Beat707.com

Drum Machine Hacks and Mods

05

MIDI Input

MIDI Input

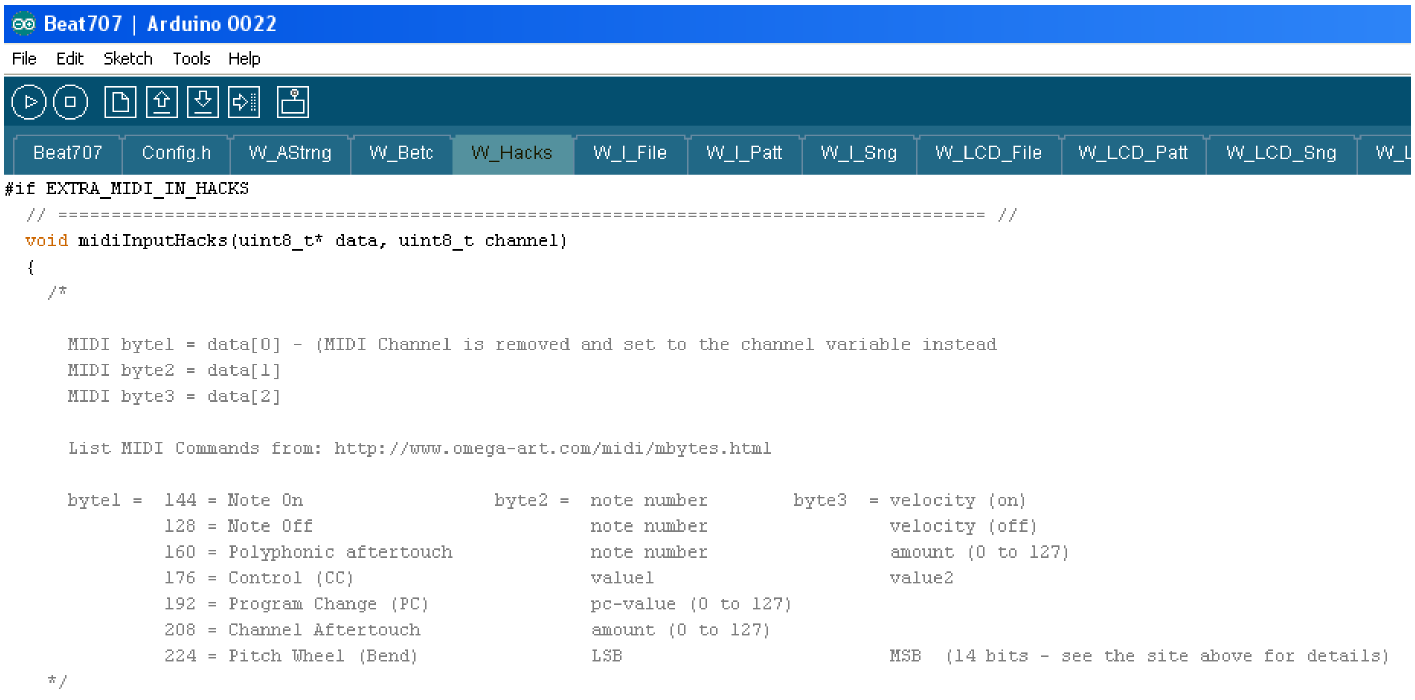
If you wish to use your own MIDI Controller / Keyboard to do some pretty crazy things with Beat707, you can, thanks to a new option we have added to the Config.h Tab: EXTRA_MIDI_IN_HACKS.

When set, the code will call midiInputHacks() in the W_Hacks Tab for any new Midi Input Data. By default we have included the following codes:

- Program Change to Pattern Selection
- Modulation Wheel to BPM Tempo (CC #1)
- CC #2 to Number of Steps
- Drums/S1/S2-Tracks KeyZone Split
- Pitch Wheel (Bend) to Sequence Stop/Play

But you can always erase everything and create your own code. All you need to know is how MIDI messages are passed via the data variable to the midiInputHacks() function. Also, if you change anything in the data variable, it will be passed to the MIDIECHO_BYTRACK portion of the code. Or you could just disable MIDIECHO and handle your own MIDI Output data by calling sendMidiData().

