Converting an EZ-USB Application to EZ-USB FX

Introduction

The EZ-USB FX family of chips is essentially a follow-on to the EZ-USB family. They have similar architectures and, as such, a completed or in-progress project can be easily adapted from EZ-USB to EZ-USB FX with minimal effort. The purpose of this application note is to address the differences between the two families of chips and to outline the steps necessary to make the conversion. Included in the discussion are the EZ-Bulk chips, the AN2122 and AN2126. It is assumed that the reader is familiar with the EZ-USB family of chips. Additional features available on the EZ-USB FX are also briefly discussed for those applications which might benefit from utilizing the added capabilities. If the decision to transfer an application to EZ-USB FX is based on the need for additional FX features not available in EZ-USB, then a more detailed review of the data sheets should be performed to ensure that the chip selected can meet the design requirements.

Which Chip Should I Choose?

Within each family there are a variety of chips available to meet the diverse needs of a multitude of applications. Selection of a chip for an initial design is typically based on the cheapest solution that meets the minimum application requirements (i.e. memory size, I/O capabilities, data transfer rates, etc.). The variation of features is much simpler in the EZ-USB FX family and there is not a 1-to-1 correspondence with EZ-USB chips. Criteria such as max I/O rate, number of endpoints, max UART speed and presence of a data bus are no longer issues since all EZ-USB FX chips meet or exceed the capabilities of the EZ-USB family. So what does need to be considered are issues of memory size, isochronous support, number of I/O pins required, and the address bus. summarizes the available features. Also note that the data bus is available in the place of 8 port B I/O pins on the 52 and 80 pin packages.

Hardware Differences

Although features are similar throughout the EZ-USB and FX families, there are some minor hardware differences. These are discussed below.

Pin Differences

Pins used to implement features on the two chip families differ widely from chip to chip. Further differences in the pin functions may also require small hardware changes. Table 2 details the pins utilized to perform various functions on the two families of chips. It primarily illustrates the EZ-USB features and pins and shows the appropriate mapping to an EZ-USB FX pin. Since this application note is concerned with upgrading an existing EZ-USB application, additional features provided in the EZ-USB FX family are only summarized in the table. The following notes provide additional details concerning the pin differences between the two chip families.

Reset

Active high on EZ-USB, tie to V_{CC} through a 10KΩ resistor and to GND through a 1µf capacitor. Active low on EZ-USB FX, tie to GND through a 10KΩ resistor and to V_{CC} through a 1µf capacitor.

XIN/XOUT

Connect a 1MΩ resistor between these pins in parallel with the crystal on FX, not needed on EZ-USB.

CLK24 & CLKOUT

CLK24 on the EZ-USB is phase locked to the input clock. It runs at 24 MHz (or 12 MHz if CPU12MHZ is tied HIGH on the 48 pin packages). The clock output is disabled and driven high if the CLK24OE (CPUCS.1) bit is set to “0.”

CLKOUT on the EZ-USB FX is phase locked to the input clock. Its speed is set by external EEPROM bit CONFIG 0.2 (48MHZ) and defaults to 24 MHz if no EEPROM is present. Setting CONFIG 0.2 to “0” causes the 8051 to run at 24 MHz, setting it to “1” establishes 48MHz operation. The clock speed can not be changed after boot time. Additionally the clock output is inverted if CONFIG 0.1 (CLKINV) is set to 1. The clock output is disabled and tri-stated if the CPUCS bit 1 (CLK24OE) is set to “0.” Additional information on the CONFIG 0 register is provided in the EEPROM contents section below.

SCL/SDA

On EZ-USB the recommended pull up to V_{CC} is with a 2.2KΩ resistor. On EZ-USB FX the recommended pull up to V_{CC} is with a 1KΩ resistor.

