```
// bring in this include file if you are setting up a serial port on
digital pins
#include <SoftwareSerial.h>

// SOMO Serial pins
int Rx_pin = 40;
int Tx_pin = 38;

// create the object for the virtual serial port. In this case, it's
called Somo
SoftwareSerial Somo(Rx_pin,Tx_pin);

// define a structure for holding the bytes that will be passed to the
SOMO
typedef struct _ControlMsg
{
  byte  start;
  byte  cmd;
  byte  feedback;
  byte  para1;
  byte  para2;
  byte  checksum1;
  byte  checksum2;
  byte  end;
} ControlMsg;


// I call a procedure in the setup function that initializes the SOMO
void setup(void)
{
  Initialize_SOMO();

}

// In this simple example, in the main loop function, I am playing
tracking 5 everything 2 seconds.
void loop(void)
{
        Play_Track(5)   // This plays track 5 in folder 01 on the
micro SD Card
        delay(2000)     // This gives the SOMO time to actually play
the track before playing it again
}

// here is the function for initializing the SOMO
void Initialize_SOMO()
{

  // initialize the serial port to the SOMO MP3 module
```

```
  Somo.begin(9600);
  delay(500);

  // Set the volume to max
  Somo.write(0x7E);
  Somo.write(0x06);
  Somo.write((byte)0x00);
  Somo.write((byte)0x00);
  Somo.write(0x1E);
  Somo.write(0xFF);
  Somo.write(0xDC);
  Somo.write(0xEF);

  delay(100);

}

// This is the function that actually plays tracks
void Play_Track(byte track)
{

  ControlMsg ctrlMsg;   // Create structure

  // set variables for playing a track
  byte cmd = 0x0F;     // sets the "Play a specified folder & track"
command
  byte feedback = 0;   // sets the feedback requirement to off
  byte para1 = 1;      // sets it to folder "01" on the microSD card
  byte para2 = track;  // sets the track number  (1 to 255)

  // populate structure with the bytes

  ctrlMsg.start = 0x7E;    //  always use this starter byte

  ctrlMsg.cmd = cmd;
  ctrlMsg.feedback = feedback;
  ctrlMsg.para1 = para1;
  ctrlMsg.para2 = para2;

  // Sending instructions to the SOMO requires the use of a checksum
  // Checksum (2 bytes) = 0xFFFF – (CMD + Feedback + Para1 + Para2) +
1
  word chksum = 0xFFFF - ((word)cmd + (word)feedback + (word)para1 +
(word)para2) + 1;
  ctrlMsg.checksum1 = (byte)(chksum >> 8); // upper 8 bits
  ctrlMsg.checksum2 = (byte)chksum; // lower 8 bits

  ctrlMsg.end = 0xEF;     // always use this end byte

  // Now we write the structure to the SOMO MP3 Module
```

```
  Somo.write((const byte*)&ctrlMsg, sizeof(ctrlMsg));
  Somo.flush();

}
```