TCP/IP Networking: Internet Radio Using OLED Display and MP3 Audio Decoder

Author: Howard Schlunder
Rodger Richey
Microchip Technology Inc.

INTRODUCTION

Internet Radios are defined as a “hardware device that receives and plays audio from Internet Radio stations or a user’s PC”. The audio is streamed to the radio using MPEG-1 Audio Layer 3 (MP3), Windows® Media Audio (WMA) or Advanced Audio Coding (AAC) compressed audio formats. The “radio stations” range anywhere from public AM or FM radio stations that broadcast over the air as well as on the Internet, to University radio stations down to any individual wishing to create their own radio station.

The idea of an Internet Radio is not a new idea. You can buy commercially available Internet Radios, ranging in price from $129 US to $400 US, from companies such as Barix™, Logitech™/Slim Devices, Roku™ Labs and Philips®. Most of these make the connection using wired Ethernet; some have a wireless connection.

The focus of this application note is to show how to create a low-cost Internet Radio that connects to SHOUTcast servers and plays MP3 audio. The hardware uses the PIC18F67J60 microcontroller with integrated 10Base-T MAC and PHY and an external MP3 audio decoder. The software uses the standard Microchip TCP/IP Stack with external serial SRAM buffering to ease the streaming of compressed audio data to the MP3 decoder. Figure 1 shows a picture of the Internet Radio Demonstration Board (DM183033) that is available for purchase on MicrochipDirect or through one of Microchip’s distributors. Figure 2 shows the block diagram for the Internet Radio design used in this application note.

FIGURE 1: INTERNET RADIO DEMONSTRATION BOARD

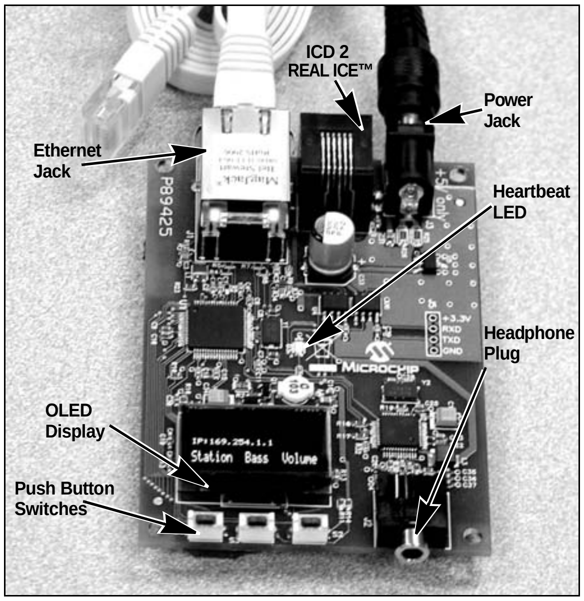
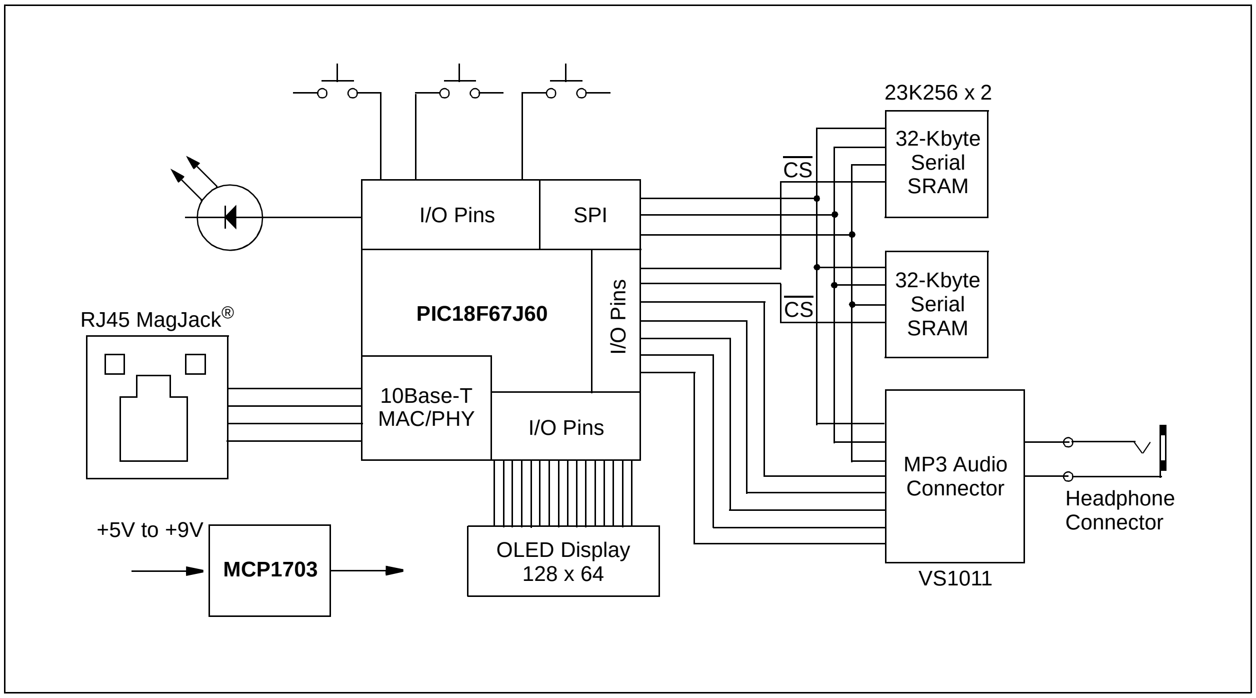


FIGURE 2: INTERNET RADIO BLOCK DIAGRAM



MAKING THE CONNECTION

The heart of this design is the PIC18F67J60 microcontroller. This MCU has an integrated 10Base-T MAC and PHY peripheral in addition to the standard peripheral set of 64-pin PIC18 MCUs. It controls the entire process, from making the connection to the audio server, to streaming the data to the MP3 decoder, to displaying status on the LEDs and OLED display and reading user input on the push button switches.

