15. Registers

15.1 Introduction

This section describes the MoBL-USB FX2LP18 registers in the order they appear in the memory map, see Figure 5-3 on page 81. The registers are named according to the following conventions.

Most registers deal with endpoints*.* The general register format is **DDDnFFF**, where:

- **DDD** is endpoint direction, IN or OUT with respect to the USB host.

n is the endpoint number, where:

- 'ISO' indicates isochronous endpoints as a group.

FFF is the function, where:

- CS is a control and status register
- IRQ is an Interrupt Request bit
- IE is an Interrupt Enable bit
- BC, BCL, and BCH are byte count registers. BC is used for single byte counts, and BCH/BCL are used as the high and low bytes of 16-bit byte counts.
- DATA is a single-register access to a FIFO.
- BUF is the start address of a buffer.

15.1.1 Example Register Format

- EP1INBC is the Endpoint 1 IN byte count.

15.1.2 Other Conventions

USB–Indicates a global (not endpoint-specific) USB function.

ADDR–Is an address.

VAL–Means valid.

FRAME–Is a frame count.

PTR–Is an address pointer.

Register Name			Register Function				Address
b7	b6	b5	b4	b3	b2	b1	b0
bitname	**bitname**	**bitname**	**bitname**	**bitname**	**bitname**	**bitname**	**bitname**
R, W access	R, W access	R, W access	R, W access	R, W access	R, W access	R, W access	R, W access
Default val	Default val	Default val	Default val	Default val	Default val	Default val	Default val

The register table above illustrates the register description format used in this chapter.

- The top line shows the register name, functional description, and address in the memory.
- The second line shows the bit position in the register.
- The third line shows the name of each bit in the register.
- The fourth line shows CPU accessibility: R(ead), W(rite), or R/W.
- The fifth line shows the default value. These values apply after a hard reset.

15.2 Special Function Registers (SFR)

MoBL-USB FX2LP18 implements many control registers as SFRs (Special Function Registers). These SFRs are shown in Table 15-1. **bold** type indicates SFRs which are not in the standard 8051, but are included in the MoBL-USB FX2LP18.

Table 15-1. MoBL-USB FX2LP18 Special Function Registers (SFR)

x	8x	9x	Ax	Bx	Cx	Dx	Ex	Fx
0	**IOA**	**IOB**	**IOC**	**IOD**	**SCON1**	PSW	ACC	B
1	SP	**EXIF**	**INT2CLR**	**IOE**	**SBUF1**			
2	DPL0	**MPAGE**	**INT4CLR**	**OEA**				
3	DPH0			**OEB**				
4	**DPL1**			**OEC**				
5	**DPH1**			**OED**				
6	**DPS**			**OEE**				
7	PCON							
8	TCON	SCON0	**IE**	**IP**	**T2CON**	**EICON**	**EIE**	**EIP**
9	TMOD	SBUF0						
A	TL0	**AUTOPTRH1**	**EP2468STAT**	**EP01STAT**	**RCAP2L**			
B	TL1	**AUTOPTRL1**	**EP24FIFOFLGS**	**GPIFTRIG**	**RCAP2H**			
C	TH0		**EP68FIFOFLGS**		**TL2**			
D	TH1	**AUTOPTRH2**		**GPIFSGLDATH**	**TH2**			
E	**CKCON**	**AUTOPTRL2**		**GPIFSGLDATLX**				
F			**AUTOPTR-SETUP**	**GPIFSGLDATLNOX**				

All un-labeled SFRs are reserved.

15.3 About SFRs

Because the SFRs are directly addressable internal registers, firmware can access them quickly, without the overhead of loading the data pointer and performing a MOVX instruction. For example, the firmware reads the Port B pins using a single instruction, as shown below.

Single instruction to read port B:

```
mov    a,IOB
```

In the same manner, firmware writes the value 0x55 to Port C using only one MOV instruction, as shown below.

Single instruction to read port C:

```
mov    IOC,#55h
```

SFRs in Table 15-1 on page 238 rows 0 and 8 are bit-addressable; individual bits of the registers may be efficiently set, cleared, or toggled using special bit-addressing instructions (for example, **setb IOB.2** sets bit 2 of the IOB register).

IOA			**Port A (bit addressable)**				**SFR 0x80**
b7	b6	b5	b4	b3	b2	b1	b0
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

IOB			**Port B (bit addressable)**				**SFR 0x90**
b7	b6	b5	b4	b3	b2	b1	b0
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

AUTOPTRH1			**Autopointer 1 Address HIGH**				**SFR 0x9A**
b7	b6	b5	b4	b3	b2	b1	b0
A15	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

AUTOPTRL1			**Autopointer 1 Address LOW**				**SFR 0x9B**
b7	b6	b5	b4	b3	b2	b1	b0
A7	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

AUTOPTRH2			**Autopointer 2 Address HIGH**				**SFR 0x9D**
b7	b6	b5	b4	b3	b2	b1	b0
A15	**A14**	**A13**	**A12**	**A11**	**A10**	**A9**	**A8**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

AUTOPTRL2			**Autopointer 2 Address LOW**				**SFR 0x9E**
b7	b6	b5	b4	b3	b2	b1	b0
A7	**A6**	**A5**	**A4**	**A3**	**A2**	**A1**	**A0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	0	0	0

