Interfacing Microchip’s MCP41XXX and MCP42XXX Digital Potentiometers to a PICmicro® Microcontroller

Author: Craig L. King & Ezana Haile
Microchip Technology Inc.

OVERVIEW

The MCP41XXX and MCP42XXX family of digital potentiometers communicate using a standard 3-wire SPI™ compatible interface. This application note will discuss communications between these devices and a PIC16F876 microcontroller. The code supplied with this application note will include both absolute and relocatable assembly code, written for both hardware SPI and firmware SPI implementations.

COMMUNICATION

Instructions for the MCP41XXX and MCP42XXX devices consist of 16 clock cycles or two bytes. Figure 1 shows the format of these two bytes using a standard 3-wire SPI interface. The first byte is the command byte which must contain four bits to program the state of the digital potentiometer. The command byte determines the operation that is performed as well as identifies which potentiometer will execute the command (MCP42XXX devices contain two potentiometers). The second byte is the data byte. The MCP41XXX and MCP42XXX potentiometers are 8-bit or 256 tap potentiometers. All 8 bits in the data byte are wiper data bits. Depending on the state of P0 and P1 in the command byte, the data byte sets the wiper’s position or positions.

The four command bits to consider are bits 4:5 (C0:C1) and bits 0:1 (P0:P1). C0 and C1 determine which command is being issued. For the MCP41XXX and MCP42XXX devices, there are three possible commands:
- Write new data to potentiometer(s)
- Shutdown potentiometer(s)
- NOP (No Operation).

The MCP42XXX devices contains two potentiometers, P0 and P1. P0 uses pins 5, 6 and 7. P1 uses pins 8, 9 and 10. Using these two bits, the user can select either, both or neither potentiometer. A ‘1’ for either P1 or P0 will cause the data to be written to the respective data register and a ‘0’ for P1 or P0 will cause no change. The MCP41XXX devices contain only one potentiometer. For these devices, P1 is a don’t care.

![Figure 1: Instruction sequence and command byte summary for MCP41XXX AND MCP42XXX DIGITAL potentiometers](image)

SPITM is a trademark of Motorola Inc.
IMPLEMENTATION

Appendix A has the absolute assembly code using hardware SPI implementation. The Synchronous Serial Port (SSP) is first initialized to clock data out on the falling edge, drive the clock high when idle and clock with a frequency of Fosc/4. This is done by setting the SSPCON register to 0x30. Communication to the potentiometer is initiated by pulling the chip select line low. A command byte of 13h is then loaded into the SSPBUF of the PIC16F876. This command byte value will instruct a write command to potentiometer P0 and P1. The BF bit in the SSPSTAT register is then monitored. When this bit is high, the 8-bit transfer is complete. Once this transfer is complete, the data byte is then loaded into the SSPBUF register with the resistor value to be programmed into the digital potentiometer. This example shows the potentiometer being set to code 8Ch (140d). Again, the BF bit of the SSPSTAT register is monitored. Once this byte is transferred, chip select is raised and the instruction is complete. At the rising edge of chip select, the MCP41XXX or MCP42XXX will change the wiper position.

Appendix B shows absolute assembly code using firmware SPI implementation. The same pins are used to generate the clock and data signals as the hardware SPI example. Port initialization occurs, setting the CS, CLK and SDO port pins to outputs. The TRANSMIT routine handles the firmware SPI implementation, excluding the toggling of CS. Communication is again initiated by pulling chip select low. The Working Resister (W) is pre-loaded with the command byte and a call to the TRANSMIT routine is made. This routine generates 8 clock cycles and also sends out the W register on the data line. Upon completion of this routine, the W register is then pre-loaded with the data byte, or the resistor value to be programmed into the digital potentiometer. A second call to the TRANSMIT routine follows and communication is completed by raising the chip select line. At the rising edge of chip select, the MCP41XXX or MCP42XXX device will execute the command and the wiper position will change.

Appendix C is a relocatable version of the hardware SPI code in Appendix A. Appendix D is the relocatable version of the software SPI code in Appendix B. The linker script file (16F876.1kr) is shown in Appendix E. This file controls where the relocatable segments are placed in the PIC16F876 program memory and defines the processors available RAM space for the linker. Please consult the MPASM™ User’s Guide for more details on how to write and assemble relocatable code.

