I^2C COMMUNICATION BETWEEN PIC18FXX2 MICROCONTROLLER AND 24XX256 SERIAL EEPROM

Nina Bencheva, Yoana Ruseva

Department of Communication Systems and Technologies, "A. Kantchev" University of Rousse, 8 Studentska Street, 7017 Rousse, Bulgaria, phone: +359 887 746 257,
e-mail: nbencheva@ecs.ru.acad.bg

The paper presents an algorithm and a program to accomplish the necessary functions to utilize 24XX256 SEEPROMs in applications using the PIC18FXX2 microcontroller. The implementation is for a single master communication to one slave I^2C device. The hardware and the software connections between PIC18FXX2 and 24XX256 SEEPROM are presented. PIC18FXX2 support for I2C bus transfers and 24XX256 functionality are described.

The algorithm and the program given for a byte write have been fully tested. They can be easily modified for page write and every type of read operations.

Keywords: I^2C, PIC18FXX2 microcontroller, 24XX256 serial EEPROM

1. INTRODUCTION

The majority of the embedded applications use non-volatile memory that is available on the controllers or external EEPROMs devices. Parallel EEPROMs require too many I/O pins to be connected. Serial EEPROMs (SEEPROM) are easily interfaced to most microcontrollers using the industry standard I^2C. The Microchip Technology Inc. 24AA256/24LC256/24FC256 (24XX256) is a 32K x 8 (256 Kbit) SEEPROM. The 24XX256 SEEPROMs are suitable for embedded system code, firmware, lookup tables, and data storage applications.

The demo program for I^2C communication between PIC18FXX2 microcontroller and 24XX256 SEEPROM [2] has a number of disadvantages. The most serious of them is that in fact a write is not executed in SEEPROM at all, it is only imitated. In the flow diagram for acknowledgement polling [1] there is an incorrect operation block and the program can never go out from the loop including this block.

This paper considers an algorithm and a program to accomplish the necessary functions to utilize 24XX256 SEEPROMs in applications using the PIC18FXX2 microcontroller. The implementation is for a single master communication to one slave I^2C device.

2. INTERFACING THE 24XX256 SEEPROM TO THE PIC18FXX2 MICROCONTROLLER

The hardware connection between the PIC18FXX2 microcontroller and 24XX256 SEEPROM is shown in Fig. 1. The I^2C protocol utilizes a master/slave bi-directional communication bus. The two pins of PIC18FXX2, used for data transfer:

- Serial clock (SCL) - RC3/SCK/SCL
- Serial data (SDA) - RC4/SDI/SDA

are connected to the 24XX256's SCL and SDA pins correspondingly.

The chip address inputs A0, A1 and A2 of 24XX256 can be used to connect from 1 to 8 devices on the system bus. The levels on these inputs are compared with the corresponding bits in the slave address. The chip is selected if the comparison is true. In this application A0, A1 and A2 are hard-wired to logic '0'.

The WP (Write Protect) pin is connected to V_{SS} in order to enable the normal memory operation (read/write the entire memory address space from 0000H to 7FFFH).

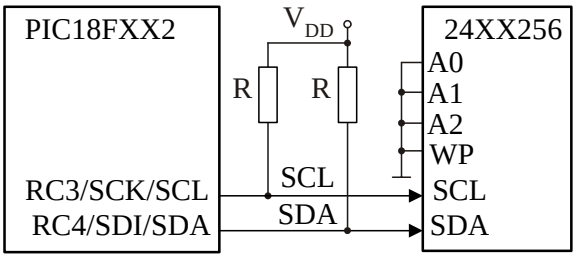

Fig. 1. Hardware connection between PIC18FXX2 and 24XX256 SEEPROM

3. PIC18FXX2 SUPPORT FOR I²C BUS TRANSFERS

The initialization required to use the PIC18FXX2 as the I^2C bus master is shown in Fig. 2.

