Expanding the ports of a Raspberry Pi using a MCP23S17.

The MCP23S17 is a 16-Bit I/O Expander with Serial (SPI) Interface. The topology of the design is shown below.

To control the ports of the part various registers need to be addressed. The full Raspberry code for doing this is included in this document. The register address map is shown below.

**TABLE 1-2: REGISTER ADDRESSES**

<table>
<thead>
<tr>
<th>Address</th>
<th>Address</th>
<th>Access to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOCON.BANK = 1</td>
<td>IOCON.BANK = 0</td>
<td>IODIRA</td>
</tr>
<tr>
<td>00h</td>
<td>00h</td>
<td>IODIRB</td>
</tr>
<tr>
<td>10h</td>
<td>01h</td>
<td>IOPOLA</td>
</tr>
<tr>
<td>11h</td>
<td>02h</td>
<td>IOPOLB</td>
</tr>
<tr>
<td>02h</td>
<td>03h</td>
<td>GPINTENA</td>
</tr>
<tr>
<td>12h</td>
<td>05h</td>
<td>GPINTENB</td>
</tr>
<tr>
<td>03h</td>
<td>06h</td>
<td>DEFVALA</td>
</tr>
<tr>
<td>13h</td>
<td>07h</td>
<td>DEFVALB</td>
</tr>
<tr>
<td>04h</td>
<td>08h</td>
<td>INTCONA</td>
</tr>
<tr>
<td>14h</td>
<td>09h</td>
<td>INTCONB</td>
</tr>
<tr>
<td>05h</td>
<td>0Ah</td>
<td>IOCON</td>
</tr>
<tr>
<td>15h</td>
<td>0Bh</td>
<td>IOCON</td>
</tr>
<tr>
<td>06h</td>
<td>0Ch</td>
<td>GPPUA</td>
</tr>
<tr>
<td>16h</td>
<td>0Dh</td>
<td>GPPUB</td>
</tr>
<tr>
<td>07h</td>
<td>0Eh</td>
<td>INTFA</td>
</tr>
<tr>
<td>17h</td>
<td>0Fh</td>
<td>INTFB</td>
</tr>
<tr>
<td>08h</td>
<td>10h</td>
<td>INTCAPA</td>
</tr>
<tr>
<td>18h</td>
<td>11h</td>
<td>INTCAPB</td>
</tr>
<tr>
<td>09h</td>
<td>12h</td>
<td>GPIOA</td>
</tr>
<tr>
<td>19h</td>
<td>13h</td>
<td>GPIOB</td>
</tr>
<tr>
<td>0Ah</td>
<td>14h</td>
<td>OLATA</td>
</tr>
<tr>
<td>1Ah</td>
<td>15h</td>
<td>OLATB</td>
</tr>
</tbody>
</table>
The basic registers are listed below:-

**IODIR - I/O DIRECTION REGISTER**

This controls the direction of the data I/O. When a bit is set, the corresponding pin becomes an input. When a bit is clear, the corresponding pin becomes an output.

For example to set all bits of portA to be outputs:-

```c
writeByte (IODIRA, 0x00) ;
```

and to set all bits of portB to be inputs:-

```c
writeByte (IODIRB, 0xFF) ;
```

Writing and reading data then becomes a simple matter of addressing the GPIO ports:-

```c
writeByte (GPIOA, <word>) ;
```

ie
```c
writeByte (GPIOA, 0xFF) ;
```

and reading the value of a port
```c
data = readByte (GPIOB) ;
```

**IPOL - INPUT POLARITY REGISTER**

This register allows the user to configure the polarity on the corresponding GPIO port bits. If a bit is set, the corresponding GPIO register bit will reflect the inverted value on the pin.

For example to invert bit 1 of port A

```c
writeByte (IPOLB, 0x01) ;
```

**GPPU - PULL-UP RESISTOR CONFIGURATION REGISTER**

The GPPU register controls the pull-up resistors for the port pins. If a bit is set and the corresponding pin is configured as an input, the corresponding port pin is internally pulled up with a 100 kΩ resistor.

For more advanced features and to use interrupts see the MCP23S17 datasheet.

References:-
The following is included for further reading.

https://projects.drogon.net/piface-mk2/


http://www.cmdrkeen.net/tag/spi/

http://www.linuxjournal.com/article/6908
Using the information in the above links the following code was put together to test out the basic features of the MCP23S17 part. The code is intended as a starting point only. Credit and recognition for figuring out how to drive this part goes to the authors of the above links.

To compile save to a file and compile with

```bash
> gcc <filename>.c -o <filename>
```

```
/*****************************************
* basic SPI demo for mcp23s17   *
*****************************************/
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <fcntl.h>
#include <sys/ioctl.h>
#include <linux/spi/spidev.h>
#include <sys/mman.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <unistd.h>
#include <stdint.h>
#define CMD_WRITE 0x40
#define CMD_READ 0x41

static char    *spiDevice = "/dev/spidev0.0" ;
static uint8_t  spiMode   = 0 ;
static uint8_t  spiBPW    = 8 ;
static uint32_t spiSpeed  = 5000000 ;
static uint16_t spiDelay  = 0;

// MCP23S17 Registers
#define IOCON 0x0A
#define IODIRA 0x00
#define IPOLA 0x02
#define GPINTEN 0x04
#define DEFVALA 0x06
#define INTCONA 0x08
#define GPPUA 0x0C
#define INTFA 0x0E
#define INTCAPA 0x10
#define GPIOA 0x12
#define OLATA 0x14
#define IODIRB 0x01
#define IPOLB 0x03
```
```c
#define GPINTENB 0x05
#define DEFVALB 0x07
#define INTCONB 0x09
#define GPPUB 0x0D
#define INTFB 0x0F
#define INTCAPB 0x11
#define GPIOB 0x13
#define OLATB 0x15

int spi_fd;

static uint8_t readByte (uint8_t reg)
{
    uint8_t tx [4] ;
    uint8_t rx [4] ;
    struct spi_ioc_transfer spi ;

    tx [0] = CMD_READ ;
    tx [1] = reg ;
    tx [2] = 0 ;

    spi.tx_buf = (unsigned long)tx ;
    spi.rx_buf = (unsigned long)rx ;
    spi.len = 3 ;
    spi.delay_usecs = spiDelay ;
    spi.speed_hz = spiSpeed ;
    spi.bits_per_word = spiBPW ;

    ioctl (spi_fd, SPI_IOC_MESSAGE(1), &spi) ;

    return rx [2] ;
}

static void writeByte (uint8_t reg, uint8_t data)
{
    uint8_t spiBufTx [3] ;
    uint8_t spiBufRx [3] ;
    struct spi_ioc_transfer spi ;

    spiBufTx [0] = CMD_WRITE ;
    spiBufTx [1] = reg ;
    spiBufTx [2] = data ;

    spi.tx_buf = (unsigned long)spiBufTx ;
    spi.rx_buf = (unsigned long)spiBufRx ;
    spi.len = 3 ;
    spi.delay_usecs = spiDelay ;
    spi.speed_hz = spiSpeed ;
    spi.bits_per_word = spiBPW ;

    ioctl (spi_fd, SPI_IOC_MESSAGE(1), &spi) ;
```

int spi_open(char* dev) {
    if((spi_fd = open(dev, O_RDWR)) < 0){
        printf("error opening %s\n", dev);
        return -1;
    }
    return 0;
}

int main(int argc, char* argv[]) {
    unsigned char data = 0xAF;
    if(argc <= 1){
        printf("too few args, try %s /dev/spidev0.0\n", argv[0]);
        return -1;
    }
    // open and configure SPI channel. (/dev/spidev0.0 for example)
    if(spi_open(argv[1]) < 0){
        printf("spi_open failed\n");
        return -1;
    }
    writeByte (IODIRA, 0x00) ;      // Port A -> Outputs
    writeByte (GPIOA, 0xFF) ;      // Port A -> Outputs
    writeByte (IPOLB, 0x01) ;      // invert lsb
    writeByte (GPPUB, 0xFF) ;      // enable pullups.
    writeByte (IODIRB, 0xFF) ;      // Port B -> Inputs
    data = readByte (GPIOB) ;
    printf("RECEIVED: %.2X\n",data);
    close(spi_fd);
    return 0;
}