SCHEMATIC

The code for this application note was developed on the MXDEV™ Analog Evaluation Driver Board along with the MCP42XXX evaluation board. An equivalent circuit of the board used in this application note is shown in Appendix F. A full schematic of the MXDEV driver board and the MCP42XXX evaluation board can be found in the MXDEV Driver Board Users Manual (DS51221) and the MCP42XXX Evaluation Board Users Guide (DS51229). The SPI communication lines CLK and DOUT use pins RC3 and RC5, respectively. The chip select signal is generated using port pin RA4. The PIC16F876 uses crystal oscillator, X1. An MCP130 is used as the power on reset device. An MCP42010 is used as the digital potentiometer. A volt meter, V1, is used to measure the voltage at the output of the wiper which was used to determine the correct operation of the code.

CONCLUSION

The example code given in this application note shows how to interface either an MCP41XXX or an MCP42XXX device to a PICmicro. Multiple styles of implementation were given to allow the developer to use this code in almost any end-user application.

MEMORY USAGE

In the Digital Potentiometer, the following memory was used:

<table>
<thead>
<tr>
<th>Memory Type</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Program Memory</td>
<td>24 bytes</td>
</tr>
<tr>
<td>Data Memory</td>
<td>0 bytes</td>
</tr>
<tr>
<td>EEPROM Memory</td>
<td>0 bytes</td>
</tr>
</tbody>
</table>

REFERENCES


KEYWORDS

1. Potentiometer
2. Digital Potentiometers
3. MCP4XXX
4. MCP41XXX
5. MCP42XXX
6. Interfacing PICmicro microcontroller
7. SPI
8. PIC16F876
APPENDIX A: ASSEMBLY CODE USING HARDWARE SPI IMPLEMENTATION

;*******************************************************************************************
; * Interfacing Microchip’s MCP42xxx digital potentiometer to the PICmicro MCU
; * - THIS PROGRAM IS ABSOLUTE ASSEMBLY USING THE HARDWARE
; * SPI MODULE TO PROGRAM THE DIGITAL POTENTIOMETER
; *
;*******************************************************************************************

Filename: POTSPI1.ASM
Date: 11.07.2000
File Version: 1.00
Assembler: MPASM VERSION 2.50
PROGRAMER: PRO MATE DEVICE PROGRAMMER, VERSION 5.20.00
File Required: PIC16F876.inc
Author: Ezana Haile
Company: Microchip Technology Incorporated

This code demonstrates how Microchip’s MCP42xxx Digital Potentiometer (Pot) is interfaced to the PICmicro MCU (PIC16F876). The Potentiometer requires a serial communication to program the command byte and the data byte. This MCU has a built-in Serial Peripheral Interface (SPI) which can be used to program the pot effectively. The following program illustrates how to interface the digital pot using the MCU’s SPI.

To change the command byte or the Pot wiper position the user must change the COMMAND and R_VALUE variables properly and reprogram the MCU.

*******************************************************************************

#include <p16f876.inc>
ERRORLEVEL -302
__CONFIG _BODEN_OFF & _PWRTE_OFF & _CP_OFF & _WDT_OFF & _XT_OSC

SOFTWARE LICENSE AGREEMENT

The software supplied herewith by Microchip Technology Incorporated (the "Company") for its PICmicro® Microcontroller is intended and supplied to you, the Company’s customer, for use solely and exclusively on Microchip PICmicro Microcontroller products.

The software is owned by the Company and/or its supplier, and is protected under applicable copyright laws. All rights are reserved. Any use in violation of the foregoing restrictions may subject the user to criminal sanctions under applicable laws, as well as to civil liability for the breach of the terms and conditions of this license.

THIS SOFTWARE IS PROVIDED IN AN "AS IS" CONDITION. NO WARRANTIES, WHETHER EXPRESS, IMPLIED OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE APPLY TO THIS SOFTWARE. THE COMPANY SHALL NOT, IN ANY CIRCUMSTANCES, BE LIABLE FOR SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, FOR ANY REASON WHATSOEVER.
;*************************************************************
;********************  EQUATES  ************************************************************
;*************************************************************
CS EQU H'00' ;CHIP SELECT
COMMAND EQU H'13' ;VARIABLE FOR THE COMMAND BYTE
R_VALUE EQU D'140' ;VARIABLE FOR THE RESISTANCE VALUE

;*************************************************************
;********************  PROGRAM ORIGIN  ****************************************************
;*************************************************************

ORG 0X00

;-------------------------------------------------------------------------------
;-------------------  PORTB AND SPI SETTING  ----------------------------------
;-------------------------------------------------------------------------------

