MDFLY's AU5017 Embedded Micro SD/HCSD Card MP3/WAV Player Module

Date 2/27/2016 - Version 1.0

White Paper by Steve Bjork of Haunt Hackers

The scope of this document will cover the basic operations of the AU5017 Embedded player module in regards to it use with microcontrollers like the Picaxe series. The information included in this document is based on MDFLY's preliminary support sheet for the AU5017 module and bench tests ran on samples. This document was not created or supported by MDFLY. No warranty is implied by MDFLY, Haunt Hackers or Steve Bjork on the information contained within.

This new Embedded module has many improvements over earlier units. Some are:

- Supports both older Micro SD Cards and higher capacity Micro SDHC cards.
- Plays compressed MP-3 and loss-less wave format files.
- Bidirectional commutation between the player and controller via TTL serial lines.
- Simple one byte commands with short one or two byte returns.
- Busy (playing) output line can be programmed for both Active High or Active Low logic.
- True Analog Ground for better sound quality.
- Compatible with 3.3 and 5 Volt power source and logic.
- Single sided board for lower cost.
- Excesses 1200 tracks arranged in 6 groups of 200 tracks each.
- Uses filenames (Not file position) to select the track to play.
- Small Size of 40mm x 33mm (1.57" x 1.3")

Commutation Interface

While the AU5017 could be used with simple push button interface, we will be cover the more powerful use case of a microcontroller driving it. Three I/O lines (RX, TX and Busy) are used to commutate with the microcontroller. Default speed and format is 9,600 Baud with 8 bits of data, no parity and one stop bit. When using a Picaxe controller it is recommended to use the Hardware Serial option of the chip. The Busy line is only need to check the status of current track is done playing.

The baud rate can be set to eight levels from 4,800 to 115,200. It is not recommended to change the baud rate since it is not reset back to the default speed on power down. (The AU5017 can be restored to its default setting by shorting RST pad ground for one second.)

File System

All files are stored in the root directory. Each file has a 4 digit name with the extension of MP3 or WAV. The first number (of the filename) is the index (group) a file belongs. Ranging from 1-6, it's a handy for grouping files by type. The last three digits are used to select the file to play. The file 3034.MP3 is in the third index group and track number 34.

Commands $F1 to $F6 are used to select the index of 1 to 6. The commands ranging from 1 to 200 are used to play the corresponding file on the SD/SDHC card. Once the track starts playing (See Play Track Command), the Busy line will go active till it's done. (The Busy line is only active while a track is play.)

Requesting Number of Tracks

The $E3 command returns one byte that contains the number of tracks in the current index group. If the SD Card has seven tracks of the current index, then the number returned is $07.

Volume Control

There is no command to directly set the volume level of the output. But there are two commands to raise or lower the volume level by one. Both commands do return a byte that holds the new volume level of 1-30. With knowing the level of the volume, it's a simple process to raise or lower the volume till it has reaches the desired level.

Busy Line

The Busy line can operate in two modes, Active High or Active Low. Active High is when the busy line is High when playing a track and returns to low once done. The Active Low is the reverse status of the line. Since there are no commands to "Set" of the status of Busy line, the Change Busy Status command of $FB will need to be used. This command does return the new Active mode of Busy line and can be used to toggle the Busy Active State till it's in the desired mode.

The Busy line does not swing to the full level of a logical 1 on the output. For the Busy line to work reliable with 5 Volt logic, a 10K pullup resistor is needed. Tie the resistor between +5 Volt power supply and the Busy line.

Play Track Command

Sending the track number (1-200) will start the module play that track of the current index group. The Busy line will go active and the new play Track commands will be accepted. Once the track is done playing, the Busy line will return to an inactive state and new Play Track commands will be accepted. The Stop command ($FA) can be used to stop the current track that is playing.

Play Track Return Codes

When a track is selected to play by sending code 1-200 there are two possible sets of return codes. If the track file is found on the SD/HCSD Card the player will return two bytes of data. The first byte is the current index (1-6) and second byte is the track number (1-200). Should the track file not be found then just one byte is returned with the value of $FF for an error code. If a play track command is sent while a track is currently playing, then an error code of $FF is also returned.

Track Time

The commands $EC (Total Track Time) and $ED (Current Track Time) return the time in Minutes and Seconds. Of the two byte returned, the first byte is the Minutes and the Second by is the Seconds. The Current Track Time command returns the current time into the track that is playing.

The Busy Line is not the only way to inquire if a track is done playing. The fact of the Current Track Time stops counting up once the track is done playing can also be used as a test for end of track too.

Source Code Samples

The following code is for a Picaxe 40X2 running at 32 MHz (8 MHz resonator). A pause variable is entered at 4 times the standard values since the chip is running 4 times than normal speed. The hsersetup B9600_32,0 command is used once on the Picaxe's initiation code to setup the Hardware Serial I/O port.