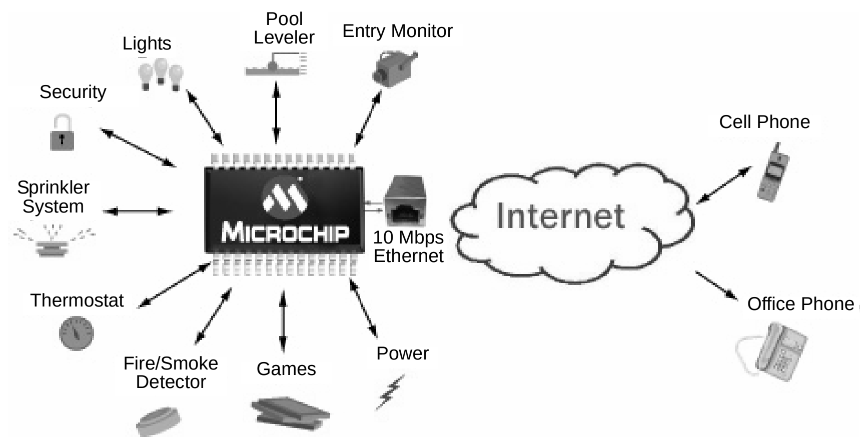
While this particular application does not use many of the resources on the PIC18F67J60, it does have a full complement of peripherals. Table 1 shows the feature set for all PIC18FXXJ60 devices.

TABLE 1: PIC18FXXJ60 MICROCONTROLLER FAMILY FEATURES

Device	Flash Program Memory (bytes)	SRAM Data Memory (bytes)	Ethernet TX/RX Buffer (bytes)	I/O	10-Bit A/D (ch)	CCP/ ECCP	MSSP			EUSART	Comparators	Timers 8/16-Bit	PSP	External Memory Bus
								SPI	Master I^{2}C™					
PIC18F66J60	64K	3808	8192	39	11	2/3	1	Y	Y	1	2	2/3	N	N
PIC18F66J65	96K	3808	8192	39	11	2/3	1	Y	Y	1	2	2/3	N	N
PIC18F67J60	128K	3808	8192	39	11	2/3	1	Y	Y	1	2	2/3	N	N
PIC18F86J60	64K	3808	8192	55	15	2/3	1	Y	Y	2	2	2/3	N	N
PIC18F86J65	96K	3808	8192	55	15	2/3	1	Y	Y	2	2	2/3	N	N
PIC18F87J60	128K	3808	8192	55	15	2/3	1	Y	Y	2	2	2/3	N	N
PIC18F96J60	64K	3808	8192	70	16	2/3	2	Y	Y	2	2	2/3	Y	Y
PIC18F96J65	96K	3808	8192	70	16	2/3	2	Y	Y	2	2	2/3	Y	Y
PIC18F97J60	128K	3808	8192	70	16	2/3	2	Y	Y	2	2	2/3	Y	Y

The second crucial piece of this application is the Microchip TCP/IP Stack. The Stack is what makes the connection to the audio server and then receives the audio stream for deployment to the MP3 decoder. The Stack is a suite of programs that provides services to all TCP/IP-based applications. You don't need to know all of the intimate details of the TCP/IP specifications to use the Stack. Microcontrollers using embedded TCP/IP Stacks can be used to enable a myriad of applications, such as those shown in Figure 3.

FIGURE 3: EMBEDDED ETHERNET ENABLED APPLICATIONS



Based on the TCP/IP reference model, the Stack is divided into multiple layers, where each layer accesses services from one or more layers below it. Per the specifications, many of the TCP/IP layers are “live”, in the sense that they not only act when a service is requested, but also when events like time-outs or new packets arrive. The Stack is modular in design and written in the C programming language. Typical implementations will use 30-60 Kbytes of code, depending on which modules are included, leaving plenty of code space on the PIC18FXXJ60 devices. Table 2 shows many of the supported protocols. The size of each protocol is not listed here in the application note but is available in the help files that come with the TCP/IP Stack.

TABLE 2: MICROCHIP TCP/IP STACK SUPPORTED PROTOCOLS

Application	HTTP	FTP	SMTP	Telnet	DHCP	DNS	SNMP	NetBIOS	Bootloader	Ping
Transport	TCP				UDP					ICMP
Internet and Network Access	IPv4, ARP									
Physical	Ethernet – ENC28J60 or PIC18F97J60 Family									

The Microchip TCP/IP Stack software provides many essential services to implement the Internet Radio, including protocols, such as TCP, UDP, DHCP, DNS, IP and ARP. The Transmission Control Protocol (TCP) transports the main MP3 audio and metadata while providing flow control to prevent the SHOUTcast server from sending more data than can be held in the Internet Radio RAM at any given time.

The User Datagram Protocol (UDP) transports DHCP and DNS packets. The Dynamic Host Configuration Protocol (DHCP) automatically provides the board's IP address, gateway address, subnet mask and other configuration parameters when the Ethernet connection is first established. These configuration parameters tell the board how the network is organized and how to reach the Internet. The Domain Name System (DNS) is used whenever the user changes the current radio station. It converts the radio station's static host name (ex: "scfire-dll-aa02.stream.aol.com") into a potentially dynamic IP address (ex: 149.174.134.200) which the rest of the TCP/IP Stack protocols require.

The Internet Protocol (IP) transports both TCP and UDP packets across the Internet to the correct destination. However, before the IP transport can be used, the Stack uses the Address Resolution Protocol (ARP) to obtain the Ethernet MAC address (ex: 00-04-A3-BE-EF-1E) associated with the local Internet gateway. All of these services work simultaneously, or in tandem, to establish the connection to the radio server and then ensure a robust user listening experience. These protocols all work in the background without requiring any manual user configuration or intervention.

While this application is based on the PIC18FXXJ60 devices, the Stack has been optimized for use on any PIC18, PIC24, dsPIC® or PIC32 device with enough program memory. It includes support for the ENC28J60 Stand-Alone Ethernet Interface Controller for each of these device platforms. The best part is that the Stack is royalty-free and requires a no fee license agreement, restricting use to Microchip microcontrollers and digital signal controllers. The Stack can be downloaded at www.microchip.com/tcpip. The files for the Internet Radio are part of this distribution.

Now that we have the embedded application defined, we need to have the audio source. There are many sources of audio available off the Internet using file formats described previously. For this application, we will focus on the SHOUTcast protocol and servers. SHOUTcast is a freeware audio streaming technology developed by Nullsoft™. SHOUTcast only uses MP3 or AAC audio encoding and an HTTP-like protocol to transfer files from the server to the client. SHOUTcast is available in both server and client forms on Windows 95/98/ME/NT/2000/XP, Macintosh® OS X, FreeBSD™, Linux and Solaris™ at www.shoutcast.com.

To make the connection from the PIC18F67J60 device to the SHOUTcast server, we must send it a message. The following example shows a typical data structure within the Internet Radio code used to establish the connection.