BSF STATUS, RP0 ;SPECIFY BANK 1
MOVlw H'00'
MOVWF TRISA ;SET PORTA AS AN OUTPUT
MOVWF TRISC ;SET PORTB AS AN OUTPUT
BCF STATUS, RP0 ;SPECIFY BANK 0
CLRF PCLATH ;ENSURE PCLATH BIT 3 IS CLEARED
CLRF INTCON ;ENSURE ALL INTERRUPTS ARE DISABLED
MOVlw 0x30 ;
MOVWF SSPCON ;SET SYNC SERIAL PORT CONTROL REGISTER

;-------------------------------------------------------------------------------
;-------------------  PROGRAM ROUTINE  ------------------------------------------
;-------------------------------------------------------------------------------

BCF PORTA, CS ;SELECT THE POT
MOVlw COMMAND ;LOAD THE COMMAND BYTE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE COMMAND BYTE
MOVlw R_VALUE ;LOAD THE RESISTANCE VALUE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE RESISTANCE VALUE
BSF PORTA, CS ;UNSELECT THE POT
GOTO FINISH ;FINISH

;-------------------------------------------------------------------------------
;-------------------  TRANSMISSION SUBROUTINE  -------------------------------------
;-------------------------------------------------------------------------------

TRANSMIT BCF STATUS, RP0 ;SPECIFY BANK 0
MOVWF SSPBUF ;PLACE DATA IN BUFFER TO SEND

LOOP BTFSS SSPSTAT, BF ;CHECK IF TRANSMISSION IS COMPLETE
GOTO LOOP ;
BCF STATUS, RP0 ;SPECIFY BANK 0
RETURN ;RETURN FROM SUBROUTINE
FINISH    GOTO    FINISH
END

;****************************** END OF PROGRAM ******************************************
APPENDIX B: ASSEMBLY CODE USING FIRMWARE SPI IMPLEMENTATION

;***************************************************************************
; Interfacing Microchip's MCP42xxx digital potentiometer to the PICmicro MCU
; - THIS PROGRAM IS ABSOLUTE ASSEMBLY USING PORTB TO ACT AS SOFTWARE
; SPI AND PROGRAM THE DIGITAL POTENTIOMETER
;***************************************************************************

;***************************************************************************
; Filename: POT_PRG.ASM
; Date: 11.02.2000
; File Version: 1.00
; Assembler: MPASM VERSION 2.50
; PROGRAMER: PRO MATE DEVICE PROGRAMMER, VERSION 5.20.00
; File Required: PIC16F876.inc
; Author: Ezana Haile
; Company: Microchip Technology Incorporated
;***************************************************************************

;***************************************************************************
; This code demonstrates how Microchip's MCP42xxx Digital Potentiometer
; (Pot) is interfaced to the PICmicro MCU (PIC16F876). The Potentiometer
; requires a serial communication to program the command byte and the data
; byte. This MCU has a built-in serial communication system which can be
; used to program the pot effectively. However, for this application three
; lines from PORTC are dedicated to serially program the Pot. This method
; is selected to clearly demonstrate the Digital pot programing sequence.
; PORTC:<3,5> are connected to SCK, and SI pins of the Pot, respectively.
; Chip Select is connected to PORTA: <4>.
; To change the command byte or the Pot wiper position the user must change
; the COMMAND and R_VALUE variables properly and reprogram the MCU.
;***************************************************************************

#include <p16f876.inc>
ERRORLEVEL -302
__CONFIG _BODEN_OFF & _PWRTE_OFF & _CP_OFF & _WDT_OFF & _XT_OSC
;*******************************************************************************************
;******************** VARIABLES  **********************************************************
;*******************************************************************************************

CBLOCK H'20'
OUT, COUNT ;VARIABLES USED TO TRANSMIT SERIAL DATA
ENDC

;*******************************************************************************************
;******************** EQUATES  ****************************************************************************
;*******************************************************************************************

CS   EQU H'00'; PORTC: <7> CHIP SELECT
SCLK EQU H'03'; <6> SERIAL CLOCK
SI   EQU H'05'; <5> SERIAL DATA
COMMAND EQU H'13'; VARIABLE FOR THE COMMAND BYTE
R_VALUE EQU D'140'; VARIABLE FOR THE RESISTANCE VALUE

;*******************************************************************************************
;******************** PROGRAM ORIGIN  ************************************************************
;*******************************************************************************************

ORG 0x00

;----------------------------------------------------------------------
;------------------- PORT SETTING -------------------------------
;----------------------------------------------------------------------

