reset (LVR)
2. Brown-out Reset (BOR)
3. Watchdog Reset (WDR)

The occurrence of a reset is recorded in the Processor Status and Control Register (*Figure 18-1*). Bits 4 (Low-voltage or Brown-out Reset bit) and 6 (Watchdog Reset bit) are used to record the occurrence of LVR/BOR and WDR respectively. The firmware can interrogate these bits to determine the cause of a reset.

The microcontroller begins execution from ROM address 0x0000 after a LVR, BOR, or WDR reset. Although this looks like interrupt vector 0, there is an important difference. Reset processing does NOT push the program counter, carry flag, and zero flag onto program stack. Attempting to execute either a RET or RETI in the reset handler will cause unpredictable execution results.

The following events take place on reset. More details on the various resets are given in the following sections.

1. All registers are reset to their default states (all bits cleared, except in Processor Status and Control Register).
2. GPIO and USB pins are set to high-impedance state.
3. The VREG pin is set to high-impedance state.
4. Interrupts are disabled.
5. USB operation is disabled and must be enabled by firmware if desired, as explained in Section 14.1.
6. For a BOR or LVR, the external oscillator is disabled and Internal Clock mode is activated, followed by a time-out period t_{START} for V_{CC} to stabilize. A WDR does not change the clock mode, and there is no delay for V_{CC} stabilization on a WDR. Note that the External Oscillator Enable (Bit 0, *Figure 9-2*) will be cleared by a WDR, but it does not take effect until suspend mode is entered.
7. The Program Stack Pointer (PSP) and Data Stack Pointer (DSP) reset to address 0x00. Firmware should move the DSP for USB applications, as explained in Section 6.5.
8. Program execution begins at address 0x0000 after the appropriate time-out period.

10.1 Low-voltage Reset (LVR)

When V_{CC} is first applied to the chip, the internal oscillator is started and the Low-voltage Reset is initially enabled by default. At the point where V_{CC} has risen above V_{LVR} (see Section 23.0 for the value of V_{LVR}), an internal counter starts counting for a period of t_{START} (see Section 24.0 for the value of t_{START}). During this t_{START} time, the microcontroller enters a partial suspend state to wait for V_{CC} to stabilize before it begins executing code from address 0x0000.

As long as the LVR circuit is enabled, this reset sequence repeats whenever the V_{CC} pin voltage drops below V_{LVR}. The LVR can be disabled by firmware by setting the Low-voltage Reset Disable bit in the Clock Configuration Register (*Figure 9-2*). In addition, the LVR is automatically disabled in suspend mode to save power. If the LVR was enabled before entering suspend mode, it becomes active again once the suspend mode ends.

When LVR is disabled during normal operation (e.g., by writing ‘0’ to the Low-voltage Reset Disable bit in the Clock Configuration Register), the chip may enter an unknown state if V_{CC} drops below V_{LVR}. Therefore, LVR should be enabled at all times during normal operation. If LVR is disabled (e.g., by firmware or during suspend mode), a secondary low-voltage monitor, BOR, becomes active, as described in the next section. The LVR/BOR Reset bit of the Processor Status and Control Register (*Figure 18-1*), is set to ‘1’ if either a LVR or BOR has occurred.

10.2 Brown-out Reset (BOR)

The Brown-out Reset (BOR) circuit is always active and behaves like the POR. BOR is asserted whenever the V_{CC} voltage to the device is below an internally defined trip voltage of approximately 2.5V. The BOR re-enables LVR. That is, once V_{CC} drops and trips BOR, the part remains in reset until V_{CC} rises above V_{LVR}. At that point, the t_{START} delay occurs before normal operation resumes, and the microcontroller starts executing code from address 0x00 after the t_{START} delay.

In suspend mode, only the BOR detection is active, giving a reset if V_{CC} drops below approximately 2.5V. Since the device is suspended and code is not executing, this lower reset voltage is safe for retaining the state of all registers and memory. Note that in suspend mode, LVR is disabled as discussed in Section 10.1.

10.3 Watchdog Reset (WDR)

The Watchdog Timer Reset (WDR) occurs when the internal Watchdog timer rolls over. Writing any value to the write-only Watchdog Reset Register at address 0x26 will clear the timer. The timer will roll over and WDR will occur if it is not cleared within t_{WATCH} (see *Figure 10-1*) of the last clear. Bit 6 (Watchdog Reset bit) of the Processor Status and Control Register is set to record this event (see Section 18.0 for more details). A Watchdog Timer Reset lasts for typically 2–4 ms after which the microcontroller begins execution at ROM address 0x0000.