EXAMPLE 1:

```
stations[0].HumanName = "SKY.FM Top Hits, 96K";
stations[0].HostName =
    "scfire-dll-aa02.stream.aol.com";
stations[0].port = 80;
stations[0].Message =
    "GET /stream/1014 HTTP/1.0\r\n"
    "Host: scfire-dll-aa02.stream.aol.com\r\n"
    "Accept: */*\r\n"
    "Icy-MetaData:1\r\n"
    "Connection: close\r\n\r\n";
```

The structure, `stations[ ]`, holds the information for the various radio stations that are preprogrammed into the microcontroller. Metadata refers to information about the data. In the case of a SHOUTcast stream, metadata refers to the song name and artist. If metadata is not enabled, the human readable name of the radio station that is provided in the structure can be displayed. With metadata enabled, the station name can be automatically obtained from the SHOUTcast server during the initial connection phase. In the current application, the `HumanName` string is not used. We use the radio station name provided in the metadata from the SHOUTcast server.

The remote DNS `HostName` is provided, specifying where on the Internet the SHOUTcast server is located. The TCP port through which the connection is made follows. Normally, the port is 80, but can vary depending on the setup of the SHOUTcast server you are connecting to.

The last piece is the `Message` to initiate the connection to the SHOUTcast server. The `GET` command specifies the name of the audio file or stream on the server. The `Host` specifies which target server we want to connect to. In some cases, there are multiple servers running through a single Internet IP address, and in most cases, the `Host` parameter is always identical to the `HostName` field. The `Accept` field indicates that we are interested in receiving any audio data type, not limited to MP3s alone. In addition to MP3s, the Internet Radio can also play uncompressed PCM WAV streams. `Icy-metadata` determines if song metadata should be inserted into the stream. The most common data inserted is artist name and song title. In the current application, we enable metadata and parse the incoming stream. Typically, the SHOUTcast server sends a variable length block of metadata after every 8192 bytes of audio. We must continuously check the location in the stream and extract the metadata. If the metadata was not filtered out of the stream, the audio decoder would play it, resulting in an audio glitch. The `Connection: close` field notifies the server that it should immediately disconnect our TCP client if the server runs out of data to send us. This generally occurs only during the initial connection phase if we provide an invalid `GET` string, or the server is overloaded and cannot handle another radio listener. By immediately disconnecting, we can attempt to automatically reconnect or give up and switch to a different radio station.

The typical response from the SHOUTcast server is shown below. Each of the tags in this response starts with an `Icy` prefix. This is part of the SHOUTcast protocol.

EXAMPLE 2:

```
ICY 200 OK
icy-notice1: <BR>This stream requires <a
href="http://www.winamp.com/">Winamp</a><BR>
icy-notice2: SHOUTcast Distributed Network Audio
Server/SolarisSparc v1.9.93atdn<BR>
icy-name: S K Y . F M - Top Hits Music -
who cares about
the chart order, less rap & more hits!
icy-genre: Pop Top 40 Dance Rock
icy-url: http://www.sky.fm
icy-pub: 1
icy-metaint: 8192
icy-br: 96
icy-irc: #shoutcast
icy-icq: 0
icy-aim: N/A
```

The main information used by the Internet Radio application is the following:

- `icy-name` – radio station name
- `icy-metaint` – the interval at which metadata arrives in the audio stream
- `icy-br` – the bit rate in kbps of the audio stream; the bit rate can also be read out of the MP3 decoder chip

If the radio receives the `icy-name` SHOUTcast response, the MP3 client task will execute a callback function, allowing the main application to save the result. The callback is `NewServerTitleProc(BYTE *strServerTitle)`, where `strServerTitle` is set to the contents of `icy-name`. The string is volatile and must be saved by the main application code if it wishes to continue using the string after returning from the callback function.

The other tags are not meaningful for this application so are discarded automatically by the MP3 client task. Once we receive the server response, the audio stream follows, which we will discuss in the next section.

Two useful pieces of information, usually transferred as metadata in the audio stream, are the song title and author. The MP3 client code checks at every metadata interval (`icy-metaint`), usually every 8192 bytes, for the following text:

- `StreamTitle='<artist name, song name>';`
- `StreamUrl='<url>';`

The callback function, `NewStreamTitleProc(BYTE *strStreamTitle)`, provides the contents of the `StreamTitle` metadata. Here again, the data is volatile and must be saved by the application. Providing this data makes a big difference in customer satisfaction with the Internet Radio because the song title and artist can be displayed.

DECODING THE AUDIO STREAM

The ~4 Kbytes of general purpose RAM inside the PIC18F67J60 are not enough to buffer the incoming audio stream and keep up when experiencing excessive packet loss on the Internet. The TCP transport protocol carrying the MP3 data across the Internet has a variable retransmission delay, which can be 300 ms or longer for packets that get lost. This requires a larger RAM buffer, which would allow the software to have enough audio data buffered which can compensate for these variable latency issues. The result of not enough RAM are clicks and pops in the audio output resulting from missing packets.

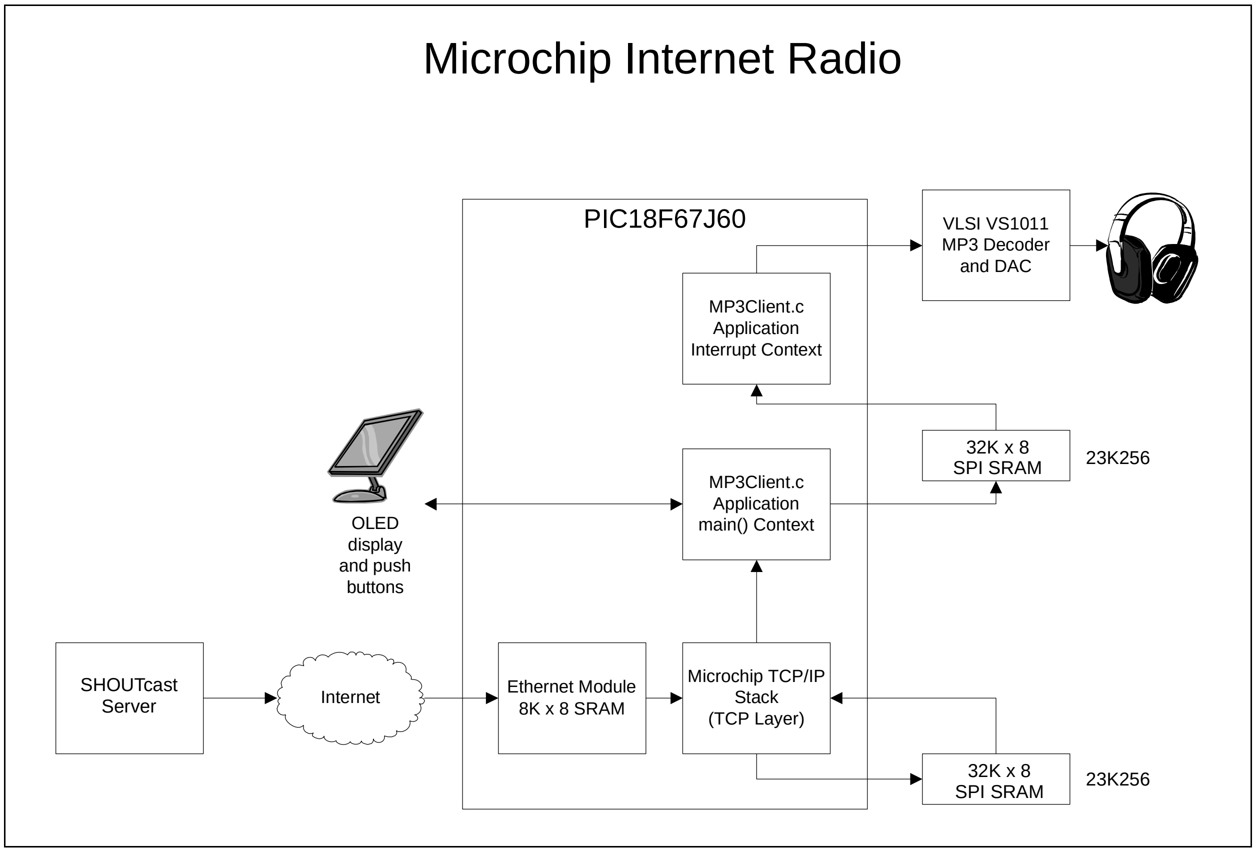
In order to provide more RAM, two external serial SRAMs from Microchip Technology (23K256) provide a total of 64 Kbytes; 32 Kbytes for the TCP layer and 32 Kbytes for the audio buffer. Figure 4 shows a flow diagram for the incoming stream for the server.