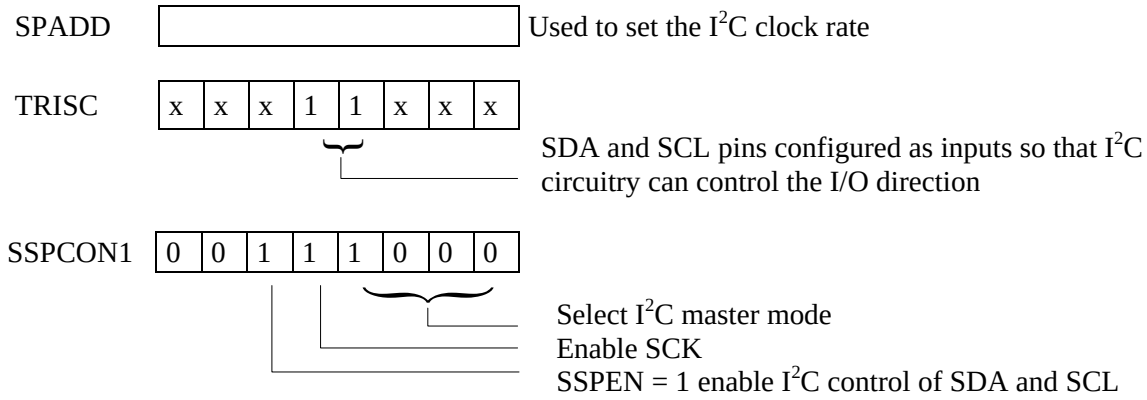

Fig. 2. Registers used in initialization

The relationship between the I^2C clock rate (F_{I2C}) and the frequency of the clock generator (F_{OSC}) is given by [3]: $\text{SSPADD} = \dfrac{F_{OSC}/4}{F_{I2C}} - 1$.

The SSPADD register has to be initialized by the value calculated.

The two bits of TRISC that program the general purpose I/O pins RC3 and RC4 to be inputs or outputs should be initialized as inputs. The I^2C control circuitry will override the input state when required.

The lower four bits of SSPCON1 register select the I^2C master mode. The SSPEN = 1 bit enables the serial port and configures the SDA and SCL pins as the serial port pins (if the corresponding TRISC bits are set as inputs). Bit 4 of SSPCON1 also has to be set to avoid stopping the I^2C clock if the setup time requirements are

not met.

SSPSTAT register is unused by I^2C master mode.

Once initialized, the registers in Fig. 3 control all the transfers. The message strings are built of seven building block commands [4]:

- SEN, send START condition. Set SEN bit to initiate START condition; hardware cleared.
- RSEN, send RESTART condition (release both lines and then send START condition). Set RESTART bit to initiate RESTART condition; hardware cleared.
- PEN, send STOP condition. Set STOP bit to initiate STOP condition; hardware cleared.
- Write to SSPBUF to send an address or data byte.
- RCEN, initiate the reading of a byte. Set RCEN to initiate the reception of a byte; hardware cleared.
- Read SSPBUF to read the received byte.
- ACKEN, set to respond to the received byte (by sending ACKDT = 0 to acknowledge or ACKDT = 1 not to acknowledge), hardware cleared.

Each of these commands is paced by the use of PIR1's SSPIF flag bit. Before the desired operation is initiated, SSPIF has to be cleared. Further activity is held up until SSPIF = 1, signaling the completion of the operation.

One more status bit, ACKSTAT, in the SSPCON2 register is used to query whether the transmitted byte is received correctly (ACKSTAT = 0) or the peripheral chip is signaling an unsuccessful transfer (ACKSTAT = 1).

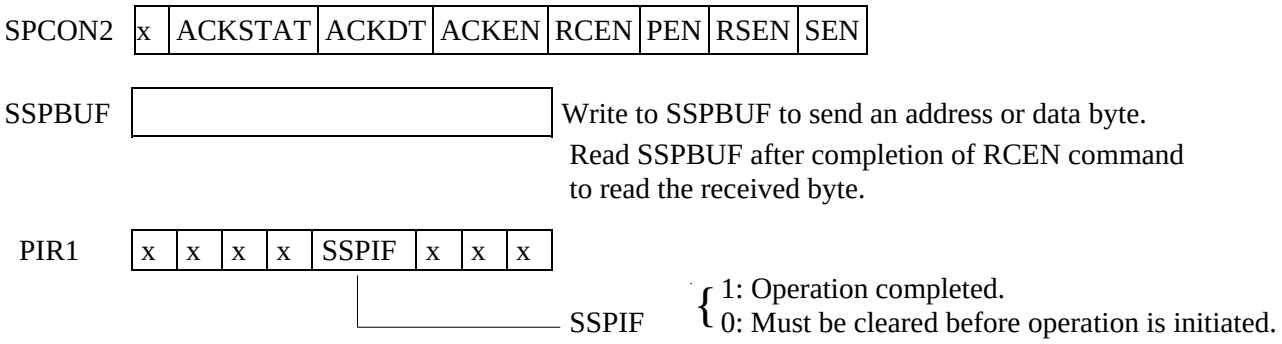

Fig. 3. Registers used in each message transfer

4. 24XX256 FUNCTIONAL DESCRIPTION

The 24XX256 supports I^2C bus and data transmission protocol. The bus must be controlled by a master device while the 24XX256 works as a slave. Both master and slave can operate as a transmitter or receiver, but the master device determines which mode is activated.