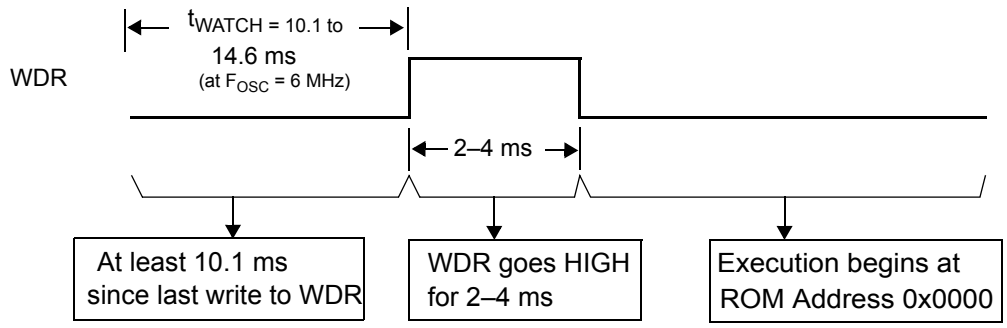


Figure 10-1. Watchdog Reset (WDR, Address 0x26)

11.0 Suspend Mode

The parts support a versatile low-power suspend mode. In suspend mode, only an enabled interrupt or a LOW state on the D–/SDATA pin will wake the part. Two options are available. For lowest power, all internal circuits can be disabled, so only an external event will resume operation. Alternatively, a low-power internal wake-up timer can be used to trigger the wake-up interrupt. This timer is described in Section 11.2, and can be used to periodically poll the system to check for changes, such as looking for movement in a mouse, while maintaining a low average power.

The is placed into a low-power state by setting the Suspend bit of the Processor Status and Control Register (*Figure 18-1*). All logic blocks in the device are turned off except the GPIO interrupt logic, the D–/SDATA pin input receiver, and (optionally) the wake-up timer. The clock oscillators, as well as the free-running and watchdog timers are shut down. Only the occurrence of an enabled GPIO interrupt, wake-up interrupt, SPI slave interrupt, or a LOW state on the D–/SDATA pin will wake the part from suspend (D– LOW indicates non-idle USB activity). Once one of these resuming conditions occurs, clocks will be restarted and the device returns to full operation after the oscillator is stable and the selected delay period expires. This delay period is determined by selection of internal vs. external clock, and by the state of the Ext. Clock Resume Delay as explained in Section 9.0.

In suspend mode, any enabled and pending interrupt will wake the part up. The state of the Interrupt Enable Sense bit (Bit 2, *Figure 18-1*) does not have any effect. As a result, any interrupts not intended for waking from suspend should be disabled through the Global Interrupt Enable Register and the USB End Point Interrupt Enable Register (Section 19.0).

If a resuming condition exists when the suspend bit is set, the part will still go into suspend and then awake after the appropriate delay time. The Run bit in the Processor Status and Control Register must be set for the part to resume out of suspend.

Once the clock is stable and the delay time has expired, the microcontroller will execute the instruction following the I/O write that placed the device into suspend mode before servicing any interrupt requests.

To achieve the lowest possible current during suspend mode, all I/O should be held at either V_{CC} or ground. In addition, the GPIO *bit* interrupts (*Figure 19-4* and *Figure 19-5*) should be disabled for any pins that are not being used for a wake-up interrupt. This should be done even if the main GPIO Interrupt Enable (*Figure 19-1*) is off.

Typical code for entering suspend is shown below:

```
...         ; All GPIO set to low-power state (no floating pins, and bit interrupts disabled unless using for wake-up)
...         ; Enable GPIO and/or wake-up timer interrupts if desired for wake-up
...         ; Select clock mode for wake-up (see Section 11.1)
mov a, 09h  ; Set suspend and run bits
iowr FFh    ; Write to Status and Control Register - Enter suspend, wait for GPIO/wake-up interrupt or USB activity
nop         ; This executes before any ISR
...         ; Remaining code for exiting suspend routine
```

11.1 Clocking Mode on Wake-up from Suspend

When exiting suspend on a wake-up event, the device can be configured to run in either Internal or External Clock mode. The mode is selected by the state of the External Oscillator Enable bit in the Clock Configuration Register (*Figure 9-2*). Using the

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved						P1[1:0] Mode1	
Read/Write	-	-	-	-	-	-	W	W
Reset	0	0	0	0	0	0	0	0

Figure 12-7. GPIO Port 1 Mode1 Register (Address 0x0D)

Bit [7:2]: Reserved

Bit [1:0]: P1[1:0] Mode 1

1 = Port Pin Mode 1 is logic HIGH

0 = Port Pin Mode 1 is logic LOW

Each pin can be independently configured as high-impedance inputs, inputs with internal pull-ups, open drain outputs, or traditional CMOS outputs with selectable drive strengths.

