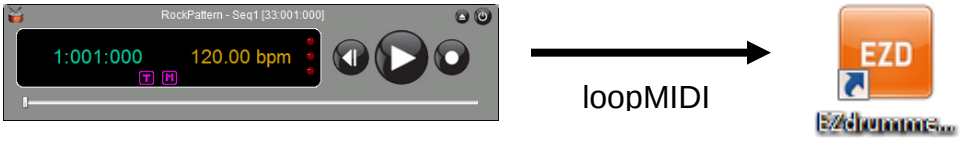
Using a Drum Map for EZ Drummer 2

In Tutorial #1 we configured DrumGen and EZ Drummer 2 to communicate using the virtual MIDI cable *loopMIDI*.



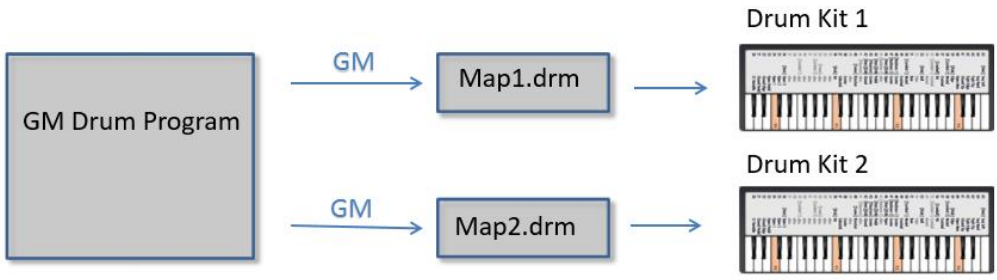
In Tutorial #2 we created a General MIDI drum track for a jazz song and played it back on EZ Drummer 2's *Jazz Basic* drum kit.

In Tutorial #3, we will access more of EZ Drummer 2's features by **using a drum map file created specifically for EZ Drummer 2.**

The 'Brush Delight' Drum Map

As mentioned in Tutorial #2, a *drum map* allows a MIDI drum pattern programmed using a certain note organization - e.g. the General MIDI (GM) standard note definitions - to be played back on a drum kit that might have a different note organization. Indeed, *the most useful drum map for any drum kit is one that maps GM drum note values to the equivalent drum sound within the kit.* For this tutorial we will use the term "drum kit" for what is more properly a software (or "virtual") MIDI drum bank.

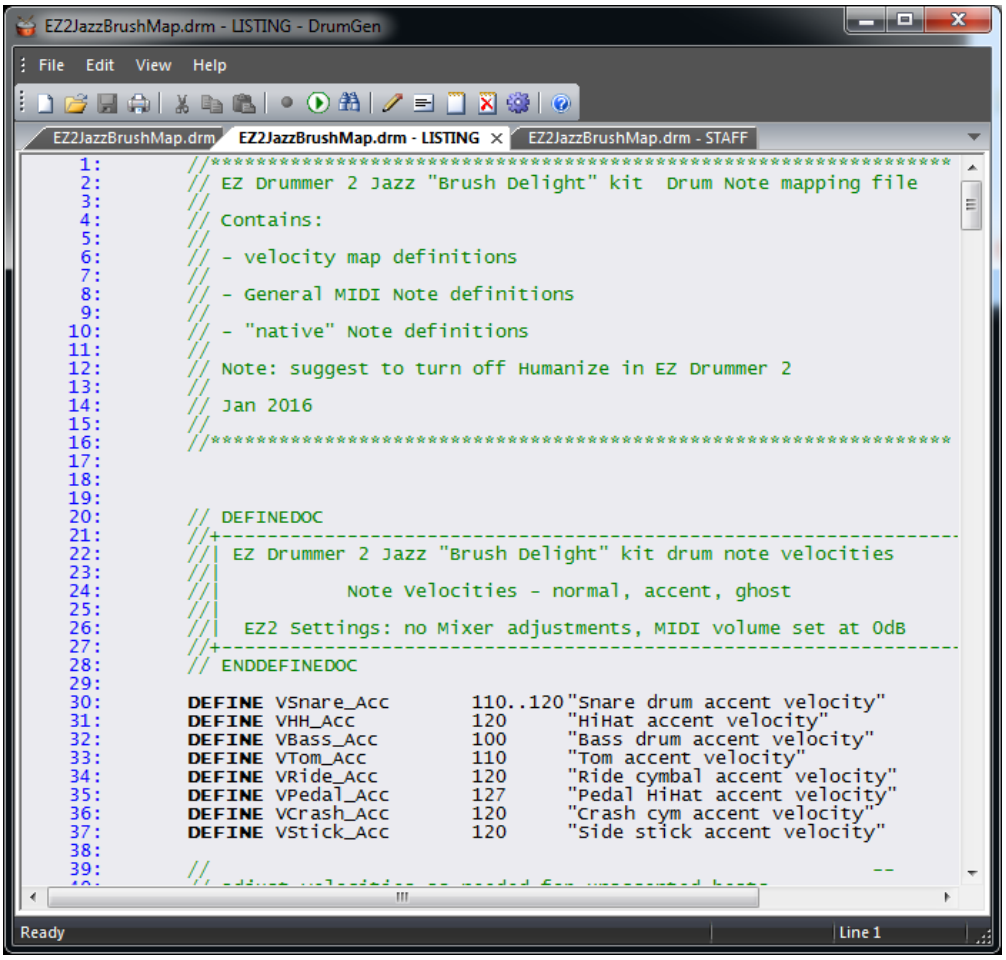
Unsurprisingly, most modern drum software products are GM-compatible by default, and in some cases extended with unique and proprietary extras that show off the power of the product. For DrumGen, drum maps typically consist of definitions within *include files*. These include files can be incorporated into any drum program intended to play back on the associated drum kit. Drum programs written, for example, using GM notes can thereby be targeted to any drum kit by using the corresponding map file ... *that's the beauty of drum maps*.