4.1 Device addressing

The I^2C protocol assigns a slave address for each unique device or device family. Microchip Technology and most non-volatile memories suppliers use a 4-bit control code 1010 for the SEEPROMs slave address. The next three bits of the control byte

are the Chip Select bits (A2, A1, A0). The last bit of the control byte ($R/\overline{W}$) defines the operation to be performed. Depending on the state of the $R/\overline{W}$ the 24XX256 will select a read or write operation. Upon receiving a '1010' code and appropriate device select bits, the slave device outputs an Acknowledge signal (ACK) on the SDA line.

The next two bytes received define the address of the first data byte.

4.2 Write operation

There are two types of write operation – byte write (Fig. 4) and page write.

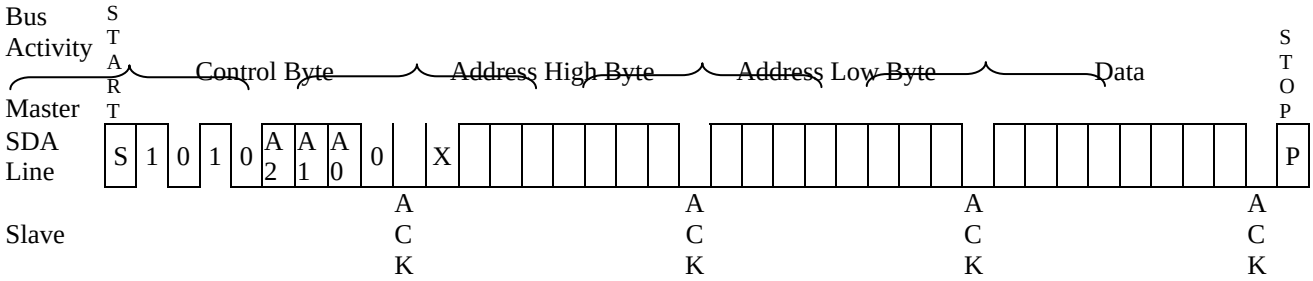

Fig. 4. Byte write

As it is shown in Fig. 4, after the control byte is acknowledged, the next byte transmitted by the master is the high-order byte of the word address, followed by the low-order byte of the word address. The address will be written into the address pointer of the 24XX256. After receiving the corresponding ACK signals from the 24XX256, the master device will transmit the data word to be written into the memory location addressed. The 24XX256 acknowledges again and the master generates a Stop condition. This initiates the internal write cycle and during this time, the 24XX256 will not generate ACK signals.

When using page mode, the control byte and address are sent for the first address only. After the data byte for the first address is sent, the data for the next consecutive address is clocked in. Upon receipt of each data byte, the six lower bits of the internal address pointer are automatically incremented by one. The device will still acknowledge between every byte of data. After the stop bit is sent, the 24XX256 will initiate the self timed write cycle. The cycle time for a page write is the same as that for a byte write.

4.3 Read operation

During read operations the SEEPROM is defined as a transmitter. Read operations are initiated in much the same way as write operations, except that the $R/\overline{W}$ bit of the control byte is set to '1'. There are three types of read operations: current address read, random read and sequential read.

The 24XX256 contains an address counter that maintains the address of the last word accessed, internally incremented by '1'. Therefore, if the previous read access was to address n (n is any legal address), the next current address read operation would access data from address n + 1.

The sequence of activities for a random read operation is shown in Fig. 5.

After a random read command, the internal address counter will point to the address location following the last read.

Sequential reads are initiated in the same way as a random read except that after the 24XX256 transmits the first data byte, the master issues an ACK instead of the Stop condition used in a random read. This ACK directs the 24XX256 to transmit the next sequentially addressed byte. To provide sequential reads, the 24XX256 internal address pointer is incremented by one at the completion of each operation. This address pointer allows the entire memory contents to be serially read during one operation.