The driving state of each GPIO pin is determined by the value written to the pin's Data Register and by its associated Mode0 and Mode1 bits. *Table 12-1* lists the configuration states based on these bits. The GPIO ports default on reset to all Data and Mode Registers cleared, so the pins are all in a high-impedance state. The available GPIO output drive strength are:

- **Hi-Z Mode** (Mode1 = 0 and Mode0 = 0)

  Q1, Q2, and Q3 (*Figure 12-1*) are OFF. The GPIO pin is not driven internally. Performing a read from the Port Data Register return the actual logic value on the port pins.
- **Low Sink Mode** (Mode1 = 1, Mode0 = 0, and the pin's Data Register = 0)

  Q1 and Q3 are OFF. Q2 is ON. The GPIO pin is capable of sinking 2 mA of current.
- **Medium Sink Mode** (Mode1 = 0, Mode0 = 1, and the pin's Data Register = 0)

  Q1 and Q3 are OFF. Q2 is ON. The GPIO pin is capable of sinking 8 mA of current.
- **High Sink Mode** (Mode1 = 1, Mode0 = 1, and the pin's Data Register = 0)

  Q1 and Q3 are OFF. Q2 is ON. The GPIO pin is capable of sinking 50 mA of current.
- **High Drive Mode** (Mode1 = 0 or 1, Mode0 = 1, and the pin's Data Register = 1)

  Q1 and Q2 are OFF. Q3 is ON. The GPIO pin is capable of sourcing 2 mA of current.
- **Resistive Mode** (Mode1 = 1, Mode0 = 0, and the pin's Data Register = 1)

  Q2 and Q3 are OFF. Q1 is ON. The GPIO pin is pulled up with an internal 14-kΩ resistor.

Note that open drain mode can be achieved by fixing the Data and Mode1 Registers LOW, and switching the Mode0 register.

Input thresholds are CMOS, or TTL as shown in the table (See Section 23.0 for the input threshold voltage in TTL or CMOS modes). Both input modes include hysteresis to minimize noise sensitivity. In suspend mode, if a pin is used for a wake-up interrupt using an external R-C circuit, CMOS mode is preferred for lowest power.

Table 12-1. Ports 0 and 1 Output Control Truth Table

Data Register	Mode1	Mode0	Output Drive Strength	Input Threshold
0	0	0	Hi-Z	CMOS
1			Hi-Z	TTL
0	0	1	Medium (8 mA) Sink	CMOS
1			High Drive	CMOS
0	1	0	Low (2 mA) Sink	CMOS
1			Resistive	CMOS
0	1	1	High (50 mA) Sink	CMOS
1			High Drive	CMOS

12.1 Auxiliary Input Port

Port 2 serves as an auxiliary input port as shown in *Figure 12-8*. The Port 2 inputs all have TTL input thresholds.

13.1 USB Enumeration

A typical USB enumeration sequence is shown below. In this description, 'Firmware' refers to embedded firmware in the controller.

1. The host computer sends a SETUP packet followed by a DATA packet to USB address 0 requesting the Device descriptor.
2. Firmware decodes the request and retrieves its Device descriptor from the program memory tables.
3. The host computer performs a control read sequence and Firmware responds by sending the Device descriptor over the USB bus, via the on-chip FIFO.
4. After receiving the descriptor, the host sends a SETUP packet followed by a DATA packet to address 0 assigning a new USB address to the device.
5. Firmware stores the new address in its USB Device Address Register after the no-data control sequence completes.
6. The host sends a request for the Device descriptor using the new USB address.
7. Firmware decodes the request and retrieves the Device descriptor from program memory tables.
8. The host performs a control read sequence and Firmware responds by sending its Device descriptor over the USB bus.
9. The host generates control reads from the device to request the Configuration and Report descriptors.
10. Once the device receives a Set Configuration request, its functions may now be used.
11. Firmware should take appropriate action for Endpoint 1 transactions, which may occur from this point.

13.2 USB Port Status and Control

USB status and control is regulated by the USB Status and Control Register as shown in *Figure 13-1*.

Bit #	7	6	5	4	3	2	1	0
Bit Name	PS/2 Pull-up Enable	VREG Enable	USB Reset-PS/2 Activity Interrupt Mode	Reserved	USB Bus Activity	D+/D- Forcing Bit		
Read/Write	R/W	R/W	R/W	-	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 13-1. USB Status and Control Register (Address 0x1F)

Bit 7: PS/2 Pull-up Enable

This bit is used to enable the internal PS/2 pull-up resistors on the SDATA and SCLK pins. Normally the output high level on these pins is V_{CC}, but note that the output will be clamped to approximately 1 Volt above V_{REG} if the VREG Enable bit is set, or if the Device Address is enabled (bit 7 of the USB Device Address Register, *Figure 14-1*).

1 = Enable PS/2 pull-up resistors. The SDATA and SCLK pins are pulled up internally to V_{CC} with two resistors of approximately 5 kΩ (see Section 23.0 for the value of R_{PS2}).

0 = Disable PS/2 pull-up resistors.

Bit 6: VREG Enable

A 3.3V voltage regulator is integrated on chip to provide a voltage source for a 1.5-kΩ pull-up resistor connected to the D– pin as required by the USB Specification. Note that the VREG output has an internal series resistance of approximately 200Ω, the external pull-up resistor required is approximately 1.3-kΩ (see *Figure 16-1*).

1 = Enable the 3.3V output voltage on the VREG pin.

0 = Disable. The VREG pin can be configured as an input.

Bit 5: USB-PS/2 Interrupt Select

This bit allows the user to select whether an USB bus reset interrupt or a PS/2 activity interrupt will be generated when the interrupt conditions are detected.

1 = PS/2 interrupt mode. A PS/2 activity interrupt will occur if the SDATA pin is continuously LOW for 128 to 256 μs.

