EE 496 – Independent Study
The Pennsylvania State University
Summer 2009

Student Name: Devin R. Ott
Faculty Advisor: Dr. Julio Urbina

Devin R. Ott

[Incomplete] PICBasic Pro Source Code for PIC18F4620 Microcontroller

Microcontroller Solution Intended for implementation in:

The PSU AES Amplifier Design Project

(A Hi-Fi Audio Power Amplifier Application)

Links:

PIC18F4620
http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010304
PIC18F4620 Datasheet
http://ww1.microchip.com/downloads/en/DeviceDoc/39626e.pdf
PICBasic Pro Compiler
http://melabs.com/products/pbp.htm
PICBasic Pro Compiler Manual
http://melabs.com/downloads/pbpm108.pdf
20x2 LCD Display
http://www.matrixorbital.com/mopal202cbftw-p-521.html
20x2 LCD Display Manual
http://www.matrixorbital.ca/manuals/MOP_series/MOP-AL/MOP-AL202C-E_2.pdf
Hitachi HD44780U (LCD-II) Dot Matrix Liquid Crystal Display Controller/Driver
http://www.matrixorbital.ca/manuals/MOP_series/MOP-AL/hd44780u.pdf

```
` Devin Robert Ott
` EE 496 Independent Study
` Summer 2009

DEFINE OSC 40       `Tells the PicBasicPro Compiler that my oscillator frequency is 40MHz.
                    `The PicBasicPro default assumes fosc=4MHz, yielding a 1us instruction time.
                    `It must be told of the actual fosc value in order to create accurate delays.

`CONFIGURE LCD SETTINGS
DEFINE LCD_DREG   PORTB         `Set LCD data port.
DEFINE LCD_DBIT   4             `Set starting bit of 4-bit LCD data.
                                `I'm using top 4 bits of PORTB (pins RB4-RB7).
DEFINE LCD_RSREG  PORTB         `Set Register Select (RS) port.
DEFINE LCD_RSBIT  3             `Set Register Select (RS) bit.
DEFINE LCD_EREG   PORTB         `Set Enable (E) port.
DEFINE LCD_EBIT   2             `Set Enable (E) bit.
DEFINE LCD_BITS   4             `Set parallel data length (4-bit or 8-bit)
DEFINE LCD_LINES  2             `Set number of LCD lines (my LCD is 2x20
DEFINE LCD_COMMANDUS    1500    `Set LCD command delay time [microseconds]
DEFINE LCD_DATAUS 50            `Set [us] delay time between data transfer


DEFINE ADC_BITS 10
DEFINE ADC_CLOCK 4
DEFINE ADC_SAMPLEUS 50
Set A/D Config Registers
`NOTE



`Assume program cycle execution time is 10ms (REPEATING)

Disable Master Clear (I'm using it as a digital I/O
Set TRIS registers
```

```
`Renaming digital INPUT pins
POWER       VAR   PORTD.4
DISPLAY     VAR   PORTD.5
AUX1        VAR   PORTC.5
AUX2        VAR   PORTC.6
AUX3        VAR   PORTC.7
MUTE        VAR   PORTC.4
ATTENUATE   VAR   PORTD.3
OFFSET_L    VAR   PORTC.1
OFFSET_R    VAR   PORTD.2

`Renaming digital OUTPUT pins
RED_LED_L   VAR   PORTB.0
GREEN_LED_L VAR   PORTB.1
RED_LED_R   VAR   PORTD.6
GREEN_LED_R VAR   PORTD.7
POWER_L     VAR   PORTC.2
POWER_R     VAR   PORTD.0
OUTPUT_L    VAR   PORTC.3
OUTPUT_R    VAR   PORTD.1
AUX1_RELAY  VAR   PORTE.3
AUX2_RELAY  VAR   PORTA.4
AUX3_RELAY  VAR   PORTE.2
ATTENUATE_RELAY   VAR   PORTC.0


`VARIABLES for DELAYS and COUNTING ---------------------------

TempSampleDelay   VAR   BYTE  `Create 8-bit variable for 2.5 sec delay of SAMPLE_TEMP subroutine
TempSampleDelay = 0       `Initialize to zero so SAMPLE_TEMP is executed in first execution cycle

flash VAR BYTE     `Create index variable for all flashing LED delay loops (FOR..NEXT)


`DATA VARIABLES ------------------------------------------------

`STEREO_INPUT subroutine data
StereoInput VAR BYTE     `Create variable to hold stereo input selection (1-3)
```

```
StereoInput = 1          `Initialize input selection to be AUX1.  This will be the default input

                         `selection after a device reset.
`TEMP_SAMPLE subroutine data
TempAs      VAR   WORD         `Create variable to hold ambient temperature 10-bit sample
TempLs      VAR   WORD         `Create variable to hold left heatsink temperature 10-bit sample
TempRs      VAR   WORD         `Create variable to hold right heatsink temperature 10-bit sample

TempAC100   VAR   WORD         `Create variable to hold ambient C temp*100
TempAF100   VAR   WORD         `Create variable to hold ambient F temp*100
TempLC100   VAR   WORD         `Create variable to hold C temp*100 of left heatsink
TempLF100   VAR   WORD         `Create variable to hold F temp*100 of left heatsink
TempRC100   VAR   WORD         `Create variable to hold C temp*100 of right heatsink
TempRF100   VAR   WORD         `Create variable to hold F temp*100 of right heatsink

`IVP_SAMPLE subroutine data
VoutLs      VAR   WORD         `Create variable to hold 10-bit sample of VOUT_L
VoutRs      VAR   WORD         `Create variable to hold 10-bit sample of VOUT_R
IoutLs      VAR   WORD         `Create variable to hold 10-bit sample of IOUT_L
IoutRs      VAR   WORD         `Create variable to hold 10-bit sample of IOUT_R

VoutLsmax   VAR   WORD         `Create variable to hold highest 10-bit sample of VOUT_L
VoutRsmax   VAR   WORD         `Create variable to hold highest 10-bit sample of VOUT_R
IoutLsmax   VAR   WORD         `Create variable to hold highest 10-bit sample of IOUT_L
IoutRsmax   VAR   WORD         `Create variable to hold highest 10-bit sample of IOUT_R

VoutLmax1000      VAR   WORD   `Create variable to hold VoutLmax value in [mV] units
VoutRmax1000      VAR   WORD   `Create variable to hold VoutRmax value in [mV] units
IoutLmax1000      VAR   WORD   `Create variable to hold IoutLmax value in [mA] units
IoutRmax1000      VAR   WORD   `Create variable to hold IoutRmax value in [mA] units
PoutLmax1000      VAR   LONG   `Create variable to hold PoutLmax value in [mW] units
PoutRmax1000      VAR   LONG   `Create variable to hold PoutRmax value in [mW] units

IVP_UpdateDelay   VAR   BYTE   `Create delay variable to calculate max I,V,P values
                               once every 255 cycles (2.5 sec)
```