Let's load EZ Drummer 2 with a particular jazz kit called *Brush Delight*.

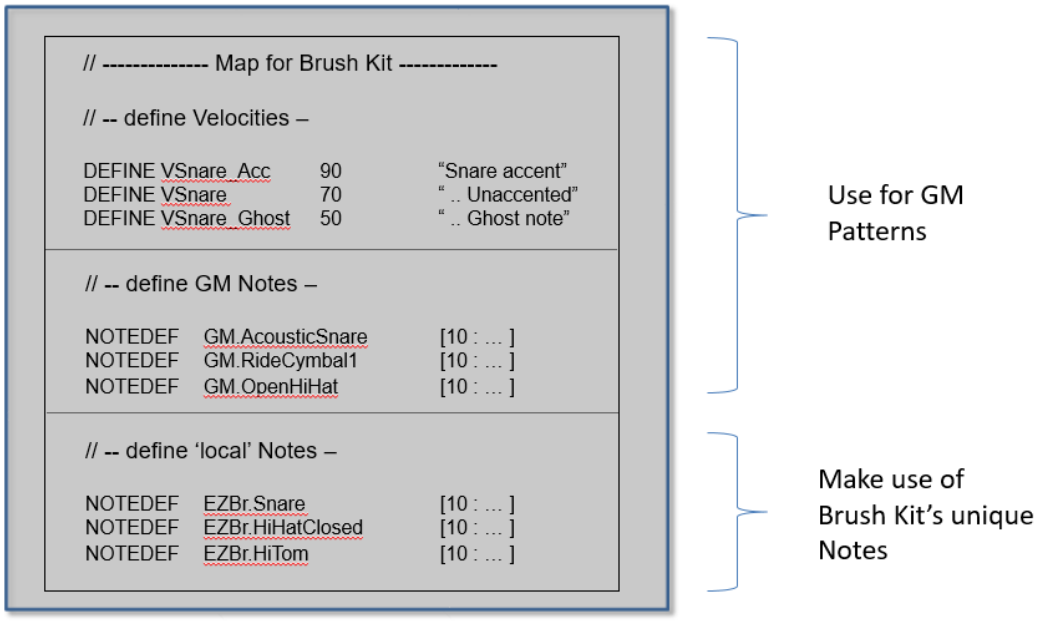
There is a map file on *www.drumgen.com* for this kit under the *Download* page; it's called `EZ2JazzBrushMap.drm`. Download it now.