0 = USB interrupt mode (default state). In this mode, a USB bus reset interrupt will occur if the single ended zero (SE0, D– and D+ are LOW) exists for 128 to 256 μs.

See Section 19.0 for more details.

Bit 4: Reserved. Must be written as a '0'.

Bit [3:0]: Mode Bit [3:0]

The EP1 Mode Bits operate in the same manner as the EP0 Mode Bits (see Section 14.2).

14.4 USB Endpoint Counter Registers

There are two Endpoint Counter registers, with identical formats for both control and non-control endpoints. These registers contain byte count information for USB transactions, as well as bits for data packet status. The format of these registers is shown in *Figure 14-4*.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Data Toggle	Data Valid	Reserved		Byte Count			
Read/Write	R/W	R/W	-	-	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 14-4. Endpoint 0 and 1 Counter Registers (Addresses 0x11 and 0x13)

Bit 7: Data Toggle

This bit selects the DATA packet's toggle state. For IN transactions, firmware must set this bit to the select the transmitted Data Toggle. For OUT or SETUP transactions, the hardware sets this bit to the state of the received Data Toggle bit.

1 = DATA1

0 = DATA0

Bit 6: Data Valid

This bit is used for OUT and SETUP tokens only. This bit is cleared to '0' if CRC, bitstuff, or PID errors have occurred. This bit does not update for some endpoint mode settings. Refer to *Table 20-3* for more details.

1 = Data is valid.

0 = Data is invalid. If enabled, the endpoint interrupt will occur even if invalid data is received.

Bit [5:4]: Reserved

Bit [3:0]: Byte Count Bit [3:0]

Byte Count Bits indicate the number of data bytes in a transaction: For IN transactions, firmware loads the count with the number of bytes to be transmitted to the host from the endpoint FIFO. Valid values are 0 to 8 inclusive. For OUT or SETUP transactions, the count is updated by hardware to the number of data bytes received, plus 2 for the CRC bytes. Valid values are 2 to 10 inclusive.

For Endpoint 0 Count Register, whenever the count updates from a SETUP or OUT transaction, the count register locks and cannot be written by the CPU. Reading the register unlocks it. This prevents firmware from overwriting a status update on incoming SETUP or OUT transactions before firmware has a chance to read the data.

15.0 USB Regulator Output

The VREG pin provides a regulated output for connecting the pull-up resistor required for USB operation. For USB, a 1.5-kΩ resistor is connected between the D– pin and the VREG voltage, to indicate low-speed USB operation. Since the VREG output has an internal series resistance of approximately 200Ω, the external pull-up resistor required is R_{PU} (see Section 23.0).

The regulator output is placed in a high-impedance state at reset, and must be enabled by firmware by setting the VREG Enable bit in the USB Status and Control Register (*Figure 13-1*). This simplifies the design of a combination PS/2-USB device, since the USB pull-up resistor can be left in place during PS/2 operation without loading the PS/2 line. In this mode, the VREG pin can be used as an input and its state can be read at port P2.0. Refer to *Figure 12-8* for the Port 2 data register. This input has a TTL threshold.

In suspend mode, the regulator is automatically disabled. If VREG Enable bit is set (*Figure 13-1*), the VREG pin is pulled up to V_{CC} with an internal 6.2-kΩ resistor. This holds the proper V_{OH} state in suspend mode.

Note that enabling the device for USB (by setting the Device Address Enable bit, *Figure 14-1*) activates the internal regulator, even if the VREG Enable bit is cleared to 0. This insures proper USB signaling in the case where the VREG pin is used as an input, and an external regulator is provided for the USB pull-up resistor. This also limits the swing on the D– and D+ pins to about 1V above the internal regulator voltage, so the Device Address Enable bit normally should only be set for USB operating modes.

The regulator output is only designed to provide current for the USB pull-up resistor. In addition, the output voltage at the VREG pin is effectively disconnected when the device transmits USB from the internal SIE. This means that the VREG pin does not provide a stable voltage during transmits, although this does not affect USB signaling.

19.2 Interrupt Latency

Interrupt latency can be calculated from the following equation:

Interrupt Latency = (Number of clock cycles remaining in the current instruction) + (10 clock cycles for the CALL instruction) + (5 clock cycles for the JMP instruction)

For example, if a 5-clock-cycle instruction such as JC is being executed when an interrupt occurs, the first instruction of the Interrupt Service Routine will execute a minimum of 16 clocks (1+10+5) or a maximum of 20 clocks (5+10+5) after the interrupt is issued. With a 6 MHz external resonator, internal CPU clock speed is 12 MHz, so 20 clocks take 20/12 MHz = 1.67 μs.