3 main body loops

```
`BEGINNING OF MAIN BODY LOOP (START, MAIN_ON, MAIN_OFF)
`WHEN EXECUTION LOOPS BACK TO THE BEGINNING OF THE PROGRAM, IT WILL BE DIRECTED (GOTO) TO THE
`START LABEL

START:      IF (POWER = 0) AND (PowerOn = 0) THEN
                GOTO MAIN_OFF
                `If POWER switch is OFF, and audio circuits are powered-down,
                `then start executing at MAIN_OFF.

            IF (POWER = 0) AND (PowerOn = 1) THEN
                GOSUB PWR_OFF_SEQUENCE
                GOTO MAIN_OFF
                `If POWER switch is OFF, and audio circuits are NOT powered-down,
                `then execute the PWR_DOWN_SEQUENCE subroutine and then start executing
                `at MAIN_OFF.

            IF (POWER = 1) AND (PowerOn = 0) THEN
                GOSUB PWR_ON_SEQUENCE
                GOTO MAIN_ON
                `If POWER switch is ON and Power-On sequence has not been
                `executed, then execute the PWR_ON_SEQUENCE subroutine and then start
                `executing at MAIN_ON.

            IF (POWER = 1) AND (PowerOn = 1) THEN
                GOTO MAIN_ON
                `If POWER switch is ON and audio circuits are powered-on, then start
                `executing at MAIN_ON.

MAIN_OFF:   incomplete code
            GOTO START
            this would perform the operations of the application when main power was off

MAIN_ON:    IF TempSampleDelay = 0 THEN
```

[PowerOn] is a bit flag
- set high during Power On Sequence
- cleared during Power OFF Sequence

```
        GOSUB TEMP_SAMPLE `If 255 program cycles have elapsed (2.5 sec),
                          `execute TEMP_SAMPLE subroutine.
        GOSUB JUDGE_TEMPS       `Evaluate heatsink temps for Hot Shutdown.
        TempSampleDelay = TempSampleDelay + 1     `Increment TEMP_SAMPLE delay variable.
ENDIF                     `This variable will overflow (reset) after 255
                          `program execution cycles.
```