For now, let's read this file directly into DrumGen, e.g. by double-clicking it. Now we can compile it and take a look at the Note definitions in the Inspector[1].

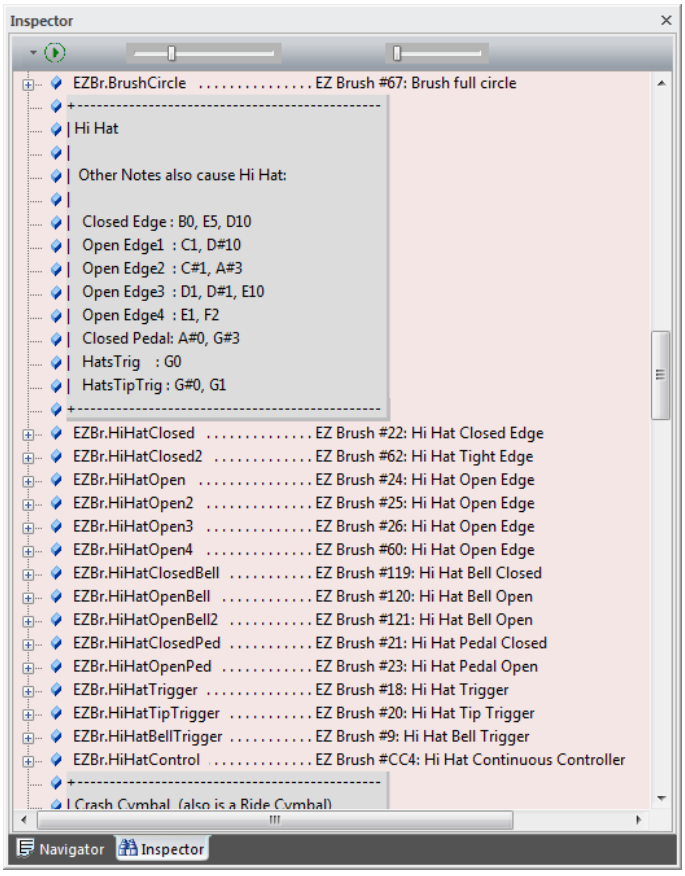


At a high level, the organization of this drum map is shown below. There are definitions that are intended to be used as note velocity values; these form a *velocity map* (all drum kits have different velocity curves). Following that are GM note definitions, and finally there are definitions of notes within the Brush Delight drum kit that fall outside the GM specification – 'local' notes.

[1] In general, this file is intended to be an *include file*; that is how we will use it later in the tutorial.



Let's take a look at Brush Delight's *local* Notes within the Inspector. The definitions are organized by drum sound – Snare, Hi Hat, and so on. In particular, let's create a Hi Hat "straight-8" Pattern in several ways.



The Inspector shows that Brush Delight has many local Hi Hat sounds that will allow us to add variety to our Patterns. Just for comparison, let's create a simple General MIDI version of the Pattern:

```
// Define Open HiHat (OHH) and Closed HiHat (CHH) notes

NOTEDEF OHH                    [GM.OpenHiHat]    VHH
NOTEDEF CHH "HiHat"            [GM.ClosedHiHat]  VHH


PATTERN Straight8_GM {

  {   OHH   8     CHH   8     }  *32

  DESCRIPTION     "8 bars of 8ths - General MIDI"

}
```