Table 1. .FX Feature Set Summary

Part No.	Package	RAM	ISO	I/O	FIFO width	Data bus	Addr bus
CY7C64601-52	52 PQFP	4K	No	16	8 bit	(port B)	No
CY7C64603-52	52 PQFP	8K	No	16	8 bit	(port B)	No
CY7C64613-52	52 PQFP	8K	Yes	16	8 bit	(port B)	No
CY7C64603-80	80 PQFP	8K	No	32	16 bit	(port B)	No
CY7C64613-80	80 PQFP	8K	Yes	32	16 bit	(port B)	No
CY7C64603-128	128 PQFP	8K	No	40	16 bit	Yes	Yes
CY7C64613-128	128 PQFP	8K	Yes	40	16 bit	Yes	Yes

Table 2. EZ-USB To FX Pin Comparison

EZ-USB							EZ-USB FX		
2131Q	2121/22S 2131S	2125/26S 2135/36S	2122T	2126T	Name	Description	128 pin	80 pin	52 pin
21	10	10	11	11	AVCC	Analog Vcc	18	5	5
18	7	7	7	7	AGND	Analog Gnd	21	8	8
1	43	43	47	47	DISCON#	Disconnect	48	28	18
77	41	41	45	45	USBD	USB D- signal	65	38	24
79	42	42	46	46	USBD	USB D+ signal	66	39	25
7-12, 15, 16, 26-29, 34-37	N/A	N/A	N/A	N/A	A0-A5, A6, A7, A8-A11, A12-A15	8051 Address bus	105-108, 114-118, 120-122, 127, 128, 1, 2	N/A	N/A
48-51, 57-60	N/A	24-27, 28-31	N/A	26-29, 30-33	D0-D3, D4-D7	8051 Data bus	8-11, 13-16	N/A	N/A
80	N/A	N/A	N/A	N/A	PSEN#	Program Store Enable	33	N/A	N/A
61	32	32	35	35	BKPT	Breakpoint	41	N/A	N/A
25	13	13	14	14	RESET	Reset (see text)	69	42	28
24	N/A	N/A	N/A	N/A	EA	External Access	51	N/A	N/A
19	8	8	9	9	XIN	Crystal Input (see text)	19	6	6
20	9	9	10	10	XOUT	Crystal Output (see text)	20	7	7
68	N/A	N/A	N/A	34	PA0 / T0OUT	Port A	25	11	N/A
69	N/A	N/A	N/A	N/A	PA1 / T1OUT	Port A	26	12	N/A
70	N/A	N/A	N/A	N/A	PA2 / OE#	Port A	27	13	N/A
71	N/A	N/A	N/A	N/A	PA3 / CS#	Port A	28	14	N/A
73	39	39	N/A	N/A	PA4 / FWR#	Port A	29	15	10
74	40	40	N/A	N/A	PA5 / FRD#	Port A	30	16	11
75	N/A	N/A	44	44	PA6 / RXD0OUT	Port A	31	17	N/A
76	N/A	N/A	8	8	PA7 / RXD1OUT	Port A	32	18	N/A
44	24	N/A	26	N/A	PB0 / T2	Port B (see text)	79	47	29
45	25	N/A	27	N/A	PB1 / T2EX	Port B (see text)	80	48	30
46	26	N/A	28	N/A	PB2 / RXD1	Port B (see text)	81	49	31
47	27	N/A	29	N/A	PB3 / TXD1	Port B (see text)	82	50	32

Table 2. EZ-USB To FX Pin Comparison (Continued)

