Overview of the GigPerformer MCU Extension

Many control surfaces utilize the Mackie Control Unit (MCU) protocol. This MCU extension is designed to communicate with such units. Popular currently made examples include the Behringer X-Touch, Icon Platform and Qcon, PreSonus FaderPorts, and SSL UF8.

Some MCU-compatible units offer displays similar to the original MCU, others do not. Either way, OSC can also be used to communicate substantial additional information about control mappings to a display device near your control surface.

Control Organization

The MCU extension works with control surface items in groups by type: knobs, buttons, and faders.

The extension is built around the concept of "banks" of each control group. You can have multiple knob banks, fader banks, etc.

The extension can "bank switch" through as many banks of such controls as you care to have in your Rackspaces.

Controlling GigPerformer with the MCU Extension

Configuration of how GigPerformer interacts with the MCU is done through widgets.

On the "Advanced" tab of each GigPerformer widget there is a field called "OSC/GPScript Name". Any widget with a name that begins with "mc_" will be examined by the extension for information about how it should be utilized by the extension.

Widgets utilized by this extension fall into three broad categories:

- Control Widgets – knobs, button, etc. widgets typically linked to plugin parameters or system functions
- Indicator Widgets – provide feedback in GP showing which banks of Control Widgets the MCU is actively controlling
- Parameter Widgets – used primarily to determine how Control Widgets are displayed

Control Widgets

“Control Widgets” are typically knobs, faders, buttons, etc. linked in GP to plugin parameters. To be utilized by the MCU extension the GPScript Name must conform to the following format:

mc_type_bankID_position

“mc” identifies it as a widget of interest to the MCU extension

“type” indicates which row of MCU controls the widget will be associated with. The choices are:

- k = knobs
- f = faders
- solo, mute, rec, sel, = button rows aligned with the faders/knobs
- fn, view = additional button rows

“bankID” is any arbitrary name for this group of widgets that is meaningful to you. Typical examples:

- “Pan” for panning controls of a mixer plugin
- “solomute” if you wanted two RGB button rows to control solo/mute on a mixer plugin

“position” must be in the range of 0 – 8 to indicate which control position the widget is linked to

Faders are numbered 0-8 with 8 being the Master fader. All others are 0-7, arranged left to right

For example, a bank of eight knob widgets you want to use to control panning in a mixer could be named:

mc_k_pan_0   mc_k_pan_1   mc_k_pan_2   mc_k_pan_3   . . .   mc_k_pan_7

The bankID “pan” in this example is entirely arbitrary. I generally use bankID’s that are descriptive, but you could just as easily name your banks things like “bank1” or “xyz” if you’re so inclined.

The purpose of this bankID is so that you can select among multiple banks. You can define as many banks as you want of each control type.

Indicator Widgets

“Indicator Widgets” are similar to Parameter Widgets, but instead of controlling the colors of widgets on the SL-MK3 they are used to show on the Rackspace screen which widget banks are actively being controlled by the SL-MK3.

For example, if you have three separate banks of knobs in a Rackspace