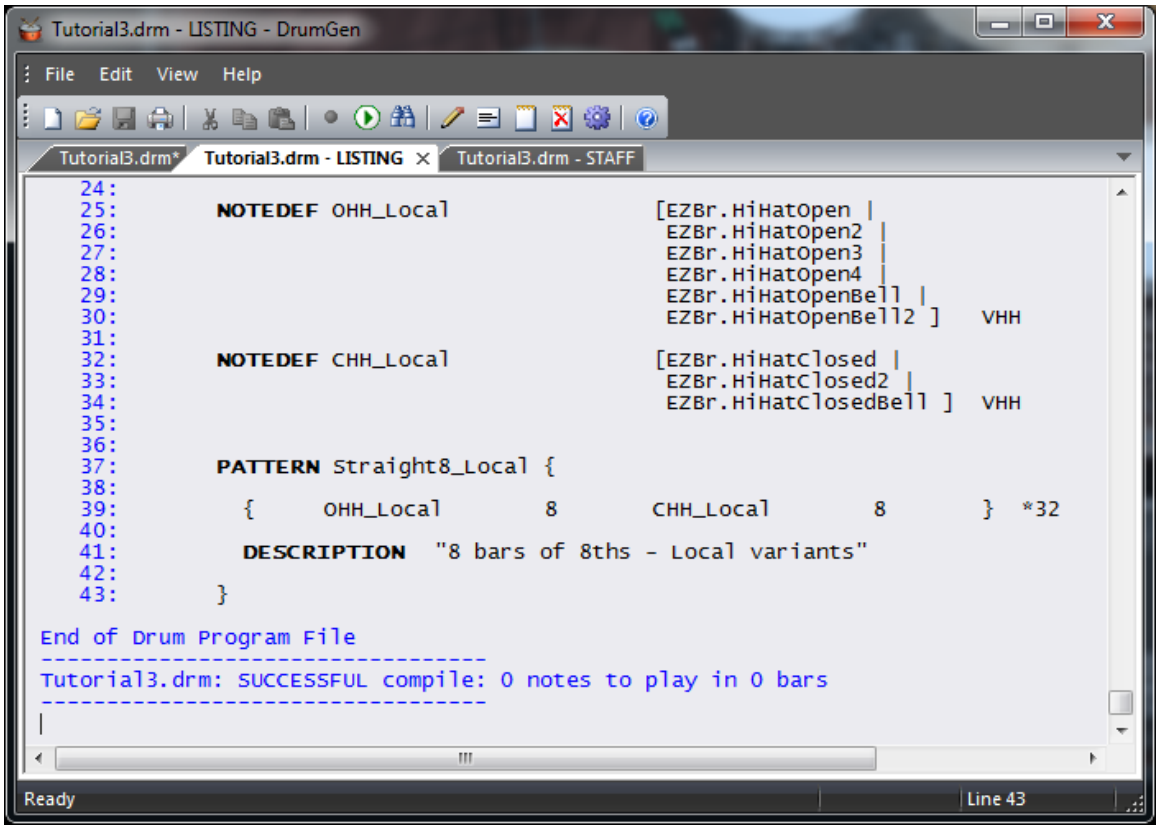
The Pattern `Straight8_GM` above, which we will soon audition in the Inspector, is 8 bars long - 64 eighth notes.

In addition, let's create a more varied Pattern using the local Note definitions:

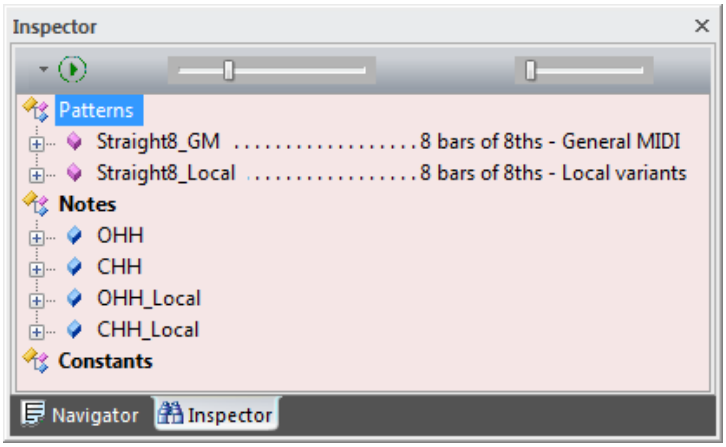
```
NOTEDEF OHH_Local       [EZBr.HiHatOpen |
                         EZBr.HiHatOpen2 |
                         EZBr.HiHatOpen3 |
                         EZBr.HiHatOpen4 |
                         EZBr.HiHatOpenBell |
                         EZBr.HiHatOpenBell2 ]  VHH

NOTEDEF CHH_Local       [EZBr.HiHatClosed |
                         EZBr.HiHatClosed2 |
                         EZBr.HiHatClosedBell ] VHH


PATTERN Straight8_Local {

  {   OHH_Local   8     CHH_Local   8     }  *32

  DESCRIPTION     "8 bars of 8ths - Local variants"

}
```

This Pattern, `Straight8_Local`, looks complicated but it is very similar to Pattern `Straight8_GM`, i.e. 64 eighth notes. The difference is the use of the 'or' ('|') operator. Each time a Note is added into the Pattern - either `OHH_Local` or `CHH_Local` - *it is randomly selected from the Note choices specified in the NOTEDEF*.

