Modular Patch Notation (MPN) Explained

Rev: 030926

Overview. The modular patch notation (MPN) introduced here provides a method to describe analog modular synthesizer patches. In MPN., a patch is composed of element structures that describe control elements and their values (e.g. knobs, switches, joysticks), signal flow through connection elements (input and output jacks, adaptors, interfaces), and a hierarchy of systems-modules-circuits (a domain hierarchy) to uniquely locate specific elements.

Since large amounts of knobs and jacks are typical of modular systems, MPN introduces shorthand mechanisms borrowed from programming theory, such as blocks and dotting, to reduce the written complexity of a patch, yielding a surprisingly compact description.

MPN also allows for writing flexibility. This to address the population that prefers a compact description over a long one, vs. those who prefer a well-indented, cascading representation using space as a visual aid for rapid interpretation of a patch specification.

Domains. Domains are used to qualify the exact location of a control or connection element. A domain is specified in *dot notation* by concatenating names separated by ".". A common domain hierarchy would be

<studio>.<system>.<panel>.<module>.<circuit>

A domain hierarchy

SkyStudio.Serge.Panel3.DualMixer.Mix2

uniquely identifies elements found "in studio *SkyStudio*, in *Panel3* of the *Serge* system, in the circuit *Mix2* in module *DualMixer*". If Mix2 had a control for input 1 called INPUT1, such control could then be uniquely identified by

SkyStudio.Serge.Panel3.DualMixer.Mix2.INPUT1

Element names and domain hierarchies are customized (extended or reduced) to reflect your studio. In practice, most patches are created within single systems, so there is rarely a need to use <studio> and <system> in a domain hierarchy.

Element Structures. These are structures used to specify control values and signal flow. Controls are specified by listing the control names and their control values

Control [control1=value1, ..., controlN=valueN**]**

Unmarked knob values are specified in standard hour notation (e.g. 5, 7:30, 12). Switch values are specified using their physical switch values. In general, for any control, values are specified using their specific units of measure (e.g. sliders using numbers, joysticks using x/y coordinates, etc.).

Signal flow is specified by listing jack output and jack input pairs joined by the flow operator "->'"

Connect [output1->input1, ..., outputN->inputN**]**

Domain dot notation can be used to qualify **Control**, **Connect** and any control, input or output name. For example,

Wog1.**Control**[LFO RATE=5]

reads "control LFO RATE in circuit Wog1 should be set to a 5pm position"; an equivalent notation would be

Control[Wog1.LFO RATE=5]

The duality of this example illustrates the power of domain dot notation: if the entire control element list is in the same locality, then qualify **Control**. But single elements can be qualified as well. In practice, it is easiest to qualify the element structure rather than doing it for each control name in the list. By the way, the same rational applies to connection structures. The same is true for connect structures

Wog1.**Connect**[STEP CV -> RATE]

which reads "a patch cord is connected from output *MIX* in circuit *Wog1* to input *RATE* in circuit *Wog1*".

Element sets can be created by surrounding the set within parenthesis. Element sets are useful to assign all members the same value, input or output. A good example would be

Control[(RATE, LFO)=12]
Connect[OUT1 -> (RATE, IN2)]

Patches. Element structures are grouped under a structure called a patch. Joining the above examples

Wog1.**Control**[LFO RATE=5]
Wog1.**Connect**[STEP CV -> RATE]

describes a patch. From both structures it is clear that LFO RATE, MIX, and RATE are located in the same domain (circuit)Wog1. With a *shortcut*, the example can be notated as

Top Wog1
Control[LFO RATE=5]
Connect[STEP CV ->RATE]

The keyword **Top** is used to specify the domain where all elements in a patch are located. If an element is not qualified, it can be found within **Top**. In general, to determine the correct qualification of an element, follow these precedence order:

1. Element name qualification
2. Element structure qualification
3. **Top**

Assuming two different Woggle Bug modules (WB1, WB2)

Top SkyStudio
WiardSystem.**Connect**
    [WB1.Wog1.OUT1 -> WB2.Wog1.CLOCK,
    WB2.Wog1.OUT1 -> +MON]

indicates that *OUT1* and *CLOCK* are found in *WB2*, both *WB1* and *WB2* are found in *WiardSystem*, and that *WiardSystem* is found in *SkyStudio*. This example also shows how the notation can be spatially distributed to increase readability.

Patches can be named and used as blocks for building more complex patches through the keyword **Patch**. All patch names are specified in caps with a predecessor "+" in the name. The notation assumes the existence of a primitive patch +MON where +MON is a monitoring, amplification or speaker system that enables sound to be heard. One definition could be

> **Patch** +MON
> **Top** GenelecMonitors
> **Control**[VOLUME=10]

Building Complex Patches. If the patch is specified to interface with others, the keywords **IN,** used by a patch to describe incoming signals from external patches, and **OUT**, for outgoing signals, are used to specify an external interface. So the above patch can be modified to

> **Patch** +MON
> **Top** GenelecMonitors
> **IN** LEFT, RIGHT
> **Control**[VOLUME=10]

to indicate that any incoming signal connected to *+MON* should be routed to inputs *LEFT, RIGHT* in *GenelecMonitors*. No **OUT** is required in this case. Dot notation can be used to qualify the signal flow, so that outputs are routed to specific inputs in other patches.

> WoggleBug.**Connect**
> [OUT1-> +MON.LEFT, OUT2-> +MON.RIGHT]

A patch can also be constructed as a modification of an existing one via the keyword **Initial**. For example

> **Patch** +MODIFIED HELLO WORLD
> **Initial** +HELLO WORLD
> **Control**[LFO RATE=12]

specifies that +MODIFIED HELLO WORLD starts from the values and connections of +HELLO WOLD, with a modification to LFO RATE from a value of 5pm to a value of 12 noon.

Other. More useful shortcuts to reduce the amount of writing:

- All elements do not need to be specified. It is common in modulars to have controls, inputs and outputs that are irrelevant to a patch. In this case, omit all superfluous elements.

- Comments can be added by using "- -" or by using the keyword **Comment**.

- Other common blocks used to speed up notation:

  - KBD.CV – Keyboard Control Voltage (pitch) output
  - KBD.GATE – Keyboard Gate (trigger) output
  - +ENVELOPE – external envelope signal output
  - +OSCILLATOR – external sound source output
  - +LFO – external LFO output
  - JOYSTICK.X – x output of a joystick
  - JOYSTICK.Y – y output of a joystick

Conclusion. MPN is capable of specifying very complex patches in very complex situations. MPN eases the task of describing analog modular programming techniques among interested parties. Manufacturers, novices and experienced programmers can benefit alike.

In practice, only a fraction of a signal flow needs to be specified, so patches will tend to be relatively small.

Thanks to Grant Richter for his interest and valuable comments. Improvements? Email me at

bill@axonhillock.com

Enjoy and share your modular knowledge! Feel free to modify MPN as you see fit. Peace be with you.

Bill Sequeira