Wireless Music Player

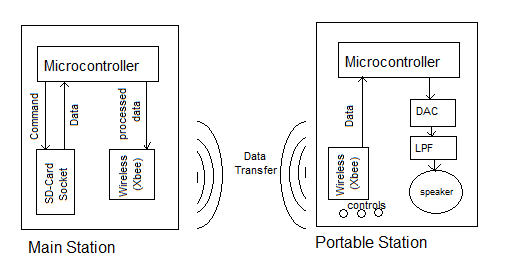
Prepared by :

Fadi Salah Neirat

Mohammad Farid Barham

Introduction :

Our wireless music player allows the user to listen to uncompressed digital audio streamed (.wav files )over a wireless link.

The music player reads uncompressed audio data from an SD card in an immobile "base station." A pair of Xbee (series I)transceiver modules are used to stream data and control signals between the base station and a portable module. The battery powered portable module can be connected to speakers at any location within 30 meters of the base station as shown below.

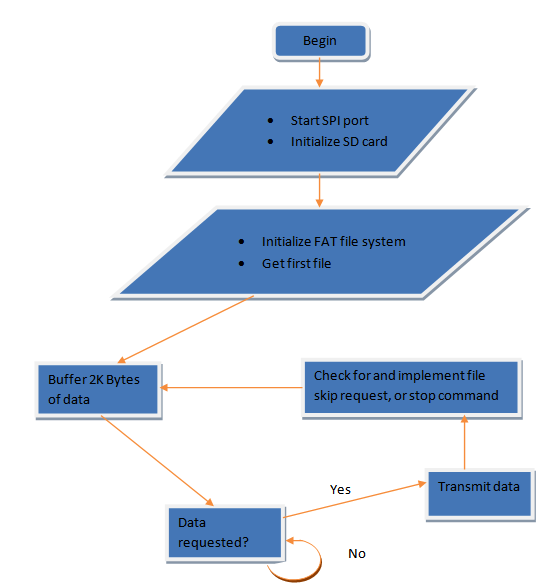
Instruments and components needed:

1. 2 copper boards.
2. PIC18f4620 & PIC18LF4620 (MICROCONTROLLER).
3. 2 XBEE wireless (IEEE 802.15.4 2.4 GHz).
4. SD-CARD.
5. SD-CARD socket.
6. Serial DAC.
7. Low Pass Filter (LPF).
8. Speaker.
9. Regulators (8v -> 5v and 5v->3.3v ).
10. 4 push buttons(To control sound state)

\*. Base Station :

The SD card is connected to the microcontroller via the SPI port. To use the SD card, it must first be initialized. This is done in the MMC\_SD\_Reset function. This function wakes up the SD card by sending 80 clock pulses, then sends the idle command (0x40) to the card. Once the card is in idle mode, the activate command (0x41) is sent to the card till it responds with 0x01. The last thing this function does is to set the sector size to 512. The sequence of events described below is specified in this application note.

After a successful initialization, the card can be read from and written to, We used a FAT library from MICROCHIP provided to us by Dr.Raed Alqadi the diagram will be as follows:



\*. Portable module:

The code on the portable module handles playback of the music data and user input. The data originating from the SD card at the base station needs to be transferred into a buffer, transmitted over serial, transferred into the receive buffer at the portable module and finally sent out over SPI to the DAC. Taking this sequence into account along with the overhead of the 802.15.4 protocol, the maximum attainable data rate is far below the Xbees' theoretical maximum of 250 Kbps. We settled for a sample rate of 8 Khz mono and a resolution of 8 bits for our music. This resulted in a bandwidth requirement of about 64 Kbps.

To ensure a continuous flow of audio data in spite of adverse network conditions, we implemented a ring buffer of 3072 bytes. A data request command (0x01) is sent to the base station if the amount of data in this buffer is less than 512 bytes. Once the command is sent, the code waits till the buffer is about 150 bytes smaller before sending another request.

Upon receiving a data request command, the base station transmits a data stream of 2048 bytes then stops and waits for a new request before sending out more data. RF systems like the Xbees are by nature half duplex; they can use their antenna for transmitting or receiving, but not both simultaneously. Once a data request command is sent to an Xbee, there will be an inevitable delay before it is transmitted, since the Xbee is also receiving an inbound data stream. We use a large ring buffer to decrease the probability of network latency causing buffer underflow and silent patches during music playback. The serial receive function on the portable module's microcontroller is interrupt driven. Data received is immediately added to the ring buffer and the write pointer is incremented.

We created a timer interrupt that fires every 125 milliseconds (8 KHz). Within this interrupt, a single sample is transmitted to the DAC from the ring buffer and the read pointer is incremented. Data is sent to the DAC via the SPI port on PORTB. The DAC we used (TLV 5616) is a 12 bit DAC, and has a 16 bit data packet structure. The first 4 bits form a command that defines speed and operating range while the lower 12 bits are the data. The SPI port is set to run at 10 MHz, to ensure that the microcontroller spends as little time in this interrupt as possible. This reduces the chance of dropped RF packets, since serial reads are handled in the main loop.

Status:

Play/Pause:

-this can be used to pause playback. Pause stops portable module from requesting new data or playing anything over the DAC. The base station does not need to be alerted about this button press as its default behavior is to wait till data is requested.

Next:

-causes the portable module to send a command to the base station to skip to the next song. It also resets the ring buffer pointers to zero to start the new song.

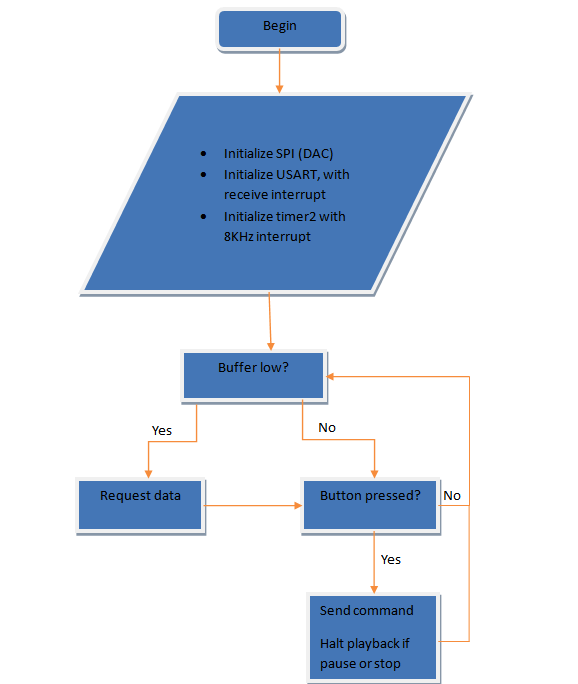
Previous:

-causes the portable module to send a command to the base station to skip to the previous song. It also resets the ring buffer pointers to zero to start the new song.

Stop:

-sends a command to the base station to stop streaming data and to reset to the first song. It also resets the ring buffer. The play button must be pressed in order to restart music playback.

The diagram will be as follows:



The last task is to filter signals to produce sound through using this this serial DAC and filter after it:

