DALLAS SEMICONDUCTOR **MAXIM**

PIC'ing the MAX5581: Interfacing a PIC Microcontroller with the MAX5581 Fast-Settling DAC

This application note shows how to use a PIC microcontroller with the MAX5581 DAC. Detailed schematics and source code are provided.

MAX5581 Overview

The MAX5581 is a 12-bit, fast-settling DAC featuring a 3-wire SPI serial interface. The MAX5581's interface can support SPI up to 20MHz with a maximum settling time of 3µs. This application note presents an application circuit and all the firmware required to interface the fastest line of PIC microcontrollers (PIC18F core) to the MAX5581 DAC. The example assembly program was written specifically for the PIC18F442 using the free assembler provided in MPLAB IDE version 6.10.0.0.

Hardware Overview

The application circuit discussed here uses the MAX5581 Evaluation (EV) Kit, which consists of the MAX5581, an ultra-high-precision voltage reference (MAX6126), two pushbutton switches, gain setting resistors, and a proven PCB layout. The PIC18F442 is not present on the MAX5581EVKIT board, but was added to the system to complete the application schematic shown in **Figure 1**. The / CS\, SCLK, DIN, and DOUT pads on the MAX5581EVKIT allow an easy connection for the SPI serial interface.
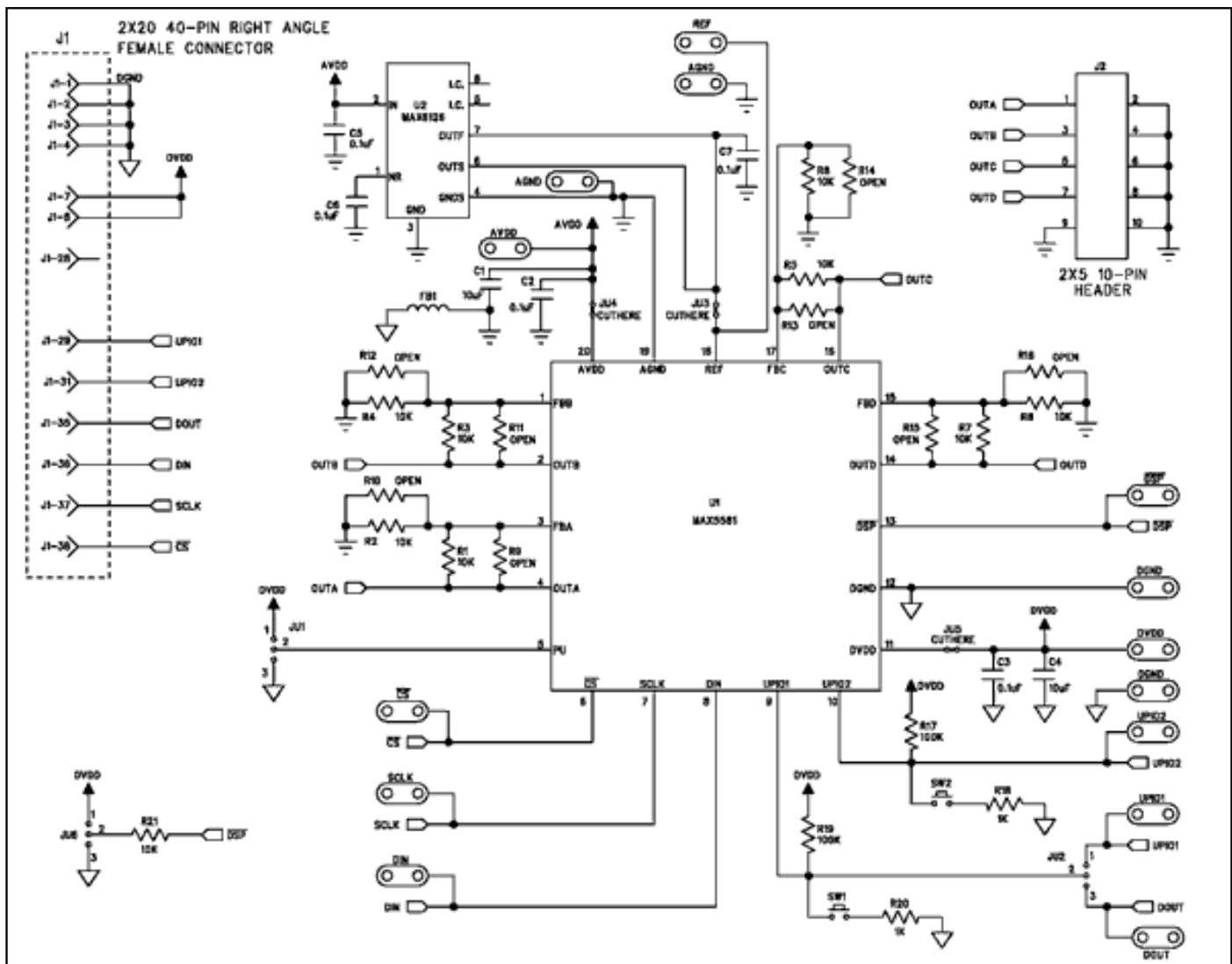
[For Larger Image](#)

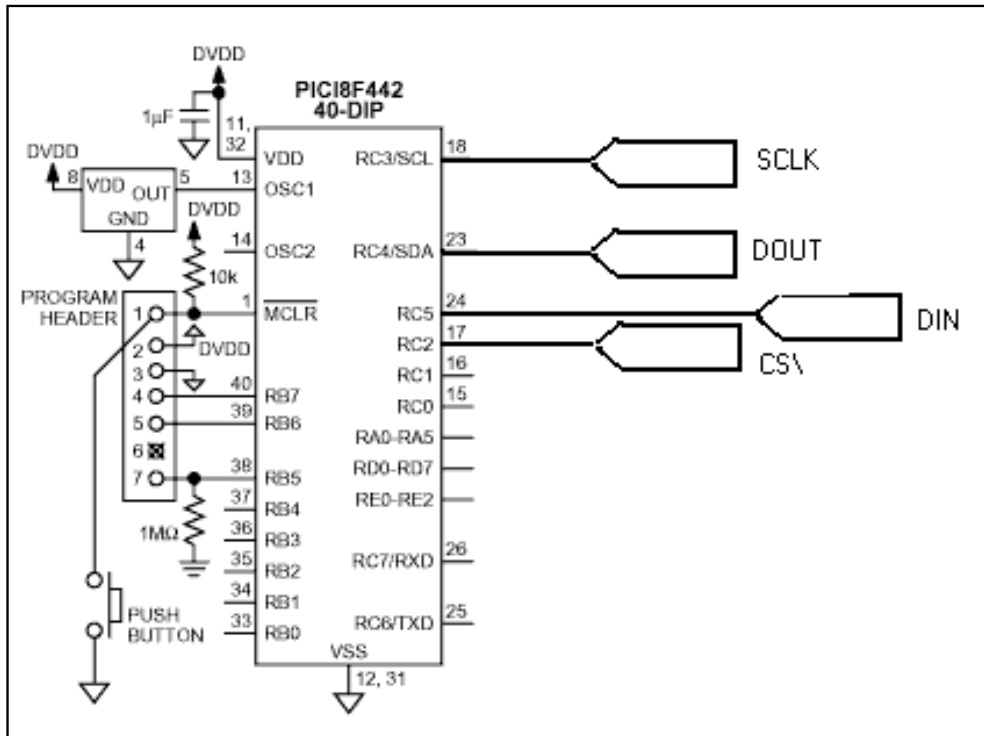*Figure 1. MAX5581 application schematic Sheet 1 of 2.*

Figure 1. MAX5581 application schematic Sheet 2 of 2.