19.3 Interrupt Sources

The following sections provide details on the different types of interrupt sources.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Wake-up Interrupt Enable	GPIO Interrupt Enable	Reserved			1.024-ms Interrupt Enable	128-μs Interrupt Enable	USB Bus Reset / PS/2 Activity Intr. Enable
Read/Write	R/W	R/W	-	-	-	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Figure 19-1. Global Interrupt Enable Register (Address 0x20)

Bit 7: Wake-up Interrupt Enable

The internal wake-up timer is normally used to wake the part from suspend mode, but it can also provide an interrupt when the part is awake. The wake-up timer is cleared whenever the Wake-up Interrupt Enable bit is written to a 0, and runs whenever that bit is written to a 1. When the interrupt is enabled, the wake-up timer provides periodic interrupts at multiples of period, as described in Section 11.2.

1 = Enable wake-up timer for periodic wake-up.

0 = Disable and power-off wake-up timer.

Bit 6: GPIO Interrupt Enable

Each GPIO pin can serve as an interrupt input. During a reset, GPIO interrupts are disabled by clearing all GPIO interrupt enable registers. Writing a '1' to a GPIO Interrupt Enable bit enables GPIO interrupts from the corresponding input pin. These registers are shown in *Figure 19-4* for Port 0 and *Figure 19-5* for Port 1. In addition to enabling the desired individual pins for interrupt, the main GPIO interrupt must be enabled, as explained in Section 19.0.

The polarity that triggers an interrupt is controlled independently for each GPIO pin by the GPIO Interrupt Polarity Registers. Setting a Polarity bit to '0' allows an interrupt on a falling GPIO edge, while setting a Polarity bit to '1' allows an interrupt on a rising GPIO edge. The Polarity Registers reset to 0 and are shown in *Figure 19-6* for Port 0 and *Figure 19-7* for Port 1.

All of the GPIO pins share a single interrupt vector, which means the firmware will need to read the GPIO ports with enabled interrupts to determine which pin or pins caused an interrupt.The GPIO interrupt structure is illustrated in *Figure 19-8*.

Note that if one port pin triggered an interrupt, no other port pins can cause a GPIO interrupt until that port pin has returned to its inactive (non-trigger) state or its corresponding port interrupt enable bit is cleared. The CY7C63221/31A does not assign interrupt priority to different port pins and the Port Interrupt Enable Registers are not affected by the interrupt acknowledge process.

1 = Enable

0 = Disable

Bit [5:3]: Reserved

Bit 2: 1.024-ms Interrupt Enable

The 1.024-ms interrupts are periodic timer interrupts from the free-running timer (based on the 6-MHz clock). The user should disable this interrupt before going into the suspend mode to avoid possible conflicts between servicing the timer interrupts (128-μs interrupt and 1.024-ms interrupt) first or the suspend request first when waking up.

1 = Enable. Periodic interrupts will be generated approximately every 1.024 ms.

0 = Disable.