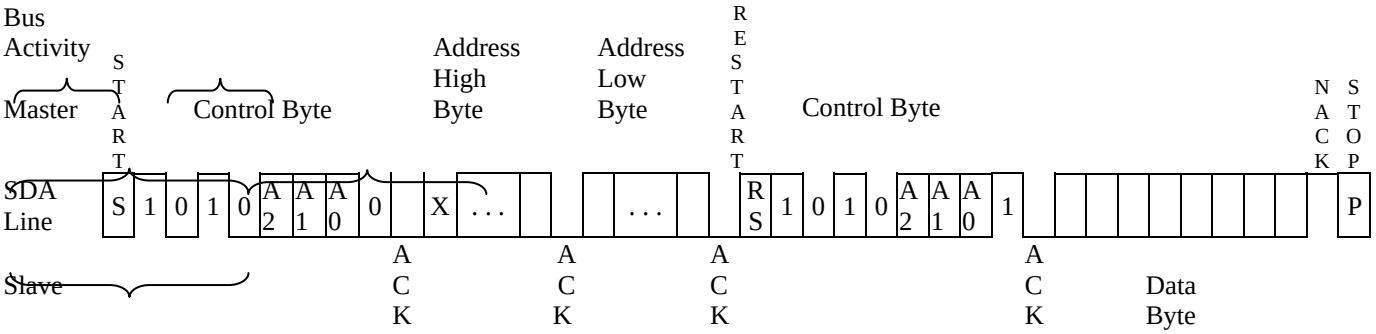

Fig. 5. Random read

5. SOFTWARE CONNECTION

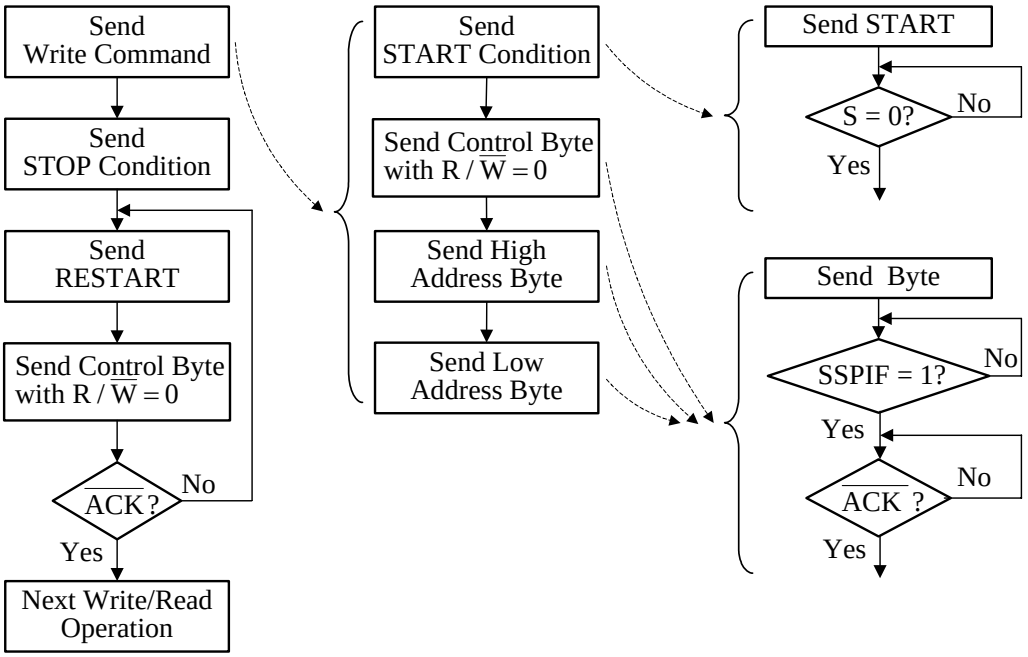

Fig. 6. Flow diagram for a byte write

The flow diagram for a byte write is shown in Fig. 6 and the corresponding assembly language program – in Fig. 7.

The master microcontroller controls the bus, generates the serial clock (SCL) and initiates the start and stop conditions. The SEEPROM is considered a slave device. It generates ACK when receiving data from the master. Normal operation begins with a start bit and ends with a stop bit. Following the start, commands begin with an 8 bit