Analog and Digital Ground Planes

It is good practice to separate the analog and digital ground planes, as shown in **Figure 2**. Use a ferrite bead, such as the TDK MMZ1608B601C, to connect both ground planes together through a ferrite bead. This prevents the microcontroller's system clock and its harmonics from feeding into the analog ground. Knowing that the PIC18F442's system clock is 40MHz, the MMZ1608B601C was chosen for its specific impedance vs. frequency characteristics. **Figure 3** shows the impedance versus frequency curve for the MMZ1608B601C.
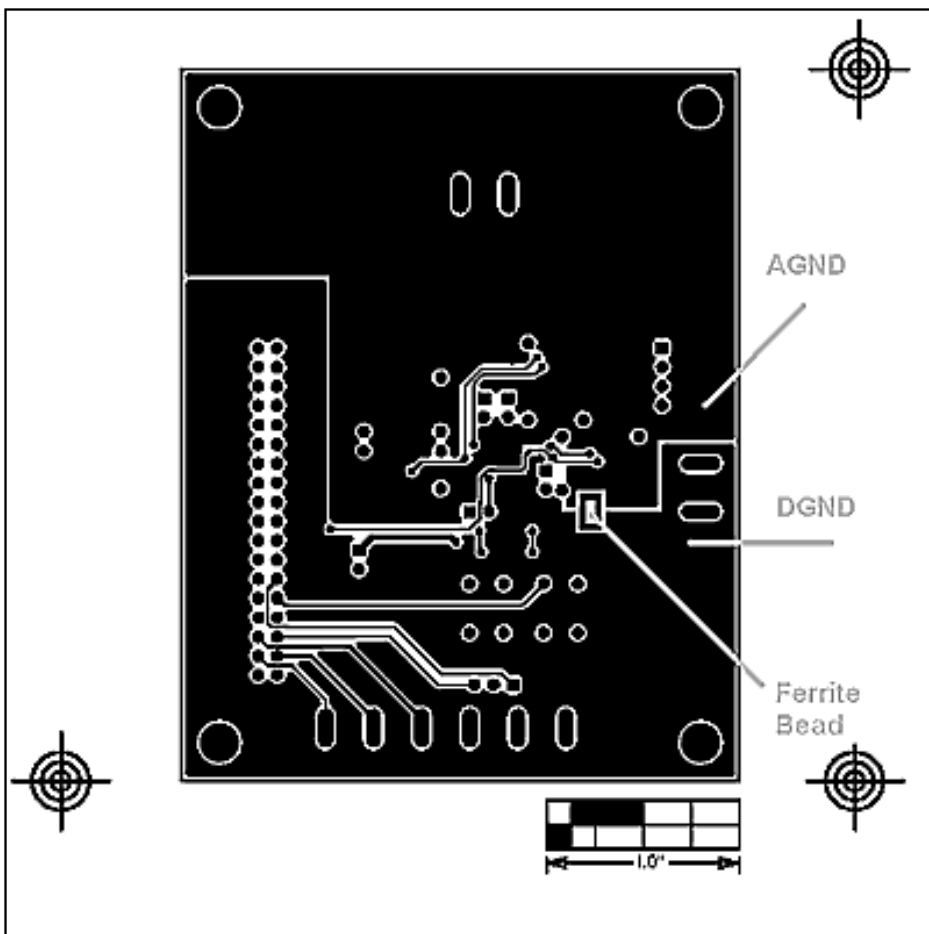
Figure 2. Separating Analog and Digital Grounds.
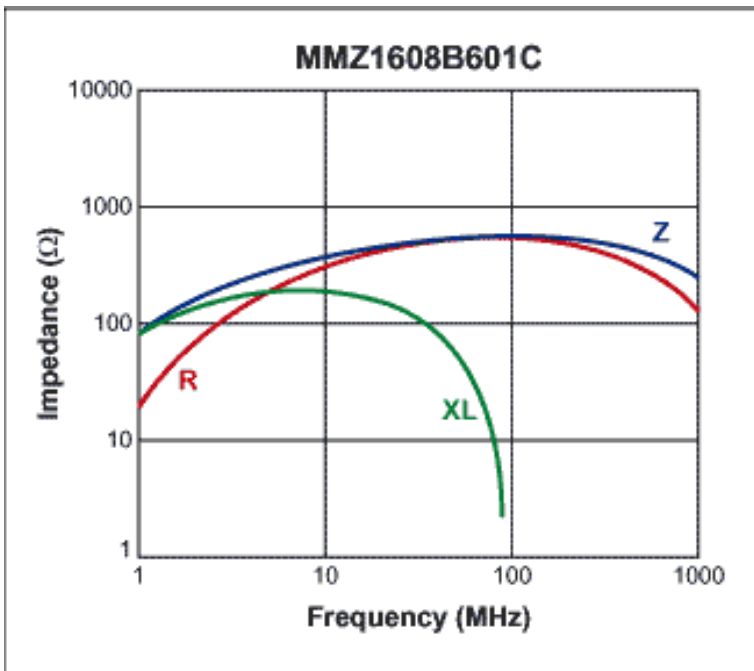
Figure 3. Impedance vs. Frequency Curve for the TDK MMZ1608B601C Ferrite Bead.

Firmware Overview

The example assembly program shown in **Listing 1** initializes the MAX5581 using the PIC18F442's internal MSSP SPI peripheral. The PIC18F442's 40MHz system clock allows the MSSP to provide an SPI clock (SCLK) up to 10MHz. **Table 1** shows the only configuration word required after power. Once the MAX5581 is initialized, the program constantly loads the DAC output registers with zero scale followed by full scale, as shown in Table 2. This constant loop results in a square wave, shown in **Figure 4**, which demonstrates the fast settling time of the MAX5581.
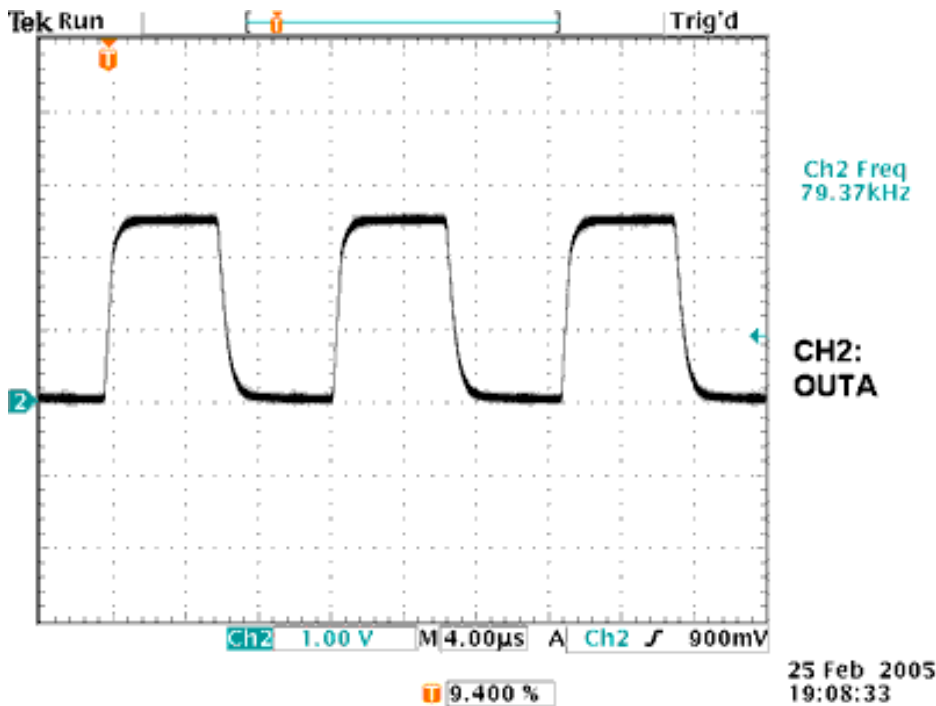
Figure 4. An Actual Scope Shot of the 80kHz Square Wave.