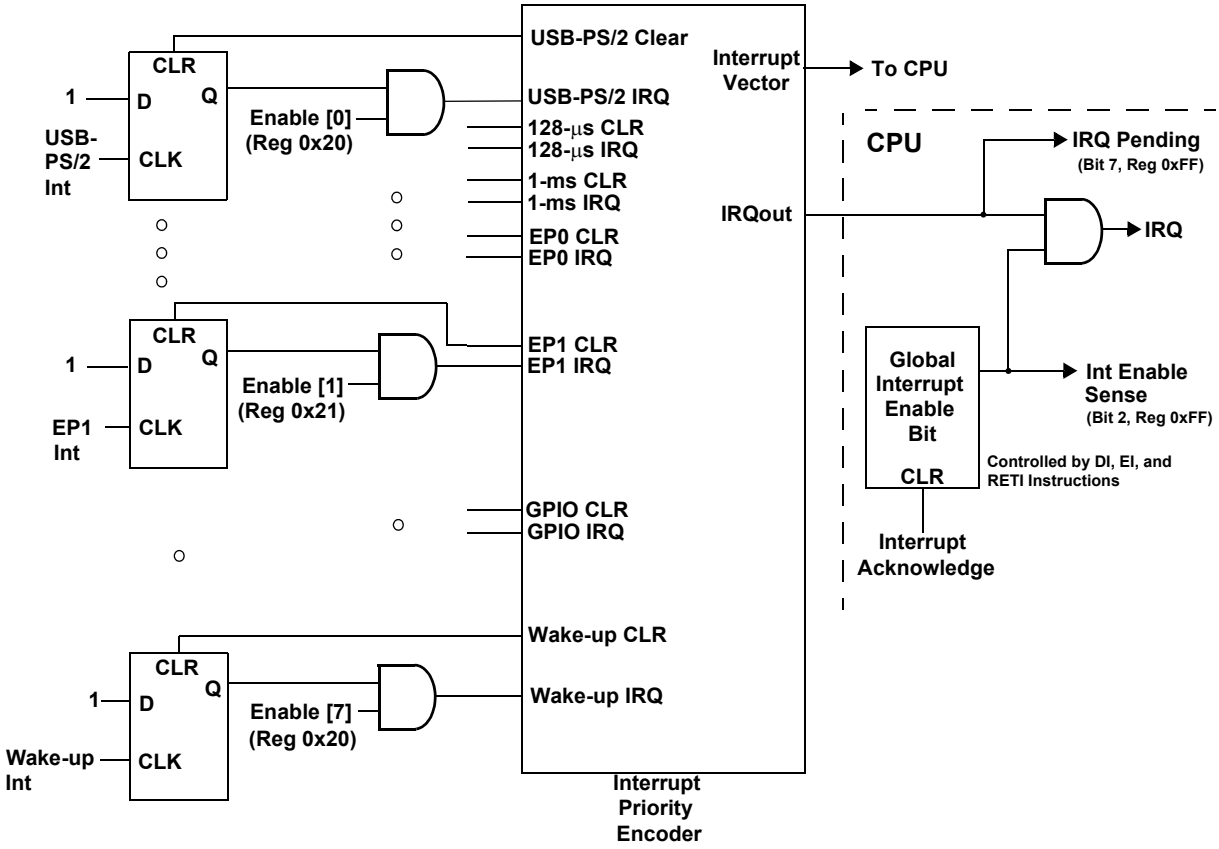


Figure 19-3. Interrupt Controller Logic Block Diagram

Bit #	7	6	5	4	3	2	1	0
Bit Name	P0 Interrupt Enable							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 19-4. Port 0 Interrupt Enable Register (Address 0x04)

Bit [7:0]: P0 [7:0] Interrupt Enable

1 = Enables GPIO interrupts from the corresponding input pin.

0 = Disables GPIO interrupts from the corresponding input pin.

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved						P1[1:0] Interrupt Enable	
Read/Write	-	-	-	-	-	-	W	W
Reset	0	0	0	0	0	0	0	0

Figure 19-5. Port 1 Interrupt Enable Register (Address 0x05)

Bit [7:0]: P1 [7:0] Interrupt Enable

1 = Enables GPIO interrupts from the corresponding input pin.

0 = Disables GPIO interrupts from the corresponding input pin.

The polarity that triggers an interrupt is controlled independently for each GPIO pin by the GPIO Interrupt Polarity Registers. *Figure 19-6* and *Figure 19-7* control the interrupt polarity of each GPIO pin.

Bit #	7	6	5	4	3	2	1	0
Bit Name	P0 Interrupt Polarity							
Read/Write	W	W	W	W	W	W	W	W
Reset	0	0	0	0	0	0	0	0

Figure 19-6. Port 0 Interrupt Polarity Register (Address 0x06)

Bit [7:0]: P0[7:0] Interrupt Polarity

1 = Rising GPIO edge

0 = Falling GPIO edge

Bit #	7	6	5	4	3	2	1	0
Bit Name	Reserved						P1[1:0] Interrupt Polarity	
Read/Write	-	-	-	-	-	-	W	W
Reset	0	0	0	0	0	0	0	0

Figure 19-7. Port 1 Interrupt Polarity Register (Address 0x07)

Bit [7:0]: P1[7:0] Interrupt Polarity

1 = Rising GPIO edge

0 = Falling GPIO edge

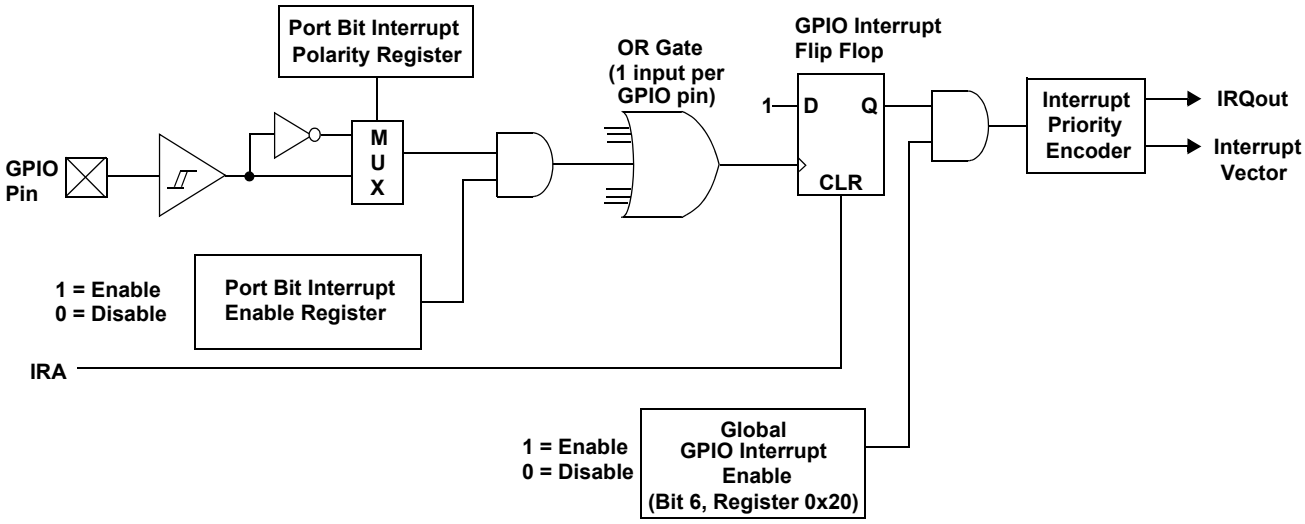


Figure 19-8. GPIO Interrupt Diagram

A 'TX 0 Byte' entry in the IN column means that the SIE will transmit a zero byte packet in response to any IN sent to the endpoint. Sending a 0 byte packet is to complete the status stage of a control transfer.