Incomplete code

GOTO START

the [MAIN_ON] loop would perform the operations of the applications when main power is ON. This loop mostly executes the many subroutines of the application

```
        `Samples temperature input pins, calculates 100*C and 100*F values.
        `Executed after MCU Reset, at the beginning of the Power-On sequence,
        `and repetitively throughout program execution (every 2.5 seconds).
TEMP_SAMPLE:    ADCIN 0, TempAs    `Sample the TEMP_A input pin, store 10-bit value in TempAs
                ADCIN 1, TempLs    `Sample the TEMP_L input pin, store 10-bit value in TempLs
                ADCIN 2, TempRs    `Sample the TEMP_R input pin, store 10-bit value in TempRs

                TempAC100 = TempAs */ 1245     `Convert TempAs to centigrade*100
                TempLC100 = TempLs */ 2490     `Convert TempLs to centigrade*100
                TempRC100 = TempRs */ 2490     `Convert TempRs to centigrade*100

                TempAF100 = TempAs */ 2241 + 3200    `Convert TempAs to fahrenheit*100
                TempLF100 = TempLs */ 4482 + 3200    `Convert TempLs to fahrenheit*100
                TempRF100 = TempRs */ 4482 + 3200    `Convert TempRs to fahrenheit*100


        `Polls each of 3 stereo input buttons, stores selection 3 (1-3) in StereoInput
STEREO_INPUT:   IF MUTE = 1 THEN          `If MUTE button is engaged, it overrides the
                      AUX1_RELAY = 0      `input selection and all auxillary input relays
                      AUX2_RELAY = 0      `are left in their open position
                      AUX3_RELAY = 0
                ELSE
                      IF AUX1 = 1 THEN
                            StereoInput = 1
                            AUX1_RELAY = 1
                            AUX2_RELAY = 0
```

```
                AUX3_RELAY = 0
            ENDIF
            IF AUX2 = 1 THEN
                StereoInput = 2
                AUX1_RELAY = 0
                AUX2_RELAY = 1
                AUX3_RELAY = 0
            ENDIF
            IF AUX3 = 1 THEN
                StereoInput = 3
                AUX1_RELAY = 0
                AUX2_RELAY = 0
                AUX3_RELAY = 1
            ENDIF
        ENDIF
        RETURN

        `SAFELY TURNS ON BOTH RIGHT & LEFT AMPS (assuming no Hot Shutdown modes)
PWR_ON_SEQUENCE:  PowerOn = 1                `Set (=1) the Power-On flag bit.
        Shutdown_PowerReset = 1 `If one of the amps is in Hot Shutdown mode, the
                                `Shutdown_PowerReset flag indicates that the POWER
                                `has been cycled.
        GOSUB TEMP_SAMPLE       `Sample temperature inputs, convert to degrees C & F.
        IF (TempLC100 >= 9000) THEN HotShutdownL = 1
        IF (TempRC100 >= 9000) THEN HotShutdownR = 1

        IF (HotShutdownL = 0) AND (HotShutdownR = 0) THEN
                                `If neither amp is in Hot Shutdown mode, then...
            POWER_L = 1         `Turn ON power to left amp
            POWER_R = 1         `Turn ON power to right amp
            FOR flash = 1 TO 10     `Generate 2 sec delay with red and green
                                    `bicolor LEDs flashing back & forth.
                RED_LED_L = 1       `Turn OFF left Red LED
                GREEN_LED_R = 1     `Turn OFF right Green LED
                GREEN_LED_L = 0     `Turn ON left Green LED
                RED_LED_R = 0       `Turn ON right Red LED
                PAUSE 100           `Delay 100ms
                GREEN_LED_L = 1     `Turn OFF left Green LED
                RED_LED_R = 1       `Turn OFF right Red LED
```

```
            RED_LED_L = 0       'Turn ON left Red LED
            GREEN_LED_R = 0     'Turn ON right Green LED
            PAUSE 100           'Delay 100ms
        NEXT flash              'Increment the index variable, flash
        RED_LED_L = 1       'Turn OFF left Red LED
        GREEN_LED_R = 1     'Turn OFF right Green LED
        OUTPUT_R = 1            'Close right amp Output Relay

ENDIF

IF (HotShutdownL = 0) AND (HotShutdownR = 1) THEN AMP_STARTUP_L
                            'If right amp is in Shutdown mode,
                            'only turn ON the left amp.
IF (HotShutdownL = 1) AND (HotShutdownR = 0) THEN AMP_STARTUP_L
                            'If left amp is in Shutdown mode,
RETURN                      'only turn ON the right amp.
```