52	28	N/A	30	N/A	PB4 / INT4	Port B (see text)	83	51	33
53	29	N/A	31	N/A	PB5 / NT5#	Port B (see text)	84	52	34
54	30	N/A	32	N/A	PB6 / INT6	Port B (see text)	85	53	35
55	31	N/A	33	N/A	PB7 / T2OUT	Port B (see text)	86	54	36
30	14	14	16	16	PC0 / RXD0	Port C	110	68	43
31	15	15	17	17	PC1 / TXD0	Port C	111	69	44
32	16	16	18	18	PC2 / INT0#	Port C	112	70	45
33	17	17	19	19	PC3 / INT1#	Port C	113	71	46
38	18	18	20	20	PC4 / T0	Port C	123	73	48
39	19	19	21	21	PC5 / T1	Port C	124	74	49
40	20	20	22	22	PC6 / WR#	Port C	125	75	50
41	21	21	23	23	PC7 / RD#	Port C	126	76	51
4	2	2	2	2	CLK24	24 MHz Clock (see text)			
66	37	37	40	40	WAKEUP#	USB Wakeup	7	4	4
65	36	36	39	39	SCL	I2C Clock (see text)	5	2	2
64	35	35	38	38	SDA	I2C Data (see text)	6	3	3
2, 22, 42, 62	11, 22, 33, 44	11, 22, 33, 44	12, 24, 36, 48	12, 24, 36, 48	Vcc	Vcc	4, 17, 36, 55, 68, 75, 100, 109	1, 21, 41, 61	1, 14, 27, 40
3, 5, 6, 13, 14, 17, 23, 43, 56, 63, 72, 78	1, 3-6, 12, 23, 34, 38	1, 3-6, 12, 23, 34, 38	1, 3-6, 13, 25, 37, 41	1, 3-6, 13, 25, 37, 41	GND	Ground	3, 12, 23, 35, 40, 47, 52, 62, 67, 72, 78, 87, 99, 119	10, 20, 29, 40, 43, 60, 72, 80	13, 21, 26, 39, 47, 52
N/A	N/A	N/A	15	15	CPU12MHZ	Processor speed select			
67	N/A	N/A	N/A	N/A	Reserved	Leave unconnected	42, 43, 50	25-27, 44, 45, 46, 55, 56, 66, 67, 77-79	20
					Reserved	Connect to Ground	22, 37, 39, 49, 53, 54, 70, 71, 73, 74, 76, 77	9, 22, 24	9, 15, 17, 19, 22, 23
					various	Port D / Slave FIFO / GPIF	56-61, 63, 64	30-37	N/A
					various	Port E / Slave FIFO / GPIF	88-95	N/A	N/A
					various	Slave FIFO / GPIF	24, 44-46, 96-98, 101-104,	57-59, 62-65	37, 38, 41, 42
					XCLKSEL	External Clock Select	38	23	16

Port B/Data bus

The EZ-USB FX 128-pin parts have a dedicated data bus. On the 52 and 80 pin parts the multiplexed port B pins can be configured as the 8051 data bus if needed. To enable this set the Interface Select bits of the IFCONFIG register (IFCONFIG[1..0]) to "01." Of course other multiplexed functions of the port B pins will be unavailable, so a decision as to what recources the application needs may be required.

I²C Interface

On EZ-USB FX the I²C controller can operate at 400 kHz for fast data transfers with peripherals that support it. For compatibility, the controller powers up at 100 kHz. 400 kHz operation can be selected by setting EEPROM bit CONFIG 0.0 (400KHZ) to "1", if desired. To maintain compatibility with 100 kHz devices set this bit to "0." This bit is copied to the I2CCTL register bit 0, which is read/write to the 8051. Thus the I²C bus speed is initially set by the EEPROM bit, but may be subsequently changed by the 8051.

EEPROM Contents

An EEPROM can be used on both families to supply only the vendor and product information required to support enumeration, or to supply the entire firmware. Additional data requiredfor application-specific use can also be stored after the VID/PID and firmware in any remaining space. For providing IDs only, the EZ-USB requires EEPROM byte 0 to be 0xB0 whereas the FX requires 0xB4. To supply firmware, EZ-USB requires byte 0 to be 0xB2 whereas FX requires 0xB6. Bytes 1–6 of the EEPROM contain the 16 bit Vendor ID, Product ID and Device ID (low byte followed by high byte) for both families. No additional information is required for EZ-USB. For EZ-USB FX, byte 7 is CONFIG 0, which contains information to configure chip operation after power up. High bits 7-3 are "0." Bit 2 is 48MHZ, bit 1 is CLKINV, and bit 0 is 400KHZ, which have all been discussed previously. Byte 8 is reserved and should be set to 0x00. Refer to table 3.