IOC			**Port C (bit addressable)**				**SFR 0xA0**
b7	b6	b5	b4	b3	b2	b1	b0
D7	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
x	x	x	x	x	x	x	x

INT2CLR			**Interrupt 2 Clear**				**SFR 0xA1**
b7	b6	b5	b4	b3	b2	b1	b0
x	**x**	**x**	**x**	**x**	**x**	**x**	**x**
W	W	W	W	W	W	W	W
x	x	x	x	x	x	x	x

INT4CLR			**Interrupt 4 Clear**				**SFR 0xA2**
b7	b6	b5	b4	b3	b2	b1	b0
x	**x**	**x**	**x**	**x**	**x**	**x**	**x**
W	W	W	W	W	W	W	W
x	x	x	x	x	x	x	x

Writing any value to INT2CLR or INT4CLR clears the INT2 or INT4 interrupt request bit for the INT2/INT4 interrupt currently being serviced.

Writing to one of these registers has the same effect as clearing the appropriate interrupt request bit in the MoBL-USB FX2LP18 external register space. For example, suppose the EP2 Empty Flag interrupt is asserted. The MoBL-USB FX2LP18 automatically sets bit 1 of the EP2FIFOIRQ register (in External Data memory space, at 0xE651), and asserts the INT4 interrupt request.

Using autovectoring, the MoBL-USB FX2LP18 automatically calls (vectors to) the EP2_FIFO_EMPTY 2 Interrupt Service Routine (ISR). The first task in the ISR is to clear the interrupt request bit, EP2FIFOIRQ.1. The firmware can do this either by accessing the EP2FIFOIRQ register (at 0xE651) and writing a ‘1’ to bit 1, or simply by writing any value to INT4CLR. The first method requires the use of the data pointer, which must be saved and restored along with the accumulator in an ISR. The second method is much faster and does not require saving the data pointer, so it is preferred.

EP2468STAT			Endpoint(s) 2,4,6,8 Status Flags				SFR 0xAA
b7	b6	b5	b4	b3	b2	b1	b0
EP8F	**EP8E**	**EP6F**	**EP6E**	**EP4F**	**EP4E**	**EP2F**	**EP2E**
R	R	R	R	R	R	R	R
0	1	0	1	1	0	1	0

The bits in EP2468STAT correspond to Endpoint Status bits in the MoBL-USB FX2LP18 register file, as follows:

Table 15-2. SFR and MoBL-USB FX2LP18 Register File Correspondences

Bit	EPSTAT SFR	MoBL-USB FX2LP18 Register.Bit	MoBL-USB FX2LP18 Register File Address
7	EP8 Full flag	EP8CS.3	E6A6
6	EP8 Empty flag	EP8CS.2	E6A6
5	EP6 Full flag	EP6CS.3	E6A5
4	EP6 Empty flag	EP6CS.2	E6A5
3	EP4 Full flag	EP4CS.3	E6A4
2	EP4 Empty flag	EP4CS.2	E6A4
1	EP2 Full flag	EP2CS.3	E6A3
0	EP2 Empty flag	EP2CS.2	E6A3

Note The Endpoint status bits represent the Packet Status.

EP24FIFOFLGS			Endpoint(s) 2, 4 Slave FIFO Status Flags				SFR 0xAB
b7	b6	b5	b4	b3	b2	b1	b0
0	**EP4PF**	**EP4EF**	**EP4FF**	**0**	**EP2PF**	**EP2EF**	**EP2FF**
R	R	R	R	R	R	R	R
0	0	1	0	0	0	1	0

EP68FIFOFLGS			Endpoint(s) 6, 8 Slave FIFO Status Flags				SFR 0xAC
b7	b6	b5	b4	b3	b2	b1	b0
0	**EP8PF**	**EP8EF**	**EP8FF**	**0**	**EP6PF**	**EP6EF**	**EP6FF**
R	R	R	R	R	R	R	R
0	1	1	0	0	1	1	0

AUTOPTRSETUP			Autopointer(s) 1 and 2 Setup				SFR 0xAF
b7	b6	b5	b4	b3	b2	b1	b0
0	**0**	**0**	**0**	**0**	**APTR2INC**	**APTR1INC**	**APTREN**
R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
0	0	0	0	0	1	1	0

MoBL-USB FX2LP18 provides two identical autopointers. They are similar to the internal ‘DPTR’ data pointers, but with an additional feature: each can automatically increment after every memory access. Using one or both of the autopointers, firmware can perform very fast block memory transfers.

The AUTOPTRSETUP register is configured as follows:

- Set APTRnINC=0 to freeze the address pointer, APTRnINC=1 to automatically increment it for every read or write of an XAUTODATn register. This bit defaults to 1, enabling the auto-increment feature.
- Set APTREN=1 to enable the autopointer for on-chip memory access.

The firmware then writes a 16-bit address to AUTOPTRHn/Ln. Then, for every read or write of an XAUTODATn register, the address pointer automatically increments (if APTRnINC=1).

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you’re on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm’s cloud-native docket research platform finds what other services can’t. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.

WHAT WILL YOU BUILD? | sales@docketalarm.com | 1-866-77-FASTCASE

fastcase®
Smarter legal research.