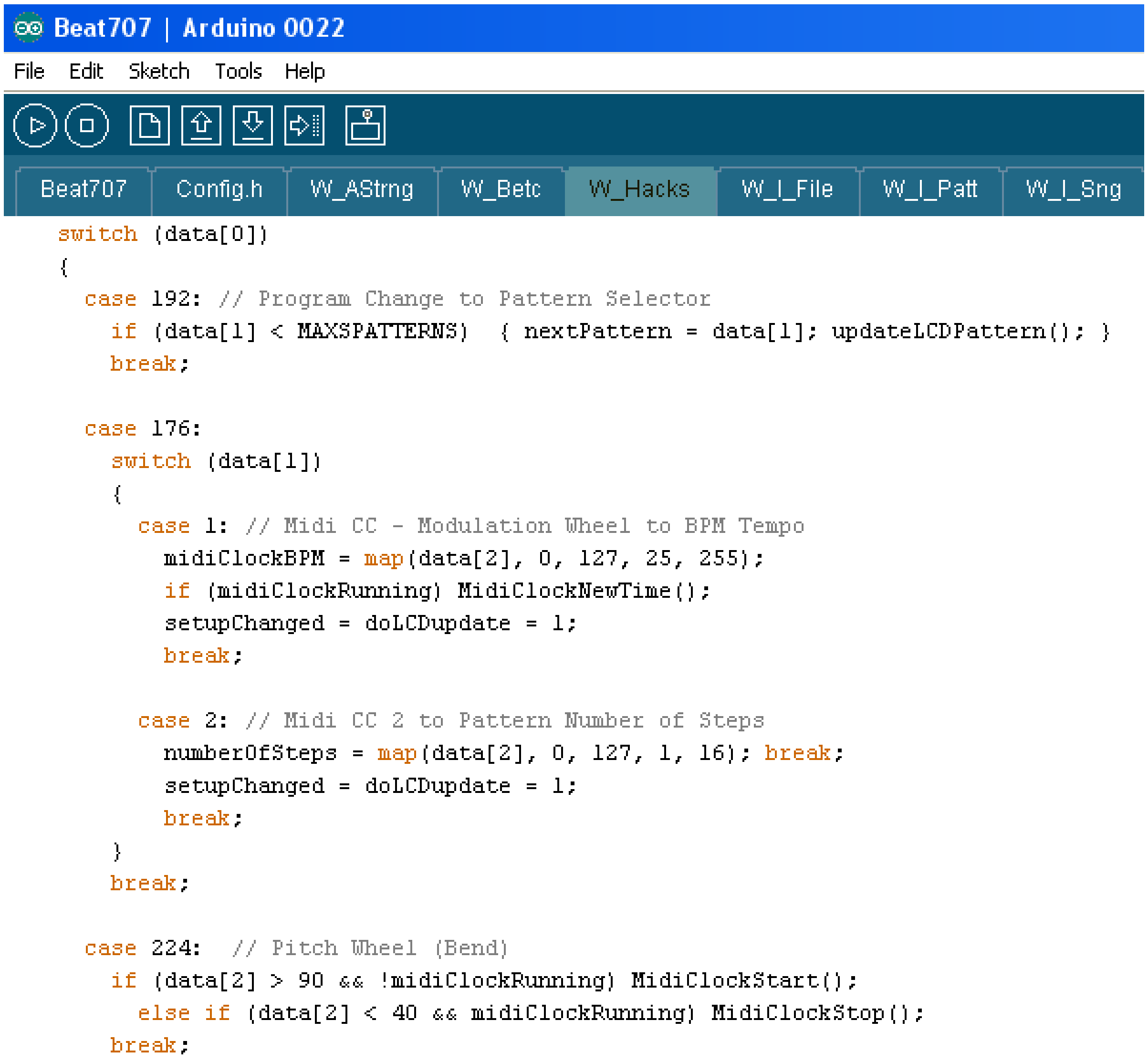
Here's the MIDI reference table included in the source-files:


Beat707 | Arduino 0022
File Edit Sketch Tools Help
Beat707 Config.h W_AStrng W_Betc W_Hacks W_I_File W_I_Patt W_I_Sng W_LCD_File W_LCD_Patt W_LCD_Sng W_L

```
#if EXTRA_MIDI_IN_HACKS
  // ======================================================================================= //
  void midiInputHacks(uint8_t* data, uint8_t channel)
  {
    /*

      MIDI byte1 = data[0] - (MIDI Channel is removed and set to the channel variable instead
      MIDI byte2 = data[1]
      MIDI byte3 = data[2]

      List MIDI Commands from: http://www.omega-art.com/midi/mbytes.html

      byte1 =  144 = Note On                 byte2 =  note number        byte3  = velocity (on)
               128 = Note Off                         note number                 velocity (off)
               160 = Polyphonic aftertouch            note number                 amount (0 to 127)
               176 = Control (CC)                     value1                      value2
               192 = Program Change (PC)              pc-value (0 to 127)
               208 = Channel Aftertouch               amount (0 to 127)
               224 = Pitch Wheel (Bend)               LSB                         MSB  (14 bits - see the site above for details)
    */
```

First we check the MIDI Byte1 code, to know what is that data been input. On the example below, the first check is for MIDI Program-Changes: 192. In this case, Byte2 will hold the change value, which we them check if its not higher than the number of existing patterns.

You could also use MIDI Input to drive external electronic devices, by using any of the free pins: Analog A0, Digital 2 and Digital 3. (D14/D2/D3 on the Beat707 Headers)



```
switch (data[0])
{
  case 192: // Program Change to Pattern Selector
    if (data[1] < MAXSPATTERNS)  { nextPattern = data[1]; updateLCDPattern(); }
    break;

  case 176:
    switch (data[1])
    {
      case 1: // Midi CC - Modulation Wheel to BPM Tempo
        midiClockBPM = map(data[2], 0, 127, 25, 255);
        if (midiClockRunning) MidiClockNewTime();
        setupChanged = doLCDupdate = 1;
        break;

      case 2: // Midi CC 2 to Pattern Number of Steps
        numberOfSteps = map(data[2], 0, 127, 1, 16); break;
        setupChanged = doLCDupdate = 1;
        break;
    }
    break;

  case 224:  // Pitch Wheel (Bend)
    if (data[2] > 90 && !midiClockRunning) MidiClockStart();
      else if (data[2] < 40 && midiClockRunning) MidiClockStop();
    break;
```