Listing 1. An Assembly Example Program that Interfaces to the MAX5581 Using the PIC18F442's Internal MSSP SPI Peripheral

Download: P18F442.INC

Listing 1.asm

```
;******************************************************************************
;
;     Filename:          Listing 1 (Absolute Code Version)
;     Date:              2/25/05
;     File Version:      1.0
;
;     Author:            Ted Salazar
;     Company:           Maxim
;
;******************************************************************************
;
;         Program Description:
;
;         This program interfaces the internal SPI MSSP
;         (Peripheral) of the PIC18F442 to the MAX5581 SPI
;         Quad DAC. The program initializes the MAX5581
;         and dynamically generates a 50% duty cycle square
;         wave with a frequency of 80KHz.
;
;
;******************************************************************************
;
; History:
; 2/25/05: Tested SPI DAC format
; 2/25/05: Initialized MAX5591
; 12/14/04: Cleared tcount timer in HWSPI_W_spidata_W
;******************************************************************************
;******************************************************************************


    ;
```

```
;*****************************************************************************
;
;    Files required:         P18F442.INC
;
;*****************************************************************************
        radix hex              ;Default to HEX
        LIST P=18F442, F=INHX32 ;Directive to define processor and file format
        #include        ;Microchip's Include File
;*****************************************************************************
;*****************************************************************************
xmit    equ             06              ; Asynchronous TX is at C6
;
;*****************************************************************************
;Configuration bits
; The __CONFIG directive defines configuration data within the .ASM file.
; The labels following the directive are defined in the P18F442.INC file.
; The PIC18FXX2 Data Sheet explains the functions of the configuration bits.
; Change the following lines to suit your application.

;T      __CONFIG        _CONFIG1H, _OSCS_OFF_1H & _RCIO_OSC_1H
;T      __CONFIG        _CONFIG2L, _BOR_ON_2L & _BORV_20_2L & _PWRT_OFF_2L
;T      __CONFIG        _CONFIG2H, _WDT_ON_2H & _WDTPS_128_2H
;T      __CONFIG        _CONFIG3H, _CCP2MX_ON_3H
;T      __CONFIG        _CONFIG4L, _STVR_ON_4L & _LVP_OFF_4L & _DEBUG_OFF_4L
;T      __CONFIG        _CONFIG5L, _CP0_OFF_5L & _CP1_OFF_5L & _CP2_OFF_5L & _CP3_OFF_5L
;T      __CONFIG        _CONFIG5H, _CPB_ON_5H & _CPD_OFF_5H
;T      __CONFIG        _CONFIG6L, _WRT0_OFF_6L & _WRT1_OFF_6L & _WRT2_OFF_6L & _WRT3_OFF_6L
;T      __CONFIG        _CONFIG6H, _WRTC_OFF_6H & _WRTB_OFF_6H & _WRTD_OFF_6H
;T      __CONFIG        _CONFIG7L, _EBTR0_OFF_7L & _EBTR1_OFF_7L & _EBTR2_OFF_7L & _EBTR3_OFF_7L
;T      __CONFIG        _CONFIG7H, _EBTRB_OFF_7H


;*****************************************************************************
;Variable definitions
; These variables are only needed if low priority interrupts are used.
; More variables may be needed to store other special function registers used
; in the interrupt routines.

                CBLOCK  0x080
                WREG_TEMP       ;variable used for context saving
                STATUS_TEMP     ;variable used for context saving
                BSR_TEMP        ;variable used for context saving
                ;
                ENDC

                CBLOCK  0x000
                EXAMPLE ;example of a variable in access RAM
                ;
                temp            ;
                temp2
                ;
                xmtreg          ;
                cntrb           ;
                cntra           ;
                bitctr          ;

                tcount  ;
                speedLbyte      ;T Being used in HWSPI_speed
                ;
                ENDC
;*****************************************************************************
;Reset vector
; This code will start executing when a reset occurs.
```

```
                ORG     0x0000

                goto    Main    ;go to start of main code

;****************************************************************************
;High priority interrupt vector
; This code will start executing when a high priority interrupt occurs or
; when any interrupt occurs if interrupt priorities are not enabled.

                ORG     0x0008

                bra     HighInt ;go to high priority interrupt routine