The SPI SRAM chips are functionally isolated from each other; however, they are both serving the function of implementing FIFO buffers. In the first chip instance, the 32K x 8 SRAM bolts directly into the TCP module of the TCP/IP Stack. The TCP layer stores all incoming application layer data in this chip. This includes the MP3 stream with the embedded song title and other metadata. All TCP, IP and Ethernet headers are stripped off while being transferred out of the Ethernet module SRAM and are not stored in this external 32K x 8 SRAM. The TCP protocol communicates the amount of free space in this external SRAM chip back to the remote SHOUTcast server, permitting the SHOUTcast server to throttle the data transmission to prevent buffer overflow. Similarly, a large amount of free space communicated to the SHOUTcast server encourages the transmission of more audio data to prevent buffer underflow.

In the `main()` program loop, the `MP3Client.c` application module periodically copies data out of the TCP buffer and into the second 32K x 8 SRAM chip. While being copied, the application strips out the song title and other metadata from the stream and displays it on the OLED. The second external SRAM is written with the raw MP3 stream with no extraneous metadata, allowing minimal processing before being finally copied into the VS1011 audio decoder.

Periodically, during code execution, a timer expires and triggers the `MP3Client.c` application's Interrupt Service Routine (ISR). In the ISR, the MP3 data is copied out of the second external SRAM and written directly to the VS1011 audio decoder. The VS1011 signals to the ISR when more data is required by asserting its DREQ signal output.

FIGURE 4: DATA FLOW DIAGRAM USING EXTERNAL SERIAL SRAM


Microchip Internet Radio
PIC18F67J60
VLSI VS1011 MP3 Decoder and DAC
MP3Client.c Application Interrupt Context
32K x 8 SPI SRAM
23K256
MP3Client.c Application main() Context
OLED display and push buttons
SHOUTcast Server
Internet
Ethernet Module 8K x 8 SRAM
Microchip TCP/IP Stack (TCP Layer)
32K x 8 SPI SRAM
23K256

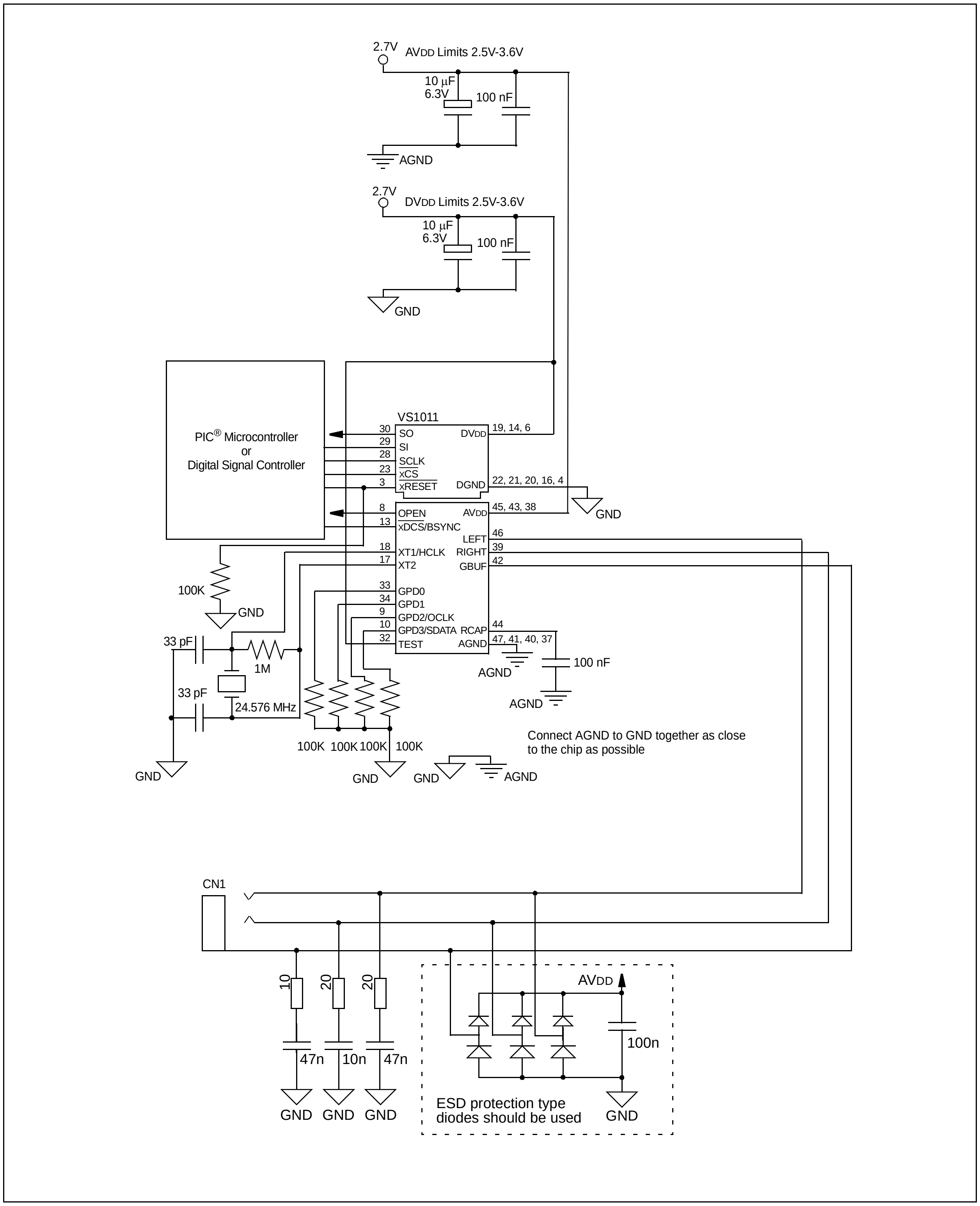
With the audio data buffered, we can stream it to the MP3 decoder. As mentioned before, we need to strip out the metadata; otherwise, the MP3 decoder tries to decode this as compressed audio data which results in blips in the reconstructed audio output. For this application, we selected the VS1011 MPEG Audio Codec from VLSI Solution Oy. This device contains all the necessary components for decoding and playing the audio stream:

- High-performance, low-power processor core
- Decodes MPEG 1.0 and 2.0 Audio Layer III, WAV, PCM and IMA ADPCM
- Up to 320 Kbit/s MP3
- Volume, bass and treble controls
- High-quality stereo DAC
- Stereo earphone driver capable of 30Ω loads
- Separate serial control and data interfaces

Figure 5 shows a generic block diagram of the connection between the PIC18F67J60 and the VS1011. The following signals, with descriptions, are used:

- SO – SPI serial output
- SI – SPI serial input
- SCLK – SPI serial clock
- $\overline{\text{xCS}}$ – SPI chip select for commands
- $\overline{\text{xRESET}}$ – chip Reset
- $\overline{\text{xDCS}}$ – SPI chip select for data
- DREQ – data request

FIGURE 5: MCU TO MP3 DECODER CONNECTIONS



Once the chip is configured, we only need to feed it data when it requests it. The occasional request to change volume, bass or treble is fed to the SPI control interface and does not interfere with data transfer. The software monitors the DREQ line from the VS1011, and while asserted, feeds data to the device. When DREQ goes high, it indicates that the VS1011 is capable of accepting at least 32 bytes of data. If DREQ goes low, the firmware stops sending data.

As mentioned before, the VS1011 has controls for volume, bass and treble. We had to make some trade-offs at this point. Our display was small and we only had three push button switches. Our goal was a simple user interface. We, therefore, only have control of radio station, volume and bass; treble was left out.

Volume is controlled by the SCI_VOL register in the VS1011. It is a 16-bit register, with the upper 8 bits for the left channel and the lower 8 bits for the right channel. A value of 0 represents the highest volume and a value of 254 represents total silence. Each step represents a 0.5 dB increment. On power-up of the application, the volume of both channels is set to 31. The `SetVolume(BYTE vRight, BYTE vLeft)` function is used to modify the volume. It follows the device settings, where 0 is maximum volume and 254 is silence.

Bass is controlled in a similar manner. The SCI_BASS register in the VS1011 contains controls for both bass and treble. Bass control has two settings: bass boost and frequency limit. The boost control value ranges from 0 to 15, with 0 being off and each step representing 1 dB of bass enhancement. The frequency limit also has a 2 to 15 range with each step representing increments of 10 Hz. The `SetBassBoost(BYTE bass, BYTE gfreq)` function is used to set both.

USER INTERFACE

Now that the hardware and software is set up and working, we need to provide status feedback to the user and allow them to control the application. The discrete LED provides a heartbeat from the TCP/IP Stack, indicating that the TCP/IP Stack is operating correctly.

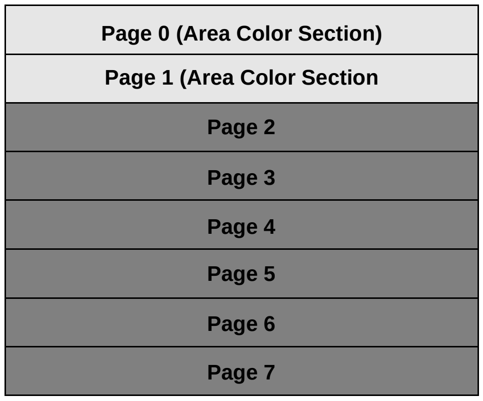
The application provides three push button switches for control. The push button switches are used to navigate the simple menu structure which is displayed on the OLED display. You can change the channel forward or backward, and increase or decrease the bass and volume levels.

The OLED display allows the application to display the status of the Internet Radio. It provides the radio station name, the song title and artist, and also the navigation menu to change the radio station, bass and volume. This display is a monochrome, 128 x 64 display from OSD Displays, part number OSD-2864ASWAG01. OLED displays provide excellent contrast, high brightness, low-power, fast response times, wide viewing angles and several colors. The only two drawbacks to the display are the lifetime and burn-in. This display has a life of 10,000 hours at maximum brightness. The life can be extended by adding an ambient light sensor and dim the display according to the ambient light. The OLED displays can also suffer from burn-in of images displayed. It is therefore recommended that a screen saver is implemented when images are displayed for long periods of time. For the purposes of this application, the navigation menu is blanked after 60 seconds and the radio station name and song title and artist are constantly rotated at 1 character shift per second. This ensures that there is no static image displayed for more than 60 seconds.

The particular display used in this application is available with SPI, I^{2}C™, and both 68 and 80 series parallel interfaces. The only difference between the Serial and Parallel modes is that you can not read from the display in the Serial modes. The Parallel mode is implemented in this application because we wanted to use a put pixel subroutine which requires reading the contents of memory and then modifying one bit of data. The other feature of this display is that it provides a voltage boost driver circuit that can be used with an assortment of external components to create the 9-12V required for the drive circuit.

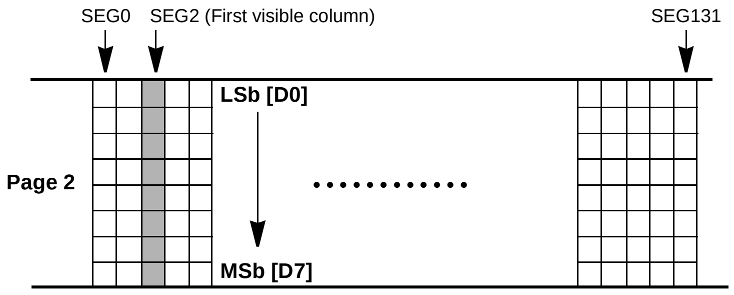
The OLED display uses the SH1101A driver from Sino Wealth. The 132 x 64-bit RAM is organized as 8 pages, 0 through 7, as shown in Figure 6. The OSD Displays supplied OLED display has only 128 columns, and therefore, has 4 extra bytes. Note that the first column that is visible on the display starts at column 2, not 0, as one would expect.