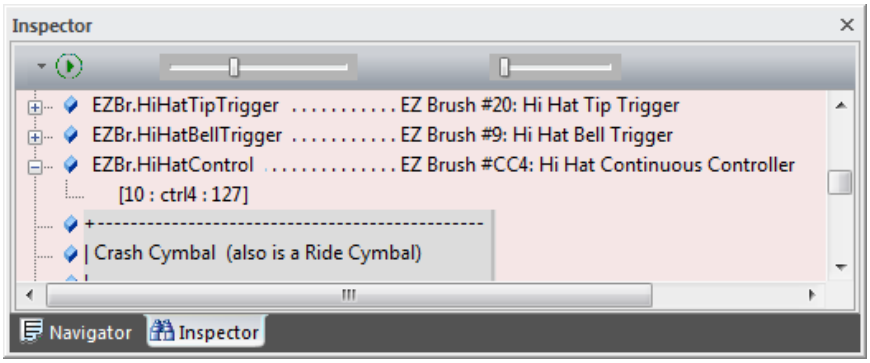
When we compile our drum program - let's call it `Tutorial3.drm` - we can compare these two Patterns in the Inspector. [The entire drum program is included at the end of this tutorial.]



By clicking on the Pattern line items in the Inspector, we can easily hear that `Straight8_Local` provides much more variability than `Straight8_GM`. (Perhaps a bit too much!)



We will illustrate one more feature - the use of a MIDI controller within EZ Drummer 2. Notice that the map file `EZ2JazzBrushMap.drm` contains a definition `EZBr.HiHatControl` shown below. This symbol represents a Controller, not a Note.



Continuous Controller #4 (CC#4) is a MIDI message sent by various electronic drum kits to indicate the position of their Hi Hat foot pedal; these drum kits are used by drummers to drive EZ Drummer 2 and other drum software. EZ Drummer 2 responds to CC#4 messages by using a more 'open' sound when playing back the closed Hi Hat, depending on the CC#4 value most recently received.

DrumGen can simulate the regular depressing of the Hi Hat pedal (with the drummer's left foot) by sending a 'ramp' of CC#4 messages. To illustrate this effect, we will start with an 8-bar Pattern of closed Hi Hat 8ths. To keep it interesting, we will give this base Pattern a recurrent 2-bar *crescendo*:

```
PATTERN Straight8_CHH {

        { CHH 8  CRESC  70..127 * 16} *4

        DESCRIPTION "8 bars of 8ths - closed Hi Hat"
}
```

The subpattern above indicates 16 closed Hi Hat 8th notes, slowly growing in volume (`CRESC 70..127`). The 2-bar subpattern is repeated four times.

We will now define a variant of this Pattern in which, while we are sending these closed Hi Hat notes to EZ Drummer 2, we will also send a series of CC#4 MIDI messages, as defined in Pattern `HH_Pedal_8`:

```
DEFINE      RampDown    127..0      "depress pedal"

PATTERN HH_Pedal_8 {
      EZBr.HiHatControl 1 TIE 1       RampDown  *4

      DESCRIPTION "slowly depress pedal every 2 bars"
}
```

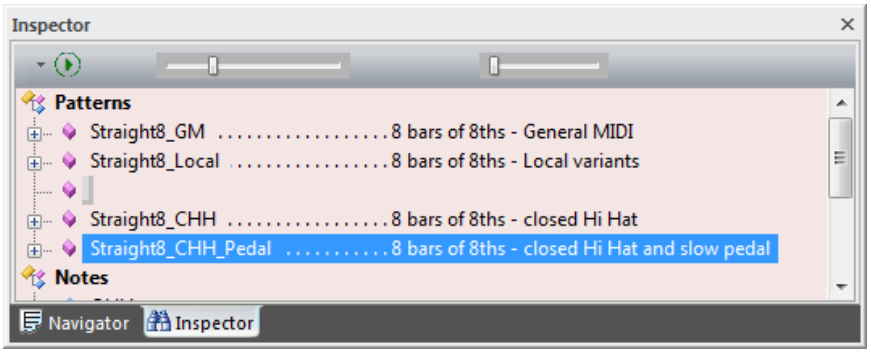
To play these two Patterns together, we can use the '+' operator:

```
[ /Straight8_CHH  + /HH_Pedal_8 ]
```