;****************************************************************************
;Low priority interrupt vector and routine
; This code will start executing when a low priority interrupt occurs.
; This code can be removed if low priority interrupts are not used.

                ORG     0x0018

                movff   STATUS,STATUS_TEMP      ;save STATUS register
                movff   WREG,WREG_TEMP          ;save working register
                movff   BSR,BSR_TEMP            ;save BSR register

;         *** low priority interrupt code goes here ***


                movff   BSR_TEMP,BSR            ;restore BSR register
                movff   WREG_TEMP,WREG          ;restore working register
                movff   STATUS_TEMP,STATUS      ;restore STATUS register
                retfie

;****************************************************************************
;High priority interrupt routine
; The high priority interrupt code is placed here to avoid conflicting with
; the low priority interrupt vector.

HighInt:

;         *** high priority interrupt code goes here ***


                retfie  FAST

;****************************************************************************
;Start of main program
; The main program code is placed here.

Main:

;         *** main code goes here ***
start
;         *** Port Initialization ***
                movlw   0x0FF
                movwf   PORTB
                clrf    PORTA
                movlw   0x06            ;T Configure PortA as Digital
                movwf   ADCON1
                movlw   0x00FB          ;T A2 OUTPUT, ALL OTHERS INPUT
                movwf   TRISA
                movlw   0x0001          ;T B0 INPUT, ALL OTHERS OUTPUT
```

```
                movwf    TRISB

                movlw    0x0093          ;T C7-C0 => bit7-0
                                         ;T OUTPUTs: C6(TX), C5(MOSI), C3(SCLK), C2(CS)
                                         ;T INPUTs:C4 (MISO) and all others
                movwf    TRISC           ;T TRISC bit3 Master = 0
                bsf      PORTC,RC2       ;T RC2 = CS\ Make CS\ high

;        *** SPI Initialization ***
                call     HWSPI_init      ;T Initialize the MSSP for SPI
;        *** SPI Configuration ***
                movlw    b'00000000'     ;T load W with test byte for CPOLCPHA 0,0
                                         ;T b'00000000' => CPOLCPHA 0,0
                                         ;T b'00000001' => CPOLCPHA 0,1
                                         ;T b'00000010' => CPOLCPHA 1,0
                                         ;T b'00000011' => CPOLCPHA 1,1
                call     HWSPI_W_configure
;        *** SPI Speed   ***
                movlw    b'00000000'     ;T load W with test byte for SPI Freq
                                         ;T b'00000000' => Fosc/4  = 10MHz
                                         ;T b'00000001' => Fosc/16 = 2.5Mhz
                                         ;T b'00000010' => Fosc/64 = 625kHz
                                         ;T b'00000011' => Reserved.
                call     HWSPI_W_speed
;********************************************************************************
;        *** MAX5581 Initialization ***
                bcf      PORTC,RC2               ;T RC2 = CS\ Make CS\ Low
                movlw    0xEC                    ;T byte0 of settling time config
                call     HWSPI_W_spidata_W       ;T HW SPI WriteRead Operation
                movlw    0x0F                    ;T byte1 of settling time config
                call     HWSPI_W_spidata_W       ;T HW SPI WriteRead Operation
                bsf      PORTC,RC2               ;T RC2 = CS\ Make CS\ high
;        *** MAX5581 Load All DAC Outputs to Zero Scale ***
Loopforever     bcf      PORTC,RC2               ;T RC2 = CS\ Make CS\ Low
                movlw    0xD0                    ;T byte0 of load all input/output to zeros
                call     HWSPI_W_spidata_W       ;T HW SPI WriteRead Operation
                movlw    0x00                    ;T byte1 of load all input/output to zeros
                call     HWSPI_W_spidata_W       ;T HW SPI WriteRead Operation
                bsf      PORTC,RC2               ;T RC2 = CS\ Make CS\ high
;        *** MAX5581 Load All DAC Outputs to Full Scale ***
                bcf      PORTC,RC2               ;T RC2 = CS\ Make CS\ Low
                movlw    0xDF                    ;T byte0 of load all input/output to zeros
                call     HWSPI_W_spidata_W       ;T HW SPI WriteRead Operation
                movlw    0xFF                    ;T byte1 of load all input/output to zeros