FIGURE 6: GRAPHIC DISPLAY DATA RAM FOR THE OLED DISPLAY



Each page of the driver is arranged as shown in Figure 6. Each byte written to the display is shown vertically. This results in a page being 8 pixels high by 128 pixels wide. As Figure 7 shows, the Least Significant bit is towards the top of the page and the Most Significant bit is towards the bottom of the page. In the example shown, page 2 is selected and the fourth byte in the page is written with 0xFF, which turns all bits in that column on.

FIGURE 7: PAGE 2 EXAMPLE SHOWING STRUCTURE OF RAM vs. WHAT IS DISPLAYED



This application displays an initial welcome screen. The opening graphic is 128 x 64 pixels and the image is stored in the file, `OSDOLED.c`. We use the function, `oledPutImage(rom unsigned char *ptr, unsigned char sizex, unsigned char sizey, unsigned char startx, unsigned char starty)`, to write this image to the display. This function allows a variable image size and placement on the display. We provide a pointer to the array of data, the x and y size of the image, and also the x and y coordinates to start.

The application also provides a font. The font is stored in the file, `OSDOLED.c`, and is only the characters from 0x20 to 0x7E on the ASCII chart. Since most applications don't use the characters outside this range, we remove them from the array to save memory. The font is 5 x 8 with the last line being blank. There are two functions associated with writing characters to the display. Characters can be written as 1 or 2 pages high using the functions, `oledWriteChar1x(char letter, unsigned char page, unsigned char column)` or `oledWriteChar2x(char letter, unsigned char page, unsigned char column)`. Because of the structure of the display, font sizes are limited to page height boundaries. For one-page tall characters, you select the page and starting column. The character is written to the page at the starting column address. The function does not check to see if the starting column value is within the size of the display. For the two-page tall characters, the only difference is that the page states the 1st of two pages to display the characters. This function also performs no checking on location.

In this application, pages 0 and 1 are used to display the song title. Pages 2 and 3 display the URL as obtained through the metadata. These pages are shifted from left to right, to display text longer than the width of the display. Page 4 displays the IP address obtained through DHCP. Pages 5-7 are used for menuing. Page 5 is only used on the submenus to display which submenu the application is currently in. Pages 6 and 7 display the action that is taken by pushing the corresponding push button switch. These pages disappear after 60 seconds as part of the screen saver. Any press of a push button switch will display the menu for input.

To better understand the operation of this display, it is recommended that you obtain copies of the SH1011A data sheet and the OSD Displays OLED data sheet. See the **"References"** section for more details.

DEBUGGING INTERNET CONNECTIONS

When developing applications using TCP/IP and Ethernet, you may find an instance where having a tool to debug an issue is required. When developing the SHOUTcast interface code, we used a tool called Wireshark to figure out what an application sends to a SHOUTcast server to initiate the stream, what the server responds with and how the metadata is embedded in the audio stream. Wireshark allows data to be captured from TCP/IP connections on your PC. You can use Winamp or other player software to make connections to SHOUTcast servers and capture the data streams. The other important piece of information acquired with Wireshark is the IP address and port number for these Internet Radio stations. This is needed to make the connection. Wireshark is a free download available at http://www.wireshark.org.

FUTURE CONSIDERATIONS

This application note was designed to provide an example of the capabilities of 8-bit microcontrollers running TCP/IP Stacks and interfacing to existing Internet infrastructure. There are many improvements that can be made to make the Internet Radio design more desirable and customer friendly.

The first change is more for aesthetics than performance. Many handheld devices are using QVGA graphics LCDs to display information. These displays are quite powerful, ranging in colors from 256 to more than 65,000 colors. They are often packaged with a resistive touch screen that can replace any push button switches. The combination of the two can result in a very powerful user interface to the Internet Radio. Companies such as Ramtex Internation ApS, Segger Microcontroller Systems and Microchip offer graphics LCD libraries that greatly simplify the interface to the display and generation of objects, menus, etc.

The second potential change to the application pertains to applications that require more processing power than the PIC18F67J60 can provide. Microchip offers a wide range of 16-bit microcontrollers that increase the performance up to 40 MIPs. The PIC24F microcontrollers offer 16 MIPs of performance and many peripherals, such as a Parallel Master Port (PMP) to interface to graphics LCDs or the Peripheral Pin Select (PPS) module that allows efficient use of I/O pins. Microchip also offers 32-bit microcontrollers in our PIC32 product line with all the same features of PIC24, with up to a 1.5 DMIPS/MHz core. To add the Ethernet interface capability, the ENC28J60 from Microchip provides the same Ethernet MAC and PHY used in the PIC18F67J60, but in a 28-pin stand-alone package with SPI connection.

Finally, connections to a subscription-based audio streaming web site, such as Slacker™, Pandora, Live365, etc., can be developed to provide unique audio streaming capabilities. These services allow users to define playlists and select artists to tailor the audio stream to the listener.

CONCLUSION

At first, the idea of creating an Internet Radio on anything other than a 32-bit microcontroller may not seem feasible. Once the data rates of streaming audio are considered and application overhead is understood, the processing power of the PIC18F67J60 family of microcontrollers coupled with an integrated 10Base-T MAC/PHY can be used to create the basic platform for these radios. The extra bandwidth can be applied to enhance the user interface with touch screens and graphics LCDs.

Many thanks to Professor Dr. Francesco P. Volpe and Christoph Stein from the Microcomputer Lab at the University of Applied Sciences Aschaffenburg in Germany for their innovative use of PIC microcontrollers in Ethernet applications.

MEMORY USAGE

Total Flash program memory on PIC18F67J60 = 131,072 bytes

Total memory used by Internet Radio = 40 Kbytes

Main routine = 4.7 Kbytes

OLED driver (includes font and introductory image) = 2.7 Kbytes

VS1011 driver = ~1 Kbyte

MP3 client = ~2.5 Kbytes

Intro graphics image = ~1 Kbyte

Font table = ~0.5 Kbytes

Miscellaneous routines = ~2.6 Kbytes

TCP/IP Stack protocols = ~26 Kbytes

Protocols used = TCP, UDP, DHCP, PIC18F97J60 Ethernet driver, ARP, IP, DNS

Total data memory on PIC18F67J60 = 3808 bytes

Total data memory used by Internet Radio = 1820 bytes

Audio and TCP/IP Stack buffering uses 64 Kbytes from the external serial SRAMs.

