EZ-USB IO Ports

Introduction

The Cypress EZ-USB family contains a high-performance 8051 core that features 24-MHz operation, 4-clock cycles, 256 bytes of internal register RAM, expanded interrupts, two UARTS and two data pointers. While the EZ-USB CPU is in most respects a standard (but enhanced) 8051 processor, there are some architectural differences that must be accounted for when writing firmware.

The major architectural difference is that the IO ports are implemented differently than in a standard 8051. This note explains the differences, and concludes with a code example and waveforms for using the 'MOVX @R0' instruction.

The differences in IO ports are as follows:

1. The 8051 uses three SFRS (Special Function Registers) to control its three IO ports P0, P1, and P3. The EZ-USB family does not implement P0–P3. Instead, it implements three different IO ports PORTA, PORTB, and PORTC, and controls them using added registers in external ('MOVX') memory space.
2. The 8051 implements a "quasi-bidirectional" IO pin, while the EZ-USB chip explicitly sets IO pin direction using added register bits.
3. Both the 8051 and the EZ-USB chip multiplex port IO and alternate functions onto the IO pins. The 8051 switches between special function IO and port IO by setting the port bit HIGH, effectively turning the IO pin to input so that the special function IO can override the pin state. The EZ-USB chip selects between port IO and special functions (such as UART RxD or TxD) using added register bits.
4. During a 'MOVX @R0' or 'MOVX @R1' instruction the 8051 sends the contents of its P2 port to the upper address lines A[15..0], and the contents of R0 or R1 to the lower address lines A[7..0]. Since the EZ-USB chip does not have a P2, the high address byte is instead provided by an added SFR at location 92H.

EZ-USB IO Port Details

Figure 1 shows the basic structure of an EZ-USB IO pin, followed by the registers for PORTA. There are corresponding registers for PORTB and PORTC.

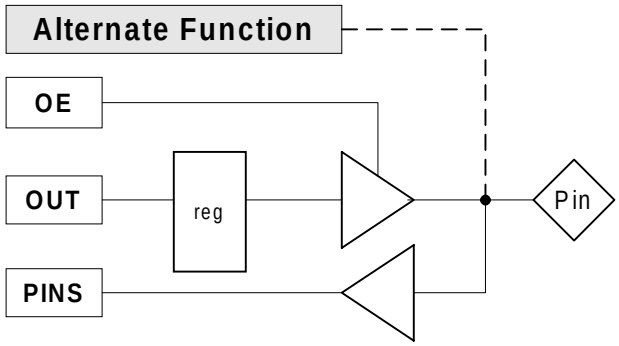

Figure 1. EZ-USB Input/Output Pin

PORTACFG			IO Port A Configuration				7F93
b7	b6	b5	b4	b3	b2	b1	b0
RxD1OUT	**RxD0OUT**	**FRD**	**FWR**	**CS**	**OE**	**T1OUT**	**T0OUT**

OEA			Port A Output Enable				7F9C
b7	b6	b5	b4	b3	b2	b1	b0
OEA7	**OEA6**	**OEA5**	**OEA4**	**OEA3**	**OEA2**	**OEA1**	**OEA0**

OUTA			Port A Outputs				7F96
b7	b6	b5	b4	b3	b2	b1	b0
OUTA7	**OUTA6**	**OUTA5**	**OUTA4**	**OUTA3**	**OUTA2**	**OUTA1**	**OUTA0**

PINSA			Port A Pins				7F99
b7	b6	b5	b4	b3	b2	b1	b0
PINA7	**PINA6**	**PINA5**	**PINA4**	**PINA3**	**PINA2**	**PINA1**	**PINA0**

The four registers that control PORTA have the following functions:

1. The PORTACFG register bits select either the PORTA IO pins (bit=0) or the indicated alternate function (bit=1). The default for these bits is 0.
2. The OEA register bits individually enable the eight PORTA output buffers (1=enabled).
3. The OUTA register bits are stored in latches that may or may not be connected to the IO pin, depending on the states of the OEA and PORTACFG pins (the direction of an alternate function overrides the OE setting).
4. The PINSA register bits indicate the pin states regardless of the pin configuration: port input, port output, or alternate function.

The 'MOVX @R0/R1' Instruction

The 8051 has two indirect address modes. One uses the data pointer (DPTR) as a 16-bit index register, and the other uses register R0 or R1 as an 8-bit index register. Data is transferred to and from outside memory using the 'MOVX' (move external) instruction in conjunction with the '@' symbol to indicate indirect addressing.

@ DPTR

8051 Instructions that use the DPTR operates identically in the 8051 and the EZ-USB family. For example, the instruction, 'MOVX A, @DPTR' does the following:

1. Outputs the 16-bit contents of DPTR to the address bus.
2. Floats the data bus for a read operation.
3. Generates a RD# strobe.
4. Reads the contents of the data bus into the accumulator.

@ R0/R1

A similar 8051 indirect address mode uses R0 or R1 to supply the *low address* byte. Where does the upper address byte come from? The high byte comes from different places in the 8051 and the EZ-USB family, as illustrated in *Figure 2*.

In the 8051, the upper address byte is provided from P2, the IO port 2 register. As was previously mentioned, the EZ-USB chips do not implement P2, but instead implement IO ports PORTA, PORTB, and PORTC using memory-mapped control registers. Since port 2 does not exist, the EZ-USB 8051 core implements a new Special Function Register (SFR) at location 0x92 whose only purpose is to supply the upper address byte for the 'MOVX @R0/R1' instructions.

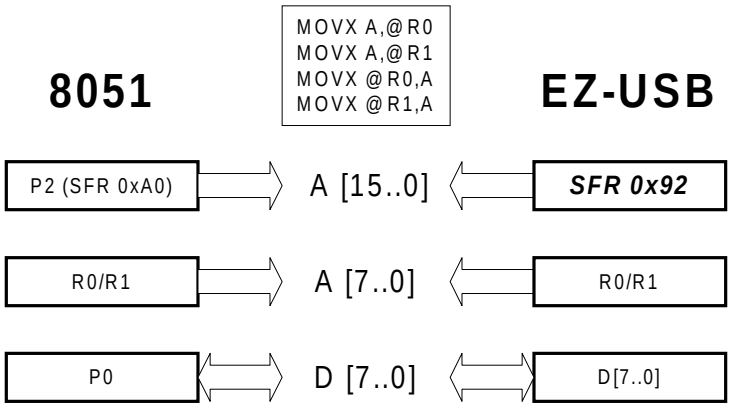

Figure 2. Different Registers Supply Upper Address A[15..8]

Listing

```
 1  ;------------------------------------------------------------
 2  ; movxtest.a51 18-Nov-98 LTH
 3  ;
 4  ; test the movx @R0 instruction in the AN2131
 5  ;------------------------------------------------------------
 6  $NOMOD51          ; disable predefined 8051 registers
 7  $nolist
 8  $INCLUDE (REG320.INC)
 9  $include (ezregs.inc)
10  $list
11  MOVXPAGE    equ     92H
12  ;
13  NAME        movxtest
14              ISEG AT 60H           ; stack
15  stack:      ds      20
16  ;
17              CSEG    AT      0     ; absolute Segment at Address 0
18              LJMP    start         ; Jump over the interrupt vectors
19  ; ------------------------------------------------
```

```
20              org     100h
21  ; ------------------------------------------------
22  start:      mov     SP,#STACK-1     ; set stack
23  ;
24              mov     dptr,#PORTCCFG
25              mov     a,#11000000b    ; read and write strobes
26              movx    @dptr,a
27              mov     dptr,#OEC       ; output enables, port C
28              movx    @dptr,a         ; same bits, b7-b6
29              mov     a,#80H          ; somewhere in external memory
30              mov     MOVXPAGE,a      ; MPAGE register at 92H
31  loop:       movx    @r0,a           ; write r0 to data bus
32              inc     a
33              sjmp    loop
34  ;
35              END
```

This code illustrates use of the ‘movx @r0’ instruction. The SFR at location 0x92 is named “MOVXPAGE” in line 11.

The following statements enable the RD# and WR# strobes:

- Lines 24–26: PORTCFG bits 7 and 6 are set to 1 to select the alternate functions (RD# and WR#). Only the WR# pulse is used in this example.
- Lines 27–28: The PORTC.7 and PORTC.6 output buffers are enabled.

The MOVXPAGE register is initialized to an off-chip upper address byte of 0x80 in line 30. Then a continuous loop writes the accumulator to external memory, increments the accumulator, and repeats. This example is meant only to show the timing of the data and WR# strobes, so the contents of R0 are unimportant. Normally R0 would also be initialized to reflect the lower external address byte.

Figure 3 shows the WR# line and bit 0 of the data bus for the example code. The D[0] line toggles for every write, as expected.

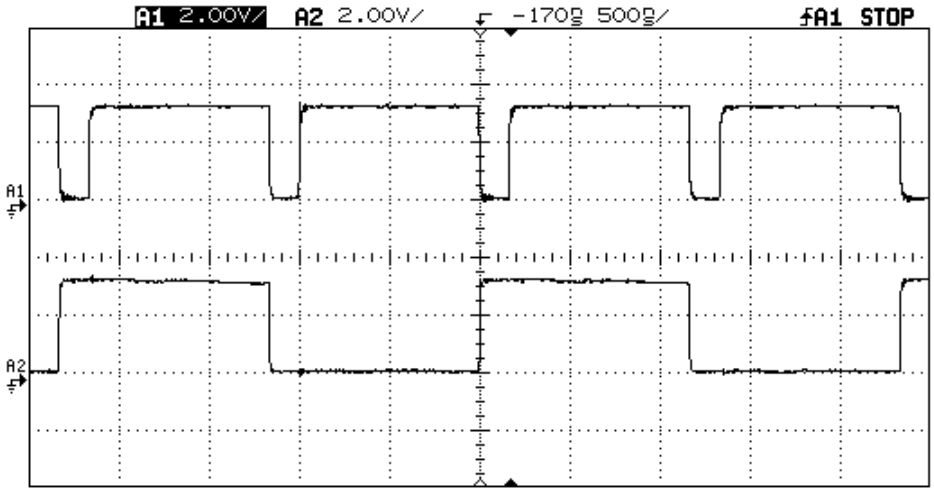

Figure 3. Waveforms for WR# and Data Bus