                call     HWSPI_W_spidata_W       ;T HW SPI WriteRead Operation
                bsf      PORTC,RC2               ;T RC2 = CS\ Make CS\ high
;        movwf   xmtreg                          ;T move w to xmtreg
;        call    asyxmtc                         ;T call UART routine
;
                goto     Loopforever             ;T loop forever
;********************************************************************************
errsrv
                movlw    0x65            ; load w with 'e' = 0x65
                movwf    xmtreg          ; move w to xmtreg
                call     asyxmtc         ; call UART routine
dead            goto     dead                            ; goto endless loop
;********************************************************************************
set_cf_error
                movlw    0x00            ; 0x00 into W
                sublw    0x00            ; Subtract W-0x00: If W<=N C set; If W>N C clear.
                return                   ; error=> cf=set
;********************************************************************************
```

```
clear_cf_ok
                movlw   0x01            ; 0x00 into W
                sublw   0x00            ; Subtract W-0x00: If W<=N C set; If W>N C clear.
                return                  ; success=> cf=clear
;******************************************************************************
HWSPI_init                             ;T SPI MSSP Initialization for M2EAM schematic
                                       ;T CPOL,CPHA = 0,0  => CKP = 0 & CKE = 1

                bcf     SSPCON1,SSPEN   ;T Disable the MSSP, SSPCON-5
;
                bcf     TRISC,SDO       ;T TRISC bit5 RC5/SDO = 0 MOSI Output
                bcf     TRISC,SCK       ;T TRISC bit3 RC3/SCK = 0 SCLK Output
                bsf     TRISC,SDI       ;T TRISC bit4 RC4/SDI = 1 MISO Input
                movlw   0x0040          ;T SSPSTAT bit8 = 0 sampled in middle
                                        ;T SSPSTAT bit6 = CKE = 1
                movwf   SSPSTAT         ;T Used to be sspstat on older PICs
                movlw   0x0020          ;T SSPCON1 bit5 SSPEN = 1 Enables sycn serial port
                                        ;T SSPCON1 bit4 = CKP = 0
                                        ;T SSPCON1 bit3= 0 = Turn MSSP ON for SPI
                                        ;T SSPCON1 bit2-0 = 000b = SCLK = Fosc/4
                                        ;T SSPCON1 bit2 = 0 = Master
                movwf   SSPCON1         ;T Used to be sspcon on older PICs
                bsf     INTCON,PEIE     ;T INTCON bit6 = PEIE = 1 = Enable periph interrupt
                bsf     PIE1,SSPIE      ;T PIE1 bit3 = SSPIE = 1 = interrupt enable
                movlw   0x00            ;T load 0x00 into W
                movwf   tcount          ;T initialize tcount to zero (0x00)
;******************************************************************************
HWSPI_W_configure
;Configure SPI Mode
;
;On Entry:       WREG = confDATA
;On Exit:
;On Success: return with C flag clear
;On Failure: return with C flag set
;
                bcf     SSPCON1,SSPEN   ;T Disable the MSSP, SSPCON1-5
                movwf   temp            ;T move the confDATA byte to temp
                btfsc   SSPCON1,SSPM3   ;T In SPI Mode?, skip if yes
                call    HWSPI_init      ;T MSSP is in wrong mode, Init for SPI
;
                btfsc   temp,1          ;T Is bit1 of confDATA byte clear? if so skip next
                goto    CPOL_1          ;T goto CPOL = 1 label => CPOL = 1
                btfsc   temp,0          ;T Is bit0 of confDATA byte clear? if so skip next
                                        ;T => CPOL = 0 , CPHA = ?
                goto    CPOLCPHA_01     ;T goto => CPOL = 0 CPHA = 1
;Configure for CPOL = 0, CPHA = 0
                bcf     SSPCON1,CKP     ;T SSPCON1 bit4 = CKP = 0
                bsf     SSPSTAT,CKE     ;T SSPSTAT bit6 = CKE = 1
                btfsc   SSPCON1,CKP     ;T Is SSPCON1 bit4 = CKP = 0 ?
                goto    badjump         ;T CKP bit test error
                btfss   SSPSTAT,CKE     ;T Is SSPSTAT bit6 = CKE = 1 ?
                goto    badjump         ;T CKE bit test error
                goto    okjump2         ;OK configured!
;
CPOL_1          btfsc   temp,0          ;T Is bit0 of confDATA byte clear? if so skip next
                                        ;T CPOL = 1 , CPHA = ?
                goto    CPOLCPHA_11     ;T goto => CPOL = 1, CPHA = 1
;Configure for CPOL = 1, CPHA = 0
                bsf     SSPCON1,CKP     ;T SSPCON1 bit4 = CKP = 1
                bsf     SSPSTAT,CKE     ;T SSPSTAT bit6 = CKE = 1
                btfss   SSPCON1,CKP     ;T Is SSPCON1 bit4 = CKP = 1 ?
                goto    badjump         ;T CKP bit test error
```

```
                    btfss    SSPSTAT,CKE      ;T Is SSPSTAT bit6 = CKE = 1 ?
                    goto     badjump          ;T CKE bit test error
                    goto     okjump2          ;OK configured!
;
  CPOLCPHA_01
  ;configure for CPOL = 0, CPHA = 1
                    bcf      SSPCON1,CKP      ;T SSPCON1 bit4 = CKP = 0
                    bcf      SSPSTAT,CKE      ;T SSPSTAT bit6 = CKE = 0
                    btfsc    SSPCON1,CKP      ;T Is SSPCON1 bit4 = CKP = 0 ?
                    goto     badjump          ;T CKP bit test error
                    btfsc    SSPSTAT,CKE      ;T Is SSPSTAT bit6 = CKE = 0 ?
                    goto     badjump          ;T CKE bit test error
                    goto     okjump2          ;OK configured!
;
  CPOLCPHA_11
  ;configure for CPOL = 1, CPHA = 1
                    bsf      SSPCON1,CKP      ;T SSPCON1 bit4 = CKP = 1
                    bcf      SSPSTAT,CKE      ;T SSPSTAT bit6 = CKE = 0
                    btfss    SSPCON1,CKP      ;T Is SSPCON1 bit4 = CKP = 1 ?
                    goto     badjump          ;T CKP bit test error
                    btfsc    SSPSTAT,CKE      ;T Is SSPSTAT bit6 = CKE = 0 ?
                    goto     badjump          ;T CKE bit test error
                    goto     okjump2          ;OK configured!
;
  okjump2           bsf      SSPCON1,SSPEN    ;T Re-enable MSSP
                    goto     clear_cf_ok
                    return
  badjump bsf       SSPCON1,SSPEN            ;T Re-enable MSSP
                    goto     set_cf_error     ;T configuration error
                    return
;*****************************************************************************
  HWSPI_W_speed
  ;On Entry:        WREG = speedDATA & checks SSPCON1-3 for SPI mode
  ;                 speedDATA = 0x00 => Fosc/4
  ;                 speedDATA = 0x01 => Fosc/16
  ;                 speedDATA = 0x02 => Fosc/64
  ;                 speedDATA = 0x03 => Timer Divisor (Not working yet)
  ;
  ;On Exit:
  ;On Success: return with C flag clear
  ;On Failure: return with C flag set
  ;
                    bcf      SSPCON1,SSPEN    ;T Disable MSSP
                    movwf    speedLbyte       ;T move speedDATA stored in W to speedLbyte
                    btfsc    SSPCON1,SSPM3    ;T In SPI Mode?, skip if yes
                    call     HWSPI_init       ;T MSSP is in wrong mode, Init for SPI
  ;
  ;Test if speedLbyte = 0x00. If yes, SPI clock speed = Fosc/4
                    movlw    0x00             ;T load 0x00 into W
                    subwf    speedLbyte,W     ;T subtract 0x00 from tcount result in w
                    btfss    STATUS,Z         ;T test zero flag, skip next instr if z set
                    goto     fdiv16           ;T goto Fosc/16 section
                    bcf      SSPCON1,SSPM1    ;T SSPCON1-1 = 0
                    bcf      SSPCON1,SSPM0    ;T SSPCON1-0 = 0
                    goto     okjump3          ;T Fosc/4 was selected
  ;Test if speedLbyte = 0x01. If yes, SPI clock speed = Fosc/16
  fdiv16            movlw    0x01             ;T load 0x01 into W
                    subwf    speedLbyte,W     ;T subtract 0x01 from tcount result in w
                    btfss    STATUS,Z         ;T test zero flag, skip next instr if z set
                    goto     fdiv64           ;T goto Fosc/64 section
                    bcf      SSPCON1,SSPM1    ;T SSPCON1-1 = 0
                    bsf      SSPCON1,SSPM0    ;T SSPCON1-0 = 1
```

```
                goto    okjump3             ;T Fosc/16 was selected
;Test if speedLbyte = 0x02. If yes, SPI clock speed = Fosc/64
fdiv64          movlw   0x02                ;T load 0x02 into W
                subwf   speedLbyte,W        ;T subtract 0x02 from tcount result in w
                btfss   STATUS,Z            ;T test zero flag, skip next instr if z set
                goto    timer               ;T goto Timer section
                bsf     SSPCON1,SSPM1       ;T SSPCON1-1 = 1
                bcf     SSPCON1,SSPM0       ;T SSPCON1-0 = 0
                goto    okjump3             ;T Fosc/64 was selected
;Test if speedLbyte >= 0x03. If yes, SPI clock speed will be set by the timer
;SETTING THE SPI CLOCK WITH THE TIMER WILL RETURN A FAILURE AT THIS TIME.
;Future To do: Implement the TIMER section
timer           movlw   0x03                ;T load 0x02 into W
                subwf   speedLbyte,W        ;T subtract 0x02 from tcount result in w