REFERENCES

SH1011A Data Sheet – www.sinowealth.com

OSD-2864ASWAG01 – www.osddisplays.com

VS1011 Data Sheet and Application Notes – www.vlsi.fi

23K256 Data Sheet – www.microchip.com

SHOUTcast – www.shoutcast.com

Microchip TCP/IP Stack – www.microchip.com/tcpip

PIC18F97J60 Family Data Sheet – www.microchip.com

Original Internet Radio Design

Prof. Dr. Francesco P. Volpe & Christoph Stein, Microcomputer Lab, University of Applied Sciences Aschaffenburg, Wuerzburger Strasse 45, D-63743 Aschaffenburg, Germany; volpe@volpe.de

APPENDIX A: SCHEMATICS

FIGURE A-1: INTERNET RADIO SCHEMATICS (SHEET 1 OF 2)

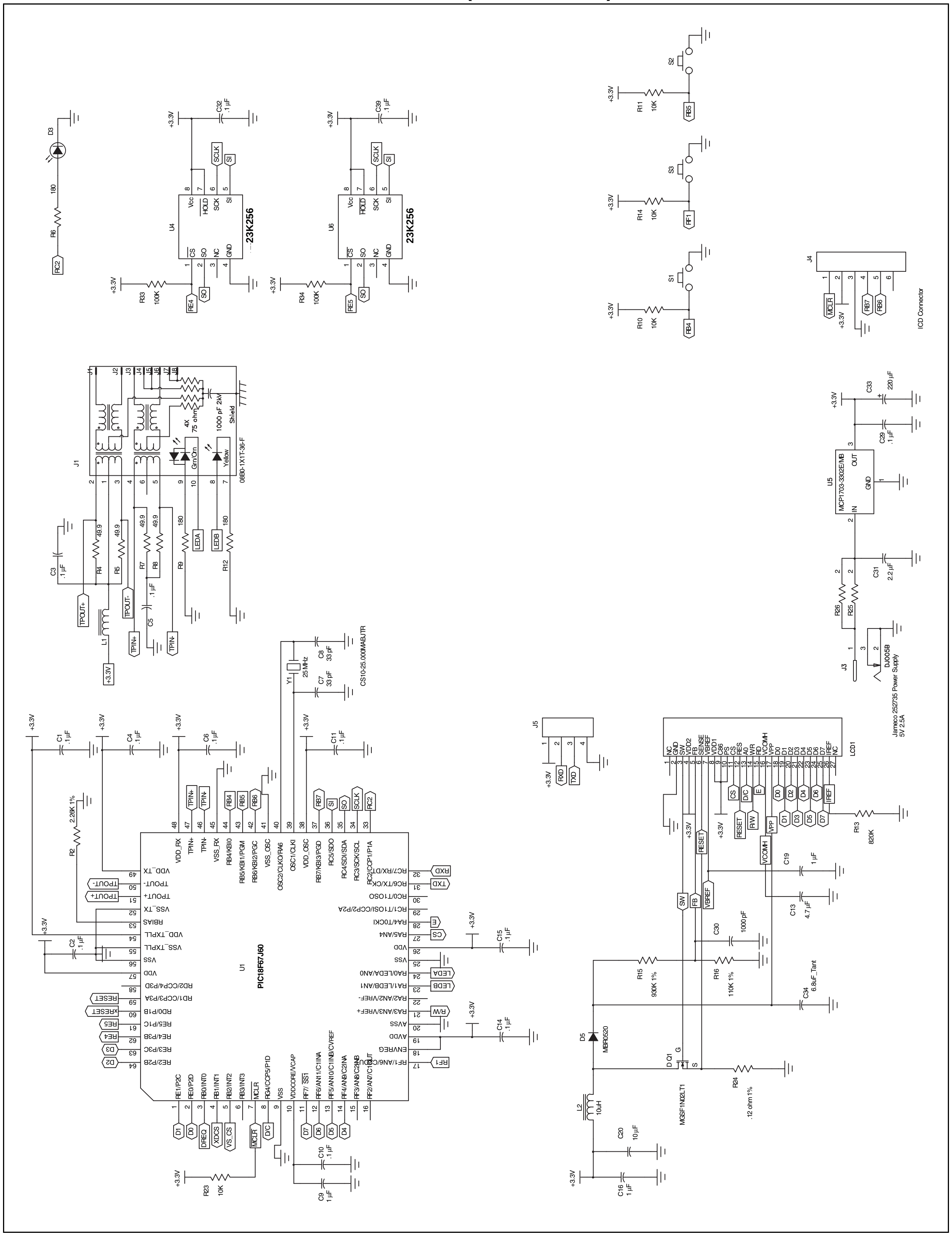
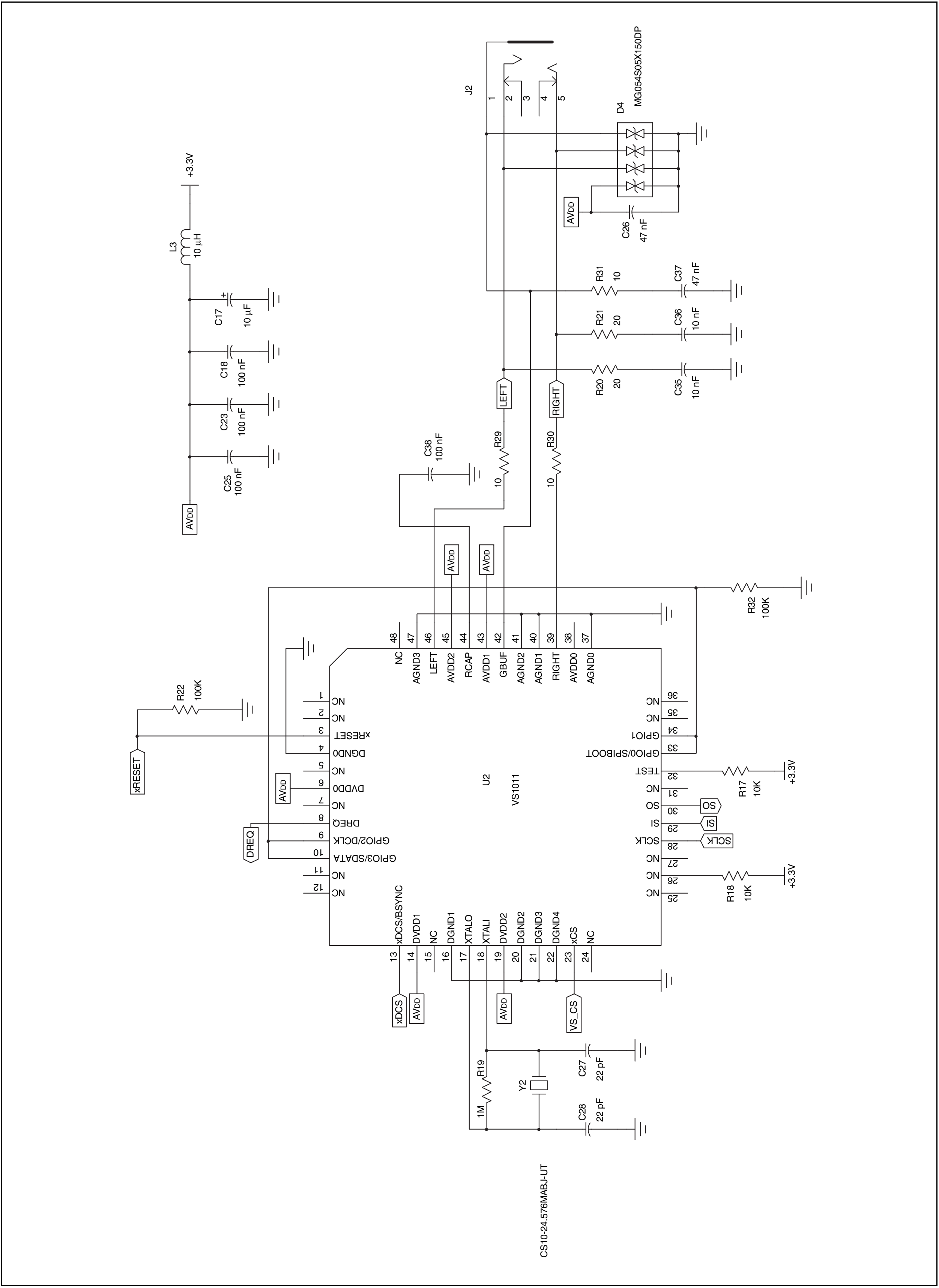