CONFIG 0

b7	b6	b5	b4	b3	b2	b1	b0
0	0	0	0	0	48MHZ	CLKINV	400KHZ

In both families the vendor, product and device IDs stored in bytes 1-6 are not normally used if the EEPROM also contains the firmware. They can be included for operation with ReNum = 0, causing the EZ-USB core to handle device requests. Firmware starts at byte 7 in EZ-USB and byte 9 in FX. In either case, the supplied hex2bix.exe utility can be used to generate the required EEPROM image. By default it produces a 0xB2 initial byte. A 0xB6 EEPROM can be created with the addition of the command line option "-F 0xB6."

Table 3. EEPROM Contents

EEPROM address	EZ-USB
0	0xB0 or 0xB2
1	Vendor ID (VID) L
2	Vendor ID (VID) H
3	Product ID (VID) L
4	Product ID (VID) H
5	Device ID (VID) L
6	Device ID (VID) H
7	firmware...

EEPROM address	EZ-USB FX
0	0xB4 or 0xB6
1	Vendor ID (VID) L
2	Vendor ID (VID) H
3	Product ID (VID) L
4	Product ID (VID) H
5	Device ID (VID) L
6	Device ID (VID) H
7	CONFIG 0
8	"00"
9	firmware...

Software Differences

To a large extent the FX family is backward compatible with EZ-USB. Every effort was made to maintain the same register names and functionality. Differences are discussed below.

Registers and Buffers

The EZ-USB register/buffer names and addresses are retained in the EZ-USB FX, although some of the functionality has changed. Additional registers required to implement added FX features appear from 0x7800 to 0x7B3F. These additional registers are not aliased to 0x1800 to 0x1B3F. The remaining EZ-USB compatible registers are still aliased to 0x1B40-0x1FFF, allowing unused bulk buffers to be used as RAM.

To make use of the Cypress predefined register and bit names for EZ-USB FX, code should include the file "tng.h" following the standard EZ-USB header files "ezusb.h" and "ezregs.h". In Development Kit software versions 1.9 and later "tng.h" has been changed to "fx.h". All files need to be included:

#include "ezusb.h"

#include "ezregs.h"

#include "tng.h"

Specific changes in register functionality are as follows:

CPUCS

The FX uses 2 additional bits as referred to in the CLKOUT section above. Bit 3 (24/48) and bit 2 (CLKINV) are loaded from an EEPROM at power on. Additionally bit 1 name has been changed from CLK24OE to CLKOE.

b7	b6	b5	b4	b3	b2	b1	b0
RV3	RV2	RV1	RV0	24/48	CLKINV	CLKOE	8051RES
R	R	R	R	R	R	R/W	R
RV3	RV2	RV1	RV0	0	0	1	1

PORTnCFG

These registers have the same function as they did in the EZ-USB. Be aware that the I/O ports have additional multiplexed functions which can be selected using the PORTnCF2

and IFCONFIG registers. These additional registers are not required as long as the application will use the same functions as in EZ-USB.

UART230

The UART 230 register is not applicable to EZ-USB FX. Higher baud rates can be generated with standard FX capabilities due to the higher clock speed. For 230 Kbaud operation an external source of 3.68 MHz is required. Also see the Timer operation section below.

I2CMODE

This register was used to enable the I^2C stop interrupt only on the AN2122/2126 EZ-USB chips. It is available to all FX chips.

USBIRQ and USBIEN

Bit 5 of both registers (IBNIR and IBNIE) was used to provide IN Bulk NAK interrupts only on AN2122/2126 EZ-USB chips. It is available to all FX chips.

USBBAV

Added bit 4 (INT2SFR). The EZ-USB FX provides a fast method for clearing the IRQ2 interrupt flag by writing to special function register INT2CLR. To enable this feature set USBBAV bit 4 to "1." The standard EZ-USB method of clearing the interrupt is still available.

b7	b6	b5	b4	b3	b2	b1	b0
-	-	-	INT2SFR	BREAK	BPPULSE	BPEN	AVEN
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

As a sidenote, the same feature is also provided for the IRQ4 interrupt flag (used by the GPIF and slave FIFOs on the FX). It is enabled by setting the INT4SFR bit (INT4SETUP.2) to "1." Write any value to the INT4CLR special function register to clear the interrupt.