executed if Right Amp is disabled

```
'TURNS ON LEFT AMP.
AMP_STARTUP_L:  POWER_L = 1         'Turn ON power to left amp
                FOR flash = 1 TO 10     'Generate 2 sec delay with red and green
                                        'bicolor LED flashing back & forth.
                    RED_LED_L = 1       'Turn OFF Red LED
                    GREEN_LED_L = 0     'Turn ON Green LED
                    PAUSE 100           'Delay 100ms
                    GREEN_LED_L = 1     'Turn OFF Green LED
                    RED_LED_L = 0       'Turn ON Red LED
                    PAUSE 100           'Delay 100ms
                NEXT flash          'Increment the index variable, flash
                RED_LED_L = 1       'Turn OFF Red LED (Now all LEDs are OFF)
                OUTPUT_L = 1        'Close left amp Output Relay
                RETURN
```

executed if left Amp is disabled

```
'TURNS ON RIGHT AMP.
AMP_STARTUP_R:  POWER_R = 1         'Turn ON power to right amp
                FOR flash = 1 TO 10     'Generate 2 sec delay with red and green
```

```
                                    `bicolor LED flashing back & forth.
            RED_LED_R = 1           `Turn OFF Red LED
            GREEN_LED_R = 0         `Turn ON Green LED
            PAUSE 100               `Delay 100ms
            GREEN_LED_R = 1         `Turn OFF Green LED
            RED_LED_R = 0           `Turn ON Red LED
            PAUSE 100               `Delay 100ms
        NEXT flash                  `Increment the index variable, flash
        RED_LED_R = 1               `Turn OFF Red LED (Now all LEDs are OFF)
        OUTPUT_R = 1                `Close right amp Output Relay
        RETURN


        `SAFELY TURNS OFF AMPS WHEN POWER IS SWITCHED OFF
PWR_OFF_SEQUENCE:
        OUTPUT_L = 0                `Immediately disconnect load from each amp output to avoid
        OUTPUT_R = 0                `turn-off transients.
        PAUSE 200                   `Wait 200ms.
        POWER_L = 0                 `Then remove power from the amps by opening their
        POWER_R = 0                 `on-board power relays.
        PowerOn = 0                 `Clear the Power-On flag bit.
        RETURN


JUDGE_TEMP_L:
        `CHECK HEATSINK TEMP OF LEFT AMPLIFIER...
        IF (TempLC100 >= 9000) AND (HotShutdownL = 0) THEN HOT_SHUTDOWN_L
                    `If TempLC reaches 90 deg-C threshold and amp is not in
                    `Shutdown mode, then execute HOT_SHUTDOWN_L subroutine.
        IF (HotShutdownL = 1) AND (Shutdown_PowerReset = 1) AND (TempLC100 < 6500) THEN
            HotShutdownL = 0        `If amp is in Hot Shutdown mode, heatsink temp is
            GOSUB AMP_STARTUP_L     `under 65 deg-C, and the POWER switch has been
        ENDIF                       `cycled, then remove amp from Hot Shutdown mode and
        RETURN                            `begin start-up sequence.


JUDGE_TEMP_R:
        `CHECK HEATSINK TEMP OF RIGHT AMPLIFIER...
        IF (TempRC100 >= 9000) AND (HotShutdownR = 0) THEN HOT_SHUTDOWN_R
```

```
                    `If TempRC reaches 90 deg-C threshold and amp is not in
                    `Shutdown mode, then execute HOT_SHUTDOWN_R subroutine.

IF (HotShutdownR = 1) AND (Shutdown_PowerReset = 1) AND (TempRC100 < 6500) THEN
      HotShutdownR = 0  `If amp is in Hot Shutdown mode, heatsink temp is
      GOSUB AMP_STARTUP_R     `under 65 deg-C, and the POWER switch has been
ENDIF                         `cycled, then remove amp from Hot Shutdown mode and
                              `begin start-up sequence.
RETURN


                    `FORCES LEFT AMP TO ENTER SHUTDOWN MODE
HOT_SHUTDOWN_L:     HotShutdownL = 1  `Set the Hot Shutdown flag bit of left amplifier.
                    OUTPUT_L = 0      `Disconnect the output relay of left amplifier.
                    PAUSE 200         `Wait 200 milliseconds.
                    POWER_L = 0       `Disconnect the Power relay of left amplifier.
                    Shutdown_PowerReset = 0 `This flag bit will be set again (=1) by the Power-on
                                      `sequence after POWER switch is cycled.  The POWER
                    RETURN            `switch must be cycled before any amplifier is
                                      `permitted to exit Hot Shutdown mode.


                    `FORCES RIGHT AMP TO ENTER SHUTDOWN MODE
HOT_SHUTDOWN_R:     HotShutdownR = 1  `Set the Hot Shutdown flag bit of right amplifier.
                    OUTPUT_R = 0      `Disconnect the output relay of right amplifier.
                    PAUSE 200         `Wait 200 milliseconds.
                    POWER_R = 0       `Disconnect the Power relay of right amplifier.
                    Shutdown_PowerReset = 0 `This flag bit will be set again (=1) by the Power-on
                                      `sequence after POWER switch is cycled.  The POWER
                    RETURN            `switch must be cycled before any amplifier is
                                      `permitted to exit Hot Shutdown mode.
```