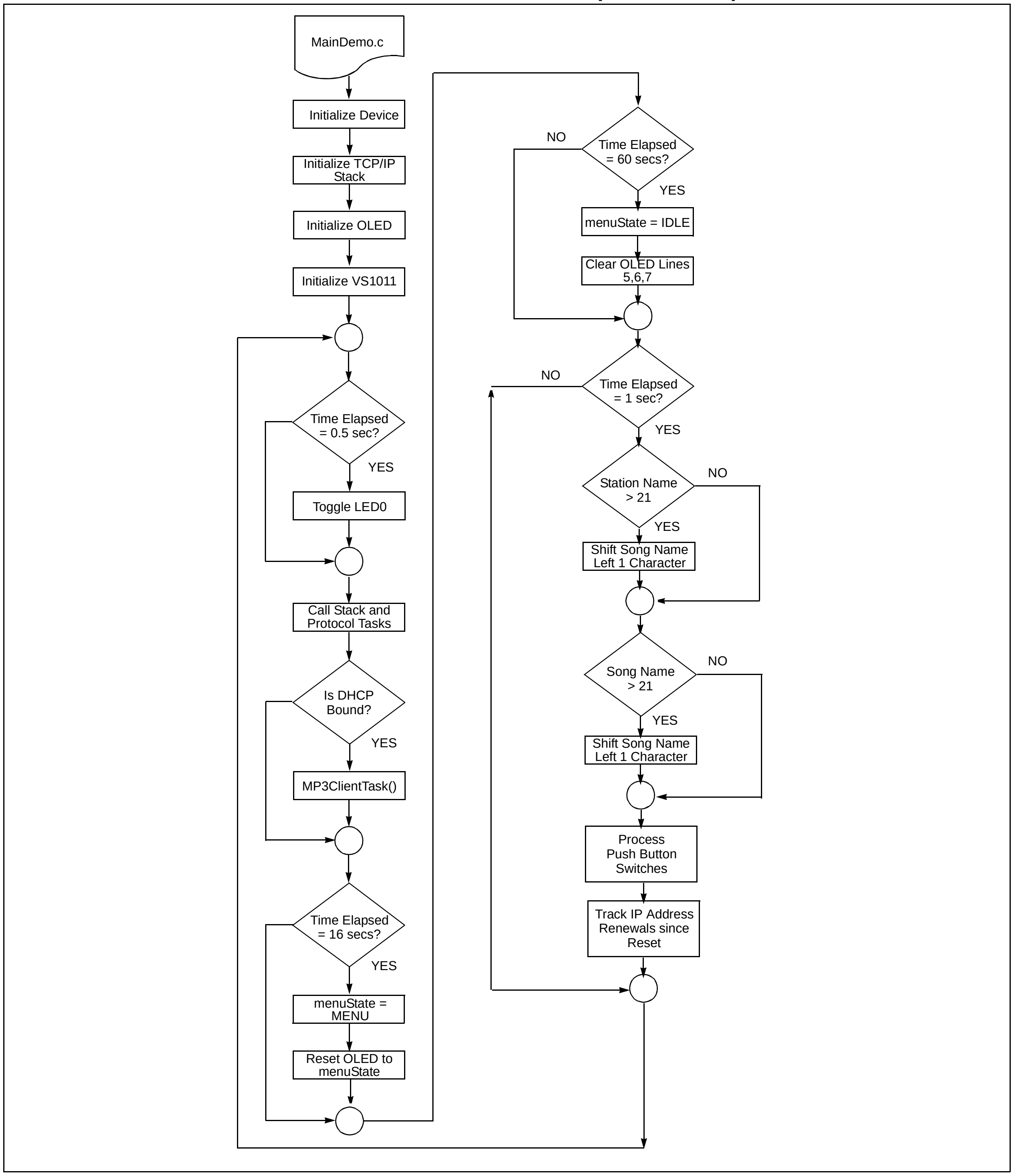
FIGURE A-2: INTERNET RADIO SCHEMATICS (SHEET 2 OF 2)



APPENDIX B: SOFTWARE FLOWCHARTS

Since the TCP/IP Stack is very large, the software flowchart is only for the main routine found in the `MainDemo.c` file. The files are part of the Microchip TCP/IP Stack distribution that can be found at www.microchip.com/tcpip.

FIGURE B-1: MAIN ROUTINE SOFTWARE FLOWCHART (`MainDemo.c`)



NOTES:

MICROCHIP

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
http://support.microchip.com
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Farmington Hills, MI
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
Kokomo, IN
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

Santa Clara
Santa Clara, CA
Tel: 408-961-6444
Fax: 408-961-6445

Toronto
Mississauga, Ontario,
Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon
Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8528-2100
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Hong Kong SAR
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8203-2660
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

ASIA/PACIFIC

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4080

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-2566-1512
Fax: 91-20-2566-1513

Japan - Yokohama
Tel: 81-45-471- 6166
Fax: 81-45-471-6122

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-572-9526
Fax: 886-3-572-6459

Taiwan - Kaohsiung
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan - Taipei
Tel: 886-2-2500-6610
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

UK - Wokingham
Tel: 44-118-921-5869
Fax: 44-118-921-5820

01/16/09