'control' byte originating from the master.

```
    list    p=18f452
    include <p18f452.inc>
    org 0x00
    goto    start
    org 0x20
start
    bsf SSPSTAT,SMP         ; I2C slew rate control disabled
    bsf SSPCON1,SSPM3       ; I2C master mode in hardware
    bsf SSPCON1,SSPEN       ; enable SSP module
    movlw   0x09
    movwf   SSPADD          ; set I2C clock rate
    bsf TRISC,3             ; I2C SCL pin is input
    bsf PORTC,3             ; (will be controlled by SSP)
    bsf TRISC,4             ; I2C SDA pin is input
    bsf PORTC,4             ; (will be controlled by SSP)
    bsf SSPCON2,SEN         ; send start bit
    btfsc   SSPCON2,SEN     ; has SEN cleared yet?
    goto    $-2             ; no,loop back to test.
    bcf PIR1,SSPIF
    nop
    movlw   0xA0            ;move control byte to SSPBUF
    movwf   SSPBUF
    btfss   PIR1,SSPIF      ; has SSP completed sending control byte?
    goto    $-2             ; no, loop back to test
    btfsc   SSPCON2,ACKSTAT ; was ACK received from slave?
    goto    $-2             ; no, loop back to test
    bcf PIR1,SSPIF          ; clear interrupt flag
    movlw   0x00            ; move High address byte to SSPBUF
    movwf   SSPBUF
    btfss   PIR1,SSPIF      ; has SSP completed sending High address byte?
    goto    $-2             ; no, loop back to test
    btfsc   SSPCON2,ACKSTAT ; was ACK received from slave?
    goto    $-2             ; no, loop back to test
    bcf PIR1,SSPIF          ; clear interrupt flag
    movlw   0x00            ; move Low address byte to SSPBUF
    movwf   SSPBUF
    btfss   PIR1,SSPIF      ; has SSP completed sending Low address byte?
    goto    $-2             ; no, loop back to test
    btfsc   SSPCON2,ACKSTAT ; was ACK received from slave?
    goto    $-2             ; no, loop back to test
    bcf PIR1,SSPIF          ; clear interrupt flag
    movlw   0xC5            ; move Data byte to SSPBUF
    movwf   SSPBUF
    btfss   PIR1,SSPIF      ; has SSP completed sending Data byte?
    goto    $-2             ; no, loop back to test
    btfsc   SSPCON2,ACKSTAT ; was ACK received from slave?
    goto    $-2             ; no, loop back to test
    bsf SSPCON2,PEN         ; send stop bit
    btfsc   SSPCON2,PEN     ; has stop bit been sent?
    goto    $-2             ; no, loop back to test
    bcf PIR1,SSPIF          ; clear interrupt flag
stb bsf SSPCON2,RSEN        ; send restart bit
    btfsc   SSPCON2,RSEN    ; has RSEN cleared yet?
    goto    $-2             ; no, loop back to test.
    bcf PIR1,SSPIF          ; clear interrupt flag
    nop
    movlw   0xA0            ;move control byte to SSPBUF
    movwf   SSPBUF
    btfss   PIR1,SSPIF      ; has SSP completed sending control byte?
    goto    $-2             ; no, loop back to test
    btfsc   SSPCON2,ACKSTAT ; was ACK received from slave?
    goto    stb             ; no, loop back to test.
    end
```

Fig. 7. Assembly language program

When writing to a SEEPROM, there are two ways to handle the internally timed write cycle time of the device. The former method is to wait until the maximum cycle time is exceeded before attempting another command. The latter, more efficient method is ACK polling. Since the device will not acknowledge during a write cycle, this can be used to determine when the cycle is completed. ACK polling is executed by sending a RESTART condition and control byte to the SEEPROM after the write cycle has been initiated by a STOP bit. If the ACKSTAT bit is low, the write is completed, otherwise the sequence is repeated.

6. CONCLUSIONS

Developing systems, that implement the I^2C protocol for communicating with SEEPROM devices, requires that a certain factors be considered during the hardware and software development phase. It is recommended that the completion of the conditions START, RESTART and STOP be handled by polling the corresponding bit SEN, RSEN and PEN because they are hardware cleared. SSPIF = 1 is used only for signaling the completion of byte transfer.

To initiate a next operation after write cycle, a loop following the STOP condition and including sending the RESTART condition, sending the control byte and ACKSTAT polling has to be implemented.

The algorithm and the program developed have been fully tested. They can be easily modified for page write and every type of read operations

7. REFERENCES

[1] Microchip Technology Inc., *256K I2C™ CMOS Serial EEPROM*, DS21203M, 2004.

[2] Microchip Technology Inc., *PICDEM 2PLUS Software Tools, Example code for PICDEM-2*, 2002.

[3] Microchip Technology Inc., *PIC18FXX2 Data Sheet*, DS39564A, 2001.

[4] Peatman J. B., *Embedded Design with the PIC18F452 Microcontroller*, Pearson Education Inc, ISBN 0-13-046213-6, 2003.