BCF STATUS, RP0 ;MAKE SURE TO BE IN BANK 0
BCF STATUS, RP1 ;

BSF STATUS, RP0 ;GO TO BANK 1
MOVLW 0x00
MOVF TRISA ;MAKE PORTA AN OUTPUT
MOVF TRISC ;MAKE PORTC AN OUTPUT
BCF STATUS, RP0 ;RETURN TO BANK 0

CLRF PORTC ;CLEAR PORTB

;----------------------------------------------------------------------
;------------------- PROGRAM ROUTINE ---------------------------------
;----------------------------------------------------------------------

BCF PORTA, CS ;SELECT THE POT
MOVLW COMMAND ;LOAD THE COMMAND BYTE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE COMMAND BYTE
MOVLW R_VALUE ;LOAD THE RESISTANCE VALUE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE RESISTANCE VALUE
BSF PORTA, CS ;UNSELECT THE POT
GOTO FINISH ;FINISH
TRANSMIT

; MOVE W TO 'OUT' VARIABLE
MOVWF OUT

; LOAD A COUNTER TO 'COUNT' THE BIT
MOVLW 0X08
MOVWF COUNT

; TRANSMISSION

L_SHIFT

; MONITOR THE 7TH BIT
BTFSC OUT, 7
GOTO HI

; IF LOW: CLEAR SERIAL-IN LINE
BCF PORTC, SI
GOTO PASS

; IF HI: SET SERIAL-IN LINE
HI

BSF PORTC, SI

PASS

; SET SERIAL CLOCK: HI
BSF PORTC, SCLK

; ROTATE OUT LEFT
RLF OUT, F

; SET SERIAL CLOCK: LOW
BCF PORTC, SCLK

; DECREMENT COUNTER UNTIL ITS ZERO
DECFSZ COUNT, F
GOTO L_SHIFT

; WHEN COUNTER IS ZERO IT'S END OF TRANSMISSION
CLRF PORTC
RETURN

; RETURN FROM SUBROUTINE

FINISH

GOTO FINISH

END

; END OF PROGRAM


Appendix C: RELOCATABLE VERSION OF THE HARDWARE SPI CODE IN APPENDIX A

;****************************************************************************
; Interfacing Microchip’s MCP42xxx digital potentiometer to the PICmicro MCU
; - This program is absolute assembly using the hardware
; SPI module to program the digital potentiometer
;
;****************************************************************************

;****************************************************************************
; Filename: POTSPI1.ASM
; Date: 11.07.2000
; File Version: 1.00
;
; Assembler: MPASM VERSION 2.50
; PROGRAMER: PRO MATE DEVICE PROGRAMMER, VERSION 5.20.00
; File Required: PIC16F876.inc
; Author: Ezana Haile
; Company: Microchip Technology Incorporated
;
;****************************************************************************

;*****************************************************************************
; This code demonstrates how Microchip’s MCP42xxx Digital Potentiometer
; (Pot) is interfaced to the PICmicro MCU (PIC16F876). The Potentiometer
; requires a serial communication to program the command byte and the data
; byte. This MCU has a built-in Serial Peripheral Interface (SPI) which can
; be used to program the pot effectively. The following program illustrates
; how to interface the digital pot using the MCU’s SPI.
; To change the command byte or the Pot wiper position the user must change
; the COMMAND and R_VALUE variables properly and reprogram the MCU.

*****************************************************************************

#include <p16f876.inc>
ERRORLEVEL -102
__CONFIG _BODEN_OFF & _PWRTE_OFF & _CP_OFF & _WDT_OFF & _XT_OSC
CS EQU H’00’ ;CHIP SELECT
COMMAND EQU H’13’ ;VARIABLE FOR THE COMMAND BYTE
R_VALUE EQU D’140’ ;VARIABLE FOR THE RESISTANCE VALUE

prog1 code

;------------------------------- PORTB AND SPI SETTING -------------------------------
BSF STATUS, RP0 ;SPECIFY BANK 1
MOVlw H’00’
MOVWF TRISA ;SET PORTA AS AN OUTPUT
MOVWF TRISC ;SET PORTB AS AN OUTPUT
BCF STATUS, RP0 ;SPECIFY BANK 0
CLRF PCLATH ;ENSURE PCLATH BIT 3 IS CLEARED
CLRF INTCON ;ENSURE ALL INTERRUPTS ARE DISABLED
MOVlw 0x30 ;
MOVWF SSPCON ;SET SYNC SERIAL PORT CONTROL REGISTER