This first bit of code is used to send a command like Player Stop that has no return code. There is a 1.5 millisecond delay so the serial hardware can compete sending the command at 9,600 Baud.

```
SoundCommandNoRead:
      hserout 0,(Temp_MSB)     ;send out sound command via 9600 baud hardware serial port
      pause 6                  ;pause for 1.5 ms delay for serial byte to get there
      return                   ;to get there and be processed by the MP-3 Player and exit
```

The next section sends a command and waits for a one byte feedback from the player module. Should the return byte not happen within 100 milliseconds then the code will slowly flash the red status LED. (This shows the player module is missing or not working.)

```
;SoundCommandRead1 - Send out Temp_MSB to sound player and waits for one byte return in
;                    Temp_MSB. There is a 100 ms timeout error to flash the red LED
;                    if no player.
SoundCommandRead1:
hserout 0,(Temp_MSB)          ;Send command to audio module
hserin 400,ErrorFlashSlow,0,1 ;Flash Red LED slow if time out (No MP-3 Player)
Get 0,Temp_MSB                ;Get first byte in buffer
Return                        ;and exit if no error

;SoundCommandRead2 - Send out Temp_MSB to sound player and waits for two byte return in
;                    Temp_Word. There is a 125 ms timeout error to flash the red LED
;                    if no player.
```

This section sends a command and waits for a two byte feedback from the player module. Should the return byte not happen within 125 milliseconds then the code will slowly flash the red status LED. (This show the player module is missing or not working.)

```
SoundCommandRead2:
hserout 0,(Temp_MSB)          ;Send command to audio module
hserin 500,ErrorFlashSlow,0,2 ;Flash Red LED slow if time out (No MP-3 Player)
Get 0,Temp_MSB                ;Get first byte in buffer
Get 1,Temp_LSB                ;Get second byte in buffer
Return                        ;and exit
```

This code will start a track and verify it did play. By using the SoundCommandRead1 subroutine with a read of just one byte, it will get the Index number (1-6) of the playing track or an error code of $FF if the track did not play.

```
SoundPlayTrackWithVerify:
gosub SoundCommandRead1       ;Tell sound player to play a track, put output in Temp_MSB
if Temp_MSB=PlayerErrorCode then ErrorFlashFast ;Error if track not found
return                        ;else exit
```

To set the Volume to a given level, a Volume UP or Volume Down commands need to be sent to read what the current volume level is. The following code will set the volume of the player to the level in Temp_LSB.

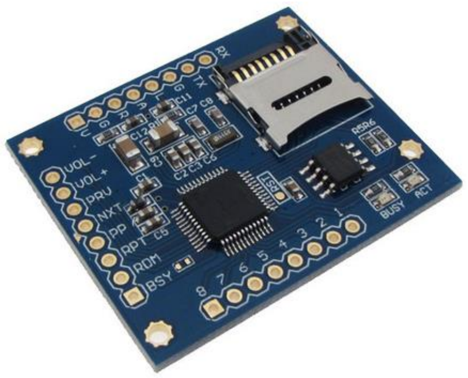
```
SetPlayerVolume:
Temp_MSB=PlayerVolumeUp             ;get command that does Volume up

SetPlayerVolumeCMD:
gosub SoundCommandRead1             ;Set volume up or down command
if Temp_LSB>Temp_MSB then SetPlayerVolume ;too low? Then set it higher
if Temp_LSB<Temp_MSB then
      Temp_MSB=PlayerVolumeDown     ;Get volume down command
      goto SetPlayerVolumeCMD       ;Tell player to run it.
endif
return                              ;and exit
```

As you will see, the code to just fade out the playing volume is very simple. It also forces the track to stop playing and waits 10 milliseconds for it take full effect.

```
FadeOutVolume:
Temp_MSB=PlayerVolumeDown     ;get command to fade down the Volume
gosub SoundCommandRead1       ;tell player to do it.
pause 200                     ;wait 50 ms before moving on
if Temp_MSB>1 then FadeOutVolume    ;if not down to lowest volume then loop again

Temp_MSB=PlayerStopPlay       ;force a Player Module stop the current track
gosub SoundCommandNoRead      ;send sound command and pause for 1.5 ms
pause 40                      ;wait another 10 ms
return                        ;and exit
```



First 8 Pin Connection Set		
V	3.3 - 5 Volt	Positive voltage. Never over 5 Volts.
G	Ground	Negative or ground of power supply.
R	Right Channel	Right Channel of audio output.
A	Audio Ground	Audio Ground for Left and Right Channels.
L	Left Channel	Left Channel of audio output.
G	Ground	A second ground connection for the power supply
TX	Transmit	TTL serial output line.
RX	Receive	TTL serial input line.

Second 8 Pin Connection Set		
BSY	Busy	Busy Output Line. (Needs a 10K pullup resistor to work with 5 Volt logic.)
The other 7 pins connections are not used by the Microcontroller		

Third 8 Pin Connection Set
All 8 pins connections are not used by the Microcontroller

Mounting on a PC board

All three 8 pin set should have a row of 8 pin solder to bottom of the AU5017 board for secure mounting. If a “stackable” header is used to receive the 8 pin header, there will be clearance to install other components under the AU5017 Module. This will save about 2 square inches of board space by stacking it.