`LCD: Selecting the LCD Display Mode (0-5)

```
`assume 10ms total program execution time

TRISD.5 = 1                     `Make pin RD5 (PORTD.5) an input
DISPLAY     VAR    PORTD.5      `Rename pin RD5 as "DISPLAY"

DisplayMode       VAR   BYTE    `Create variable to store the display mode number.
DisplayDebounce   VAR   BYTE    `Create variable to debounce DISPLAY button 100ms for after pushed



IF DisplayDebounce > 0 THEN DisplayDebounce = DisplayDebounce - 1
                                `If DisplayDebounce is above zero, subtract one from it.

`Poll the DISPLAY button, determine Display Mode number
IF (DISPLAY = 1) AND (DisplayDebounce = 0) THEN `If DISPLAY button is pushed, and if the 100ms
                                                `debounce delay has expired, then...

    DisplayMode = DisplayMode + 1               `Increment DisplayMode variable.
    IF DisplayMode = 6 THEN DisplayMode = 0     `If Display Mode variable exceeds 5, reset it
    DisplayDebounce = 10                        `restart the 100ms DISPLAY button debounce delay
ENDIF

SELECT CASE DisplayMode
    CASE 0
    CASE 1   <- (Standby)
    CASE 2
    CASE 3
    CASE 4
    CASE 5
END SELECT
```

0 = OFF → display is off
1 = PowerOff → displays only ambient temp and Time?
2 = MAIN → input selection, peak P_{out}, temp of hottest heatsink, Time?
3 = OUTPUT → peak output P, V, I
4 = TEMPERATURES → Ambient, Right heatsink, Left heatsink.
5 = CREDITS → scrolls the names of team members

★ to change the display mode, simply press the display mode button on front panel.

Miscellaneous List of some identifiers

```
`The following Power-On sequence subroutine is executed every time the main POWER switch enters
`the "ON" position.  It checks for protection flags (like Hot Shutdowns) and checks existing
`heatsink temperatures.  If no fault conditions or warning flags are detected, power will then
`be applied to both right and left amps (i.e. their power relays will be closed).  A brief 2 sec
`delay is introduced (with flashing LEDs) while the amplifier turn-on transients decay, then
`their output relays will be closed connecting each amplifier's output to its respective load.
`Next, one of the stereo input relays will be energized (AUX1, AUX2, or AUX3) according to the
`value of the input selection variable stored in the microcontroller RAM from the previous user.
`The final step now is to launch the selected LCD display mode, the number of which has also
`been stored in RAM from the previous user.
```

but this paragraph explains a little about the start-up sequence

Mains

START: Always sent back to START line before repeating one of the MAIN

MAIN_OFF:

MAIN_ON:

Subroutines

TEMP_SAMPLE: Samples temps, calculates 100*C and 100*F values.

JUDGE_TEMPS: Evaluate temps for Hot Shutdown threshold

PWR_ON_SEQUENCE: Turns ON BOTH amps
AMP_STARTUP_L: Turns ON Left amp
AMP_STARTUP_R: Turns ON Right amp

HOT_SHUTDOWN_L: Turns OFF Left amp
HOT_SHUTDOWN_R: Turns OFF Right amp

PWR_DOWN_SEQUENCE:

Delay Variables

TempSampleDelay - BYTE

Flag Variables

PowerOn - Indicates which body loop the program is executing in;
either the (PowerOn=1 loop) or (PowerOn=0 loop)

"low" when POWER=0 (power switch OFF)
Triggers Power-On sequence when POWER goes "high"
"high" indicates that Power-On sequence has occurred

HotShutdownL - set (=0) while left amp is in hot shutdown mode
HotShutdownR - set (=0) while right amp is in hot shutdown mode

Shutdown_PowerReset - set (=1) by the Power-On sequence,
cleared (=0) by a Hot Shutdown event

Mode Variables