;------------------------------- PROGRAM ROUTINE -------------------------------------
BCF PORTA, CS ;SELECT THE POT
MOVlw COMMAND ;LOAD THE COMMAND BYTE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE COMMAND BYTE
MOVlw R_VALUE ;LOAD THE RESISTANCE VALUE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE RESISTANCE VALUE
BSF PORTA, CS ;UNSELECT THE POT
GOTO FINISH ;FINISH

;------------------------------- TRANSMISSION SUBROUTINE -----------------------------
TRANSmit BCF STATUS, RP0 ;SPECIFY BANK 0
MOVWF SSPBUF ;PLACE DATA IN BUFFER TO SEND
BSF STATUS, RP0 ;SPECIFY BANK 1
LOOP BTFSS SSPSTAT, BF ;CHECK IF TRANSMISSION IS COMPLETE
GOTO LOOP ;
BCF STATUS, RP0 ;SPECIFY BANK 0
RETURN ;RETURN FROM SUBROUTINE
FINISH       GOTO       FINISH
END

;******************************* END OF PROGRAM ***************************************
Appendix D: RELOCATABLE VERSION OF THE Firmware SPI CODE IN APPENDIX B

;*******************************************************************************************
; *
; Interfacing Microchip's MCP42xxx digital potentiometer to the PICmicro MCU
; *
; - THIS PROGRAM IS ABSOLUTE ASSEMBLY USING PORTB TO ACT AS SOFTWARE
; SPI AND PROGRAM THE DIGITAL POTENTIOMETER
; *
;*******************************************************************************************

; Filename: POT_PRG.ASM
; Date: 11.02.2000
; File Version: 1.00
; Assembler: MPASM VERSION 2.50
; PROGRAMER: PRO MATE DEVICE PROGRAMMER, VERSION 5.20.00
; File Required: PIC16F876.inc
; Author: Ezana Haile
; Company: Microchip Technology Incorporated

;*******************************************************************************************

This code demonstrates how Microchip's MCP42xxx Digital Potentiometer (Pot) is interfaced to the PICmicro MCU (PIC16F876). The Potentiometer requires a serial communication to program the command byte and the data byte. This MCU has a built-in serial communication system which can be used to program the pot effectively. However, for this application three lines from PORTC are dedicated to serially program the Pot. This method is selected to clearly demonstrate the Digital pot programing sequence.

PORTC:<3,5> are connected to SCK, and SI pins of the Pot, respectively. Chip Select is connected to PORTA: <4>.

To change the command byte or the Pot wiper position the user must change the COMMAND and R_VALUE variables properly and reprogram the MCU.

;*******************************************************************************************

#include <p16f876.inc>
ERRORLEVEL -102
__CONFIG _BODEN_OFF & _PWRTE_OFF & _CP_OFF & _WDT_OFF & _XT_OSC
;******************************************************************************;
;******************  VARIABLES  **********************************************************
;******************************************************************************;
udata
OUT res 1 ;VARIABLE USED TO TRANSMIT SERIAL DATA
COUNT res 1 ;VARIABLE USED TO TRANSMIT SERIAL DATA

;******************************************************************************;
;******************  EQUATES  ************************************************************
;******************************************************************************;
CS EQU H'00' ;PORTC: <7> CHIP SELECT
SCLK EQU H'03' ; <6> SERIAL CLOCK
SI EQU H'05' ; <5> SERIAL DATA
COMMAND EQU H'13' ;VARIABLE FOR THE COMMAND BYTE
R_VALUE EQU D'140' ;VARIABLE FOR THE RESISTANCE VALUE

;******************************************************************************;
;******************  PROGRAM ORIGIN  ****************************************************
;******************************************************************************;
progl code
;
BCF STATUS, RP0 ;MAKE SURE TO BE IN BANK 0
BCF STATUS, RP1 ;
BSF STATUS, RP0 ;GO TO BANK 1
MOVLW 0x00
MOVF TRISA ;MAKE PORTA AN OUTPUT
MOVF TRISC ;MAKE PORTC AN OUTPUT
BCF STATUS, RP0 ;RETURN TO BANK 0
CLRF PORTC ;CLEAR PORTB