                btfss   STATUS,Z            ;T test zero flag, skip next instr if z set
                goto    badjmp2             ;T goto error section to return failure
                goto    badjmp2             ;T goto error section to return failure
;               bsf     SSPCON1,SSPM1       ;T SSPCON1-1 = 1
;               bsf     SSPCON1,SSPM0       ;T SSPCON1-0 = 1
;               goto    okjump3             ;T Fosc/64 was selected

okjump3         bsf     SSPCON1,SSPEN       ;T Re-enable MSSP
                bcf     STATUS,C            ;T clear c flag on success
                return

badjmp2         bsf     SSPCON1,SSPEN       ;T Re-enable MSSP
                bsf     STATUS,C            ;T set c flag on failure
                return
;****************************************************************************
HWSPI_W_spidata_W
;Simultaneously write SPI data on MOSI and read SPI data on MISO
;
;on Entry:      WREG = mosiDATA & checks bit3 of SSPCON1 for SPI mode
;On Exit:       WREG = misoDATA
;On Success: return with C flag clear
;On Failure: return with C flag set
;
                movwf   temp2               ;T move mosiDATA stored in W to WREG_TEMP
                btfsc   SSPCON1,SSPM3       ;T In SPI Mode?, skip if yes
                call    HWSPI_init          ;T MSSP is in wrong mode, Init for SPI
                movf    temp2,W             ;T load W with original mosiDATA
;
                movwf   SSPBUF              ;T move byte to transmit to SSPBUF (transmit buffer)
                movlw   0x00                ;T load 0x00 into W
                movwf   tcount              ;T initialize tcount to zero (0x00)
again1          btfsc   SSPSTAT,BF          ;T receive completed? if no, skip next
                goto    okjump1             ;T no. goto again
                incf    tcount,F            ;T increment tcount
                movlw   0xFF                ;T load w with literal
                subwf   tcount,W            ;T subtract 0xFF from tcount result in w
                btfss   STATUS,Z            ;T test zero flag, skip next instr if z set
                goto    again1              ;T loop until timeout
                goto    set_cf_error        ;T receive timeout error
                return
okjump1         movf    SSPBUF,W            ;T put received data in W
                goto    clear_cf_ok
                return
;****************************************************************************
; UART routine
asyxmtc         bcf     PORTC,xmit          ;T used to be portc,xmit
                call    full
                movlw   0x08                ;TEST_T "08"
```

```
               movwf    bitctr
asyxmt1        rrcf     xmtreg,f
               btfsc    STATUS,C
               goto     asyxmt2
               bcf      PORTC,xmit        ;T used to be portc,xmit
               goto     asyxmt3
asyxmt2        bsf      PORTC,xmit        ;T used to be portc,xmit
;
asyxmt3        call     full
               decfsz   bitctr,f
               goto     asyxmt1
;
               bsf      PORTC,xmit        ;T used to be portc,xmit
               call     full
               retlw    0
;*****************************************************************************
; UART baud rate of 115.2kbps using a 40MHz System Clock
full           movlw    d'3'
               movwf    cntrb
vdly0          movlw    d'6'              ; d'43' with 4MHz => 2400 baud
               movwf    cntra
vdly1          decfsz   cntra,f
               goto     vdly1
               decfsz   cntrb,f
               goto     vdly0
               retlw    0
;*****************************************************************************
;End of program

               END
```

Table 1. Configuration Write Command for Setting the Settling Time to 3µs for All Four DACs.

SPI Line	C7	C6	C5	C4	C3	C2	C1	C0	D7	D6	D5	D4	D3	D2	D1	D0
DIN	1	1	1	0	1	1	0	0	0	0	0	0	1	1	1	1

Table 2. Load All DAC Output Commands.

SPI Line	C3	C2	C1	C0	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
DIN (1st)	1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0
DIN (2nd)	1	1	0	1	1	1	1	1	1	1	1	1	1	1	1	1

In Table 2, the first command sets all the DAC outputs to zero scale. The second command sets all the DAC outputs to full scale.

More Information

MAX5581: QuickView -- [Full (PDF) Data Sheet](#) -- [Free Samples](#)

MAX5582: QuickView -- [Full (PDF) Data Sheet](#) -- [Free Samples](#)

MAX5583: QuickView -- [Full (PDF) Data Sheet](#) -- [Free Samples](#)

MAX5584: QuickView -- [Full (PDF) Data Sheet](#) -- [Free Samples](#)

MAX5585: QuickView -- [Full (PDF) Data Sheet](#) -- [Free Samples](#)