The full Pattern incorporating this construct is:

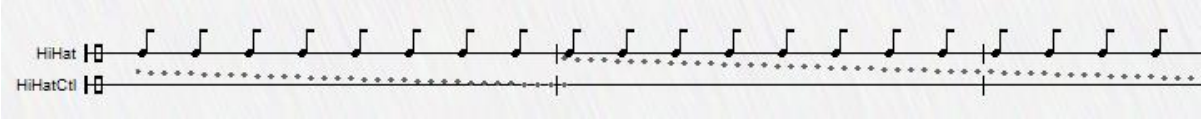
```
PATTERN Straight8_CHH_Pedal {

  [ /Straight8_CHH  + /HH_Pedal_8 ]

  DESCRIPTION     "8 bars of 8ths - closed Hi Hat and slow pedal"
}
```

By clicking in the Inspector we can hear how the addition of CC#4 messages changes the resultant sounds between `Straight8_CHH` and `Straight8_CHH_Pedal`.



We have illustrated the use of the drum map file `EZ2JazzBrushMap.drm` without actually creating a drum track - we have only a Definition Section within `Tutorial3.drm` and not a Play Section. Let's add a trivial Play Section ... but not for the purpose of creating a drum track. Compiling the following Play Section simply allows us to visualize the CC#4 messages being sent to EZ Drummer 2 along with Hi Hat notes.

```
PLAY Straight8_CHH_Pedal

STAFF "ENVELOPES"
```



That's all for now.

Following is the final `Tutorial3.drm` (**bolding** added; feel free to cut and paste):

```
// ***********************
//  Tutorial #3
// ***********************

NOSHOW
  INCLUDEDIR      "D:\DrumGen Tutorials"
  INCLUDE         "EZ2JazzBrushMap.drm"
SHOW

// Define Open HiHat (OHH) and Closed HiHat (CHH) notes

NOTEDEF OHH                    [GM.OpenHiHat]    VHH
NOTEDEF CHH "HiHat"            [GM.ClosedHiHat]  VHH


PATTERN Straight8_GM {

  {   OHH   8     CHH   8     }  *32

  DESCRIPTION     "8 bars of 8ths - General MIDI"

}


NOTEDEF OHH_Local       [EZBr.HiHatOpen |
                         EZBr.HiHatOpen2 |
                         EZBr.HiHatOpen3 |
                         EZBr.HiHatOpen4 |
                         EZBr.HiHatOpenBell |
                         EZBr.HiHatOpenBell2 ]  VHH

NOTEDEF CHH_Local       [EZBr.HiHatClosed |
                         EZBr.HiHatClosed2 |
                         EZBr.HiHatClosedBell ] VHH


PATTERN Straight8_Local {

  {   OHH_Local   8     CHH_Local   8     }  *32

  DESCRIPTION     "8 bars of 8ths - Local variants"

}

DEFINE      RampDown    127..0      "depress pedal"

NOSHOW
  PATTERN HH_Pedal_8 {

      EZBr.HiHatControl 1 TIE 1       RampDown  *4

      DESCRIPTION "slowly depress pedal every 2 bars"
  }
SHOW
```

```
// PATTERNDOC
//
PATTERN Straight8_CHH {
      { CHH 8  CRESC  70..127 * 16} *4
      DESCRIPTION "8 bars of 8ths - closed Hi Hat"
}

PATTERN Straight8_CHH_Pedal {

  [ /Straight8_CHH  + /HH_Pedal_8 ]

  DESCRIPTION     "8 bars of 8ths - closed Hi Hat and slow pedal"
}

PLAY Straight8_CHH_Pedal

STAFF "ENVELOPES"
```