BCF PORTA, CS ;SELECT THE POT
MOVLW COMMAND ;LOAD THE COMMAND BYTE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE COMMAND BYTE
MOVLW R_VALUE ;LOAD THE RESISTANCE VALUE IN THE ACCUMULATOR
CALL TRANSMIT ;TRANSMIT THE RESISTANCE VALUE
BSF PORTA, CS ;UNSELECT THE POT
GOTO FINISH ;FINISH
TRANSMIT MOVWF OUT ;MOVE W TO 'OUT' VARIABLE

MOVlw 0x08 ;LOAD A COUNTER TO 'COUNT' THE BIT
MOVF W COUNT ;TRANSMISSION

L_SHIFT BTFSC OUT, 7 ;MONITOR THE 7TH BIT
GOTO HI ;IF LOW: CLEAR SERIAL-IN LINE
BCF PORTC, SI
GOTO PASS ;IF HI: SET SERIAL-IN LINE

HI BSF PORTC, SI

PASS BSF PORTC, SCLK ;SET SERIAL CLOCK: HI
RLF OUT, F ;ROTATE OUT LEFT
BCF PORTC, SCLK ;SET SERIAL CLOCK: LOW

DECFSZ COUNT, F ;DECREMENT COUNTER UNTIL ITS ZERO
GOTO L_SHIFT

CLR PORTC ;WHEN COUNTER IS ZERO IT'S END OF
;TRANSMISSION
RETURN ;RETURN FROM SUBROUTINE

;*******************************************************************************************
FINISH GOTO FINISH
END

;************************ END OF PROGRAM  ***********************************************
APPENDIX E: LINKER SCRIPT FILE

// File: 16f876.lkr
// Sample linker command file for 16F876

LIBPATH .

CODEPAGE   NAME=vectors  START=0x0      END=0x4      PROTECTED
CODEPAGE   NAME=page0    START=0x5      END=0x7FF
CODEPAGE   NAME=page1    START=0x800    END=0xFFF
CODEPAGE   NAME=page2    START=0x1000   END=0x17FF
CODEPAGE   NAME=page3    START=0x1800   END=0x1FFF
CODEPAGE   NAME=.idlocs  START=0x2000   END=0x2003   PROTECTED
CODEPAGE   NAME=.config  START=0x2007   END=0x2007   PROTECTED

DATABANK   NAME=sfr0     START=0x0      END=0x1F     PROTECTED
DATABANK   NAME=sfr1     START=0x80     END=0x9F     PROTECTED
DATABANK   NAME=sfr2     START=0x100    END=0x10F    PROTECTED
DATABANK   NAME=sfr3     START=0x180    END=0x18F    PROTECTED
DATABANK   NAME=gpr0     START=0x20     END=0x6F
DATABANK   NAME=gpr1     START=0xA0     END=0xEF
DATABANK   NAME=gpr2     START=0x110    END=0x16F
DATABANK   NAME=gpr3     START=0x190    END=0x1EF
SHAREBANK  NAME=gprnobnk START=0x70     END=0x7F
SHAREBANK  NAME=gprnobnk START=0xF0     END=0xFF
SHAREBANK  NAME=gprnobnk START=0x170    END=0x17F
SHAREBANK  NAME=gprnobnk START=0x1F0    END=0x1FF

SECTION    NAME=STARTUP  ROM=vectors    // Reset and interrupt vectors
SECTION    NAME=PROG1    ROM=page0      // ROM code space - page0
SECTION    NAME=PROG2    ROM=page1      // ROM code space - page1
SECTION    NAME=PROG3    ROM=page2      // ROM code space - page2
SECTION    NAME=PROG4    ROM=page3      // ROM code space - page3
SECTION    NAME=IDLOCS   ROM=.idlocs    // ID locations
SECTION    NAME=CONFIG   ROM=.config    // Configuration bits location
APPENDIX F: SCHEMATIC

[Diagram of electrical schematic with labels such as +5V, GND, Reset 1, U3, POR, VOLT Meter, MCLR/Vpp/THV, RA0, RC3, RC5, OSC1/CLKIN, VDD, VSS, U1, PIC16F876, U2, MCP42XXX, etc.]
“All rights reserved. Copyright © 2001, Microchip Technology Incorporated, USA. Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.”

Trademarks

The Microchip name, logo, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, KEELOQ, SEEVAL, MPLAB and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

Total Endurance, ICSP, In-Circuit Serial Programming, FilterLab, MXDEV, microID, FlexROM, fuzzyLAB, MPASM, MPLINK, MPLIB, PICDEM, ICEPIC, Migratable Memory, FanSense, ECONOMONITOR, SelectMode and microPort are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2001, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.