FASTXFR

The fast transfer is handled by the DMA on the EZ-USB FX. Bit 7 (FISO) and bit 6 (FBLK) have no function. The remaining bits can still be used to alter the read and write strobes.

Fast Transfer

This is one of the more common changes encountered when moving from EZ-USB to FX. To move data at high speed in the EZ-USB required use of the Fast Transfer mechanism. When Fast Transfer was enabled, by setting FISO or FBLK in the FASTXFR register to "1", a MOVX A,@DPTR and MOVX @DPTR,A would generate FWR and FRD strobes respectively. Other bits in the FASTXFR register would modify the characteristics of the FRD and FWR strobes. Additionally, changing the CPU stretch value would cause the strobes to be extended by altering the number of instruction cycles used by MOVX from 2 through 9. Bulk transfers also made use of the Auto Pointer feature to treat a block of RAM as a FIFO.

The 2Mbyte/sec Fast Transfer on the EZ-USB has been replaced by use of the 48 MByte/sec Direct Memory Access (DMA) Engine on the EZ-USB FX. To take advantage of the high speed EZ-USB Fast Transfer, routines were often written in assembly language. The set up for a DMA transfer can be written in assembly or a high level language. Since it is handled by the DMA engine, the speed of the actual data transfer will be the same regardless of the language used.

To conduct a DMA transfer on a block of memory the 8051 code supplies a source using the DMASRC register, a destination using the DMADEST register, and a length using the DMALEN register. DMALEN values 0x01 through 0xFF move 1 through 255 bytes respectively. The DMALEN value 0x00 specifies a 256 byte transfer. The transfer is initiated by writing any value to the DMAGO register. The firmware can check for completion by monitoring bit 7 (DONE) of the DMAGO register. If enabled, a vectored INT4 interrupt request is generated on a zero-to-one transition of the DONE bit. The source and destination registers have definitions that allow access as a word, as above, or as a byte. At the byte level, use the definitions DMASRCH, DMASRCL, DMADESTH, and DMADESTL. To cause the 8051 to write data to the data bus using the DMA and to enable the FRD and FWR strobes in a manner identical to the EZ-USB Fast Transfer, the source or destination DMAEXTFIFO is used. DMA registers are incremented by the DMA engine as the trasnfer progresses. Subsequent transfers should initialize the register again instead of assuming that they retain their previous values. The exception is for the FIFO related registers that always refer to the same location, such as DMAEXTFIFO and AINDATA. These types of registers are not altered by the DMA engine.

Since the DMA is not using the MOVX instruction to perform the transfer, stretch memory cycles have no effect on the length of the read and write strobes. DMA external strobes are extended using the DMABURST register. Setting bits 2-4 (DSTR[2..0]) has the same effect on the strobes as the stretch bits MD[2..0] of the CKCON register in EZ-USB. Otherwise DMA read and write strobes appear the same as the fast transfer strobes for the same settings of bits 0-5 of the FASTXFR register.

There are some important restrictions on DMA source/destination pairs that are described in the Technical Reference Manual, section 11.2.1. To summarize these, the 8051 can not be executing code in the same 2K internal memory block that is being used for a DMA transfer. In the 0x1800-0x1FFF block the 8051 can also not access bulk endpoint buffer memory or bulk endpoint byte count registers. A source block or a destination block may not cross the internal to external boundary during a transfer. The 8051 also may not execute code from external memory while an external source or destination is being used.

One other thing to point out, although it is the same on EZ-USB and EZ-USB FX, is that there is a program memory hole at location 0x7FE5. This location, used by the AUTODATA register, is unavailable for program code since it is being used to enable the Fast Transfer mechanism.

The following examples below show how to adapt Fast Transfers from the EZ-USB to DMA transfers.