An 'Ignore' means that the device sends no handshake tokens.

An 'Accept' means that the SIE will respond with an ACK to a valid SETUP transaction.

Comments Column:

Some Mode Bits are automatically changed by the SIE in response to many USB transactions. For example, if the Mode Bits [3:0] are set to '1111' which is ACK IN-Status OUT mode as shown in *Table 20-1*, the SIE will change the endpoint Mode Bits [3:0] to NAK IN-Status OUT mode (1110) after ACKing a valid status stage OUT token. The firmware needs to update the mode for the SIE to respond appropriately. See *Table 20-1* for more details on what modes will be changed by the SIE.

Any SETUP packet to an enabled endpoint with mode set to accept SETUPs will be changed by the SIE to 0001 (NAKing). Any mode set to accept a SETUP will send an ACK handshake to a valid SETUP token.

A disabled endpoint will remain disabled until changed by firmware, and all endpoints reset to the Disabled mode (0000). Firmware normally enables the endpoint mode after a SetConfiguration request.

The control endpoint has three status bits for identifying the token type received (SETUP, IN, or OUT), but the endpoint must be placed in the correct mode to function as such. Non-Control endpoint should not be placed into modes that accept SETUPs.

21.0 Register Summary

	Address	Register Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Read/Write)/ Both(B)	Default/ Reset
GPIO CONFIGURATION PORTS 0, 1, AND 2	0x00	Port 0 Data	P0								BBBBBBBB	00000000
	0x01	Port 1 Data	Reserved						P1[1:0]		------BB	00000000
	0x02	Port 2 Data	Reserved		D+(SCLK) State	D- (SDATA) State	Reserved	P2.2(Int Clk Mode only)	P2.1 (Int Clk Mode only)	P2.0 Vreg Pin State	--RR-RRR	00000000
	0x0A	GPIO Port 0 Mode 0	P0[7:0] Mode0								WWWWWWWW	00000000
	0x0B	GPIO Port 0 Mode 1	P0[7:0] Mode1								WWWWWWWW	00000000
	0x0C	GPIO Port 1 Mode 0	Reserved						P1[1:0] Mode0		------WW	00000000
	0x0D	GPIO Port 1 Mode 1	Reserved						P1[1:0] Mode1		------WW	00000000
	0x04	Port 0 Interrupt Enable	P0[7:0] Interrupt Enable								WWWWWWWW	00000000
	0x05	Port 1 Interrupt Enable	Reserved						P1[1:0] Interrupt Enable		------WW	00000000
	0x06	Port 0 Interrupt Polarity	P0[7:0] Interrupt Polarity								WWWWWWWW	00000000
	0x07	Port 1 Interrupt Polarity	Reserved						P1[1:0] Interrupt Polarity		------WW	00000000
Clock Config.	0xF8	Clock Configuration	Ext. Clock Resume Delay	Wake-up Timer Adjust Bit [2:0]			Low Voltage Reset Disable	Precision USB Clocking Enable	Internal Clock Output Disable	External Oscillator Enable	BBBBBBBB	00000000
ENDPOINT 0, I AND 2 CONFIGURATION	0x10	USB Device Address	Device Address Enable	Device Address							BBBBBBBB	00000000
	0x12	EP0 Mode	SETUP Received	IN Received	OUT Received	ACKed Transaction	Mode Bit				BBBBBBBB	00000000
	0x14	EP1 Mode Register	STALL	Reserved		ACKed Transaction	Mode Bit				B--BBBBB	00000000
	0x11, 0x13	EP0 and 1Counter	Data 0/1 Toggle	Data Valid	Reserved		Byte Count				BB--BBBB	00000000
USB-SC	0x1F	USB Status and Control	PS/2 Pull-up Enable	VREG Enable	USB Reset-PS/2 Activity Interrupt Mode	Reserved	USB Bus Activity	D+/D- Forcing Bit			BBB-BBBB	00000000
INTERRUPT	0x20	Global Interrupt Enable	Wake-up Interrupt Enable	GPIO Interrupt Enable	Reserved			1.024 ms Interrupt Enable	128 µs Interrupt Enable	USB Bus Reset-PS/2 Activity Intr. Enable	BB---BBB	00000000
	0x21	Endpoint Interrupt Enable	Reserved						EP1 Interrupt Enable	EP0 Interrupt Enable	------BB	00000000
TIMER	0x24	Timer LSB	Timer Bit [7:0]								RRRRRRRR	00000000
	0x25	Timer (MSB)	Reserved				Timer Bit [11:8]				----RRRR	00000000
PROC SC.	0xFF	Process Status & Control	IRQ Pending	Watch Dog Reset	Bus Interrupt Event	LVR/BOR Reset	Suspend	Interrupt Enable Sense	Reserved	Run	RBBBBR-B	See Section 18.0

22.0 Absolute Maximum Ratings

Storage Temperature .....–65°C to +150°C
Ambient Temperature with Power Applied .....–0°C to +70°C
Supply voltage on V_{CC} relative to V_{SS} .....–0.5V to +7.0V
DC Input Voltage..... –0.5V to +V_{CC}+0.5V
DC Voltage Applied to Outputs in High Z State..... –0.5V to + V_{CC}+0.5V
Maximum Total Sink Output Current into Port 0 and 1 and Pins..... 70 mA
Maximum Total Source Output Current into Port 0 and 1 and Pins ..... 30 mA
Maximum On-chip Power Dissipation on any GPIO Pin .....50 mW
Power Dissipation .....300 mW
Static Discharge Voltage ..... >2000V
Latch-up Current ..... >200 mA

23.0 DC Characteristics

F_{OSC} = 6 MHz; Operating Temperature = 0 to 70°C

	Parameter	Min	Max	Units	Conditions
	General				
V_{CC1}	Operating Voltage	V_{LVR}	5.5	V	Note 4
V_{CC2}	Operating Voltage	4.35	5.25	V	Note 4
I_{CC1}	V_{CC} Operating Supply Current - Internal Oscillator Mode. Typical I_{CC1} = 16 mA[5]		20	mA	V_{CC} = 5.5V, no GPIO loading V_{CC} = 5.0V. T = Room Temperature
I_{CC2}	V_{CC} Operating Supply Current - External Oscillator Mode. Typical I_{CC2}= 13 mA[5]		17	mA	V_{CC} = 5.5V, no GPIO loading V_{CC} = 5.0V. T = Room Temperature
I_{SB1}	Standby Current - No Wake-up Osc		25	µA	Oscillator off, D– > 2.7V
I_{SB2}	Standby Current - With Wake-up Osc		75	µA	Oscillator off, D– > 2.7V
V_{PP}	Programming Voltage (disabled)	–0.4	0.4	V	
T_{RSNTR}	Resonator Start-up Interval		256	µs	V_{CC} = 5.0V, ceramic resonator
I_{IL}	Input Leakage Current		1	µA	Any I/O pin
I_{SNK}	Max I_{SS} GPIO Sink Current		70	mA	Cumulative across all ports[6]
I_{SRC}	Max I_{CC} GPIO Source Current		30	mA	Cumulative across all ports[6]
	Low-voltage and Power-on Reset				
V_{LVR}	Low-voltage Reset Trip Voltage	3.5	4.0	V	V_{CC} below V_{LVR} for >100 ns[7]
t_{VCCS}	V_{CC} Power-on Slew Time		100	ms	linear ramp: 0 to 4V[8]
	USB Interface				
V_{REG}	VREG Regulator Output Voltage	3.0	3.6	V	Load = R_{PU} +R_{PD}[9, 10]
C_{REG}	Capacitance on VREG Pin		300	pF	External cap not required
V_{OHU}	Static Output High, driven	2.8	3.6	V	R_{PD} to Gnd[4]
V_{OLU}	Static Output Low		0.3	V	With R_{PU} to VREG pin
V_{OHZ}	Static Output High, idle or suspend	2.7	3.6	V	R_{PD} connected D– to Gnd, R_{PU} connected D– to VREG pin[4]

Notes:
4. Full functionality is guaranteed in V_{CC1} range, except USB transmitter specifications and GPIO output currents are guaranteed for V_{CC2} range.
5. Bench measurements taken under nominal operating conditions. Spec cannot be guaranteed at final test.
6. Total current cumulative across all Port pins, limited to minimize Power and Ground-Drop noise effects.
7. LVR is automatically disabled during suspend mode.
8. LVR will re-occur whenever V_{CC} drops below V_{LVR}. In suspend or with LVR disabled, BOR occurs whenever V_{CC} drops below approximately 2.5V.
9. V_{RG} specified for regulator enabled, idle conditions (i.e., no USB traffic), with load resistors listed. During USB transmits from the internal SIE, the VREG output is not regulated, and should not be used as a general source of regulated voltage in that case. During receive of USB data, the VREG output drops when D– is LOW due to internal series resistance of approximately 200Ω at the VREG pin.
10. In suspend mode, V_{RG} is only valid if R_{PU} is connected from D– to VREG pin, and R_{PD} is connected from D– to ground

Document History Page

Document Title: CY7C63221/31A enCoRe™ Low-speed USB Peripheral Controller Document Number: 38-08028				
REV.	**ECN NO.**	**Issue Date**	**Orig. of Change**	**Description of Change**
**	116226	06/17/02	DSG	Change from Spec number: 38-01049 to 38-08028
*A	116976	10/23/02	BON	Reformat. Add note 5 to 24.0. Add DIE sale, Section 21.0. Change *Figure 9-1*
*B	270731	See ECN	BON	Replaced the 16-Lead (300-Mil) Molded SOIC S1 graphic with the correct 18-Lead (300-Mil) Molded SOIC S1 one in the Package Diagram Section and corrected part numbers for lead-free packages.