Fast Transfer for moving 64 bytes of external FIFO data to bulk endpoint 4 IN buffer:

```
movdptr,#FASTXFR; Set up fast bulk transfer
mova,#01000000b; FBLK=1, RPOL,RM1..0=0
movx@dptr,a; Load FASTXFR register
Init:movdptr,#AUTOPTRH
mova,#HIGH(IN4BUF); High portion of IN4BUF
movx@dptr,a; Load AUTOPTRH
movdptr,#AUTOPTRL
mova,#LOW(IN4BUF); Low portion of IN4BUF
movx@dptr,a; Load AUTOPTRL
movdptr,#AUTODATA;point to the “FIFO” register
movr7,#8;loop counter - 8 times

loop:movx@dptr,a; write to IN buffer from ext
movx@dptr,a; again...
movx@dptr,a
movx@dptr,a
movx@dptr,a
movx@dptr,a
movx@dptr,a
djnzr7,loop; Do eight more times
```

Equivalent DMA transfer for moving 64 bytes of external FIFO data to bulk endpoint 4 IN buffer in C code:

```
DMASRC = DMAEXTFIFO;// Source - data bus
DMADEST = IN4BUF;// destination - endpt 4 IN
DMALEN = 64;// Transfer 64 bytes
DMAGO = 1;// Initiate transfers
while(!(DMAGO && bmDONE));// Wait for completion
```

Fast Transfer for moving 256 bytes from isochronous FIFO endpoint 8 to external data in assembly language:

```
movdptr,#FASTXFR; Set up fast bulk transfer
mova,#10000000b; FISO=1, RPOL,RM1..0=0
movx@dptr,a; Load FASTXFR register
Init:movdptr,#OUT8DATA;point to the “FIFO” register
movr7,#32;loop counter, 8 bytes/loop

loop:movxa,@dptr; write in FIFO from data bus
movxa,@dptr; again...
movxa,@dptr
movxa,@dptr
movxa,@dptr
movxa,@dptr
movxa,@dptr
djnzr7,loop; Do sixty four more times
```

Equivalent DMA transfer for moving 256 bytes from isochronous FIFO endpoint 8 to external data in C code:

```
DMASRC = OUT8DATA;// Source - endpt 8 OUT
DMADEST = DMAEXTFIFO;// Destination - data bus
DMALEN = 0;// Transfer 256 bytes
DMAGO = 1;// Initiate transfer
while(!(DMAGO && bmDONE));// Wait for completion
```

Timer Operation

The EZ-USB FX timers have the same functionality as EZ-USB timers, however they will increment twice as fast when using the internal clock, CLKOUT, as a source due to the higher 48 MHz clock speed. Routines which use the timers to gauge actual lengths of time will have to alter the values used in the timing routines. The same is true for baud rate generation. The same Mode 1 and 3 baud rates can be generated but a different timer reload value will be required. An alternative is to operate the FX at 24 MHz by setting EEPROM bit CONFIG 0.2 to “0.” This will allow timing and baud rate generation to remain compatible with EZ-USB, if required. Applications which used an external clock source to drive the clock circuit are not affected. When using an external source the EZ-USB was limited to a 3 MHz input. The EZ-USB FX can accept up to a 6 MHz input, allowing a different range of baud rates to be generated.

Write Recovery Delay

After the firmware executes a write instruction to a non-data buffer EX-USB FX register, there is a delay before the data is actually written into the register itself. This is only significant if performing back to back accesses of the same register. For example, a write instruction followed by a read-modify-write instruction (such as using the &= or |= operators) could potentially produce erroneous results. The problem is that the initial write may not have been completed before the next instruction reads the register. Consider the following code segment, assuming USBCS is initially = 0x00.

```
USBCS |=bmBIT3;
USBCS &=~bmBIT2;
```

Written as is, the first instruction sets bit 3 to “1” (USBCS = 0x08). The second instruction reads the USBCS register and uses the bit mask to ensure bit 2 is “0”. The result should still be USBCS=0x08 but will actually be USBCS=0x00 because the first write hasn’t completed setting bit 3 to “1” before the second instruction accesses the register again.

To prevent this the header “ezusb.h” includes the macro definition:

```
#define WRITEDELAY() {char writedelaydummy = 0;}.
```

This is the time equivalent of 3 NOPs. To achieve the proper results the above code needs to be re-written as

```
USBCS |=bmBIT3;
WRITEDELAY();
USBCS &=~bmBIT2;
```

Use of WRITEDELAY() only needs to be included when performing back to back accesses of the same FX register. Any other code can also be used to break up the back to back accesses, as long as it uses at least three instruction cycles.

Additional FX Features

The EZ-USB FX includes many features that are not available on the EZ-USB. It is worthwhile to discuss these for a couple of reasons. The decision to move from EZ-USB to FX may be due to the necessity for some of these features in an evolving design. An existing EZ-USB application may also be moving to FX as a stepping off point for a follow on product. Whatever the reason, it may be desired to modify an EZ-USB design to take advantage of some of these enhancements to improve device performance. The added FX features will be briefly summarized, however "upgrading" an EZ-USB application should be done by thoroughly reviewing the feature implementations in the FX Technical Reference Manual and data sheet.

DMA Engine

DMA has already been covered in some detail due to the necessity to explain the equivalent of an EZ-USB Fast Transfer. In addition to what has been discussed, the DMA Engine can be used to move data between various other locations. Sources and destinations include internal RAM, external RAM, bulk endpoint buffers, isochronous FIFOs, reclaimed isochronous buffers (RAM at 0x2000-0x2FFF is available as XDATA if ISODISAB = "1"), slave FIFOs, GPIF waveform descriptors, and an external FIFO (as described in the examples).

Slave FIFOs

The EZ-USB FX provides four 64-byte FIFOS that serve as general purpose buffers between external logic and the 8051. The four FIFOs are divided into an A and B group, each with an in and out direction. They can be configured as two 8 bit data paths or a single 16 bit data path. Select and Enable pins are available for each group as well as a single set of Read, Write and Clock pins. Flags and interrupts are available to the 8051 to signal when each FIFO is full, empty, or at a user programmed level. A separate set of flags is also available via pins to external logic.

General Programmable Interface (GPIF)

The GPIF is an 8 or 16 bit user programmable interface that allows glueless interconnection with many different types of peripherals. The GPIF uses up to six CTL lines to provide strobes or other output signals and up to six RDY lines as inputs. Six address bits are available (more can be implemented using other I/O pins) and the FIFO A and B data buses are used collectively as the GPIF data bus.

The GPIF is a state machine with user programmed CTL levels at each state. States can be non-decision points, waiting a programmed time before proceeding to the next state, or decision points, sampling inputs or flags to branch to another state. The GPIF.exe tool is provided to aid the developer in constructing the state machine.

The GPIF can be programmed to conduct a single transaction, typically a single read or write, or a series of transactions. The series can be terminated based on a predetermined transaction count, or based on a condition, such as a FIFO filling to a certain level.

Special-Function Registers

Additional special-function registers have been added to the EZ-USB FX to enhance the speed of performing certain operations. Some of these registers are bit addressable, for example to allow fast manipulation of a single port pin, while the remainder are byte addressable. The bit addressable SFRs are IOA, IOB, IOC, and IOD. The byte addressable registers are IOE, SOEA, SOEB, SOEC, SOED, SOEE, INT2CLR, and INT4CLR.

The IOA - IOE registers are the port A - E data registers and the SOEA - SOEE registers are the output enables. To enable use of these SFRs, the 8051 must set PORTSETUP.0 (SFRPORT) = "1." Note that the standard EZ-USB method of accessing the registers via OUTn, OEn, and PINSn still function normally for backward code compatibility.

INT2CLR and INT4CLR allow a fast means of clearing the associated interrupt request flags. INT2CLR was briefly described in the Register section. To enable use of theses SFRs set USBBAV.4 (INT2SFC) = "1" for INT2CLR and set INT4SETUP.2 (INT4SFC) = "1" for INT4CLR. Writing any value to these SFRs clears the associated interrupt request flag.

Conclusion

With the exception of altering pin configurations to match those of the new chip, most applications can be made to operate on EZ-USB FX with very few or no changes. The more common changes are very simple to implement. For example, consider the Fast Transfer which requires a very simple set-up using the DMA on the FX. Even an application which does not use any of the enhanced FX features will see improved performance due to the increased clock speed. If an effort is made to implement some of the powerful FX capabilities then greatly improved I/O throughput can be achieved and costs can be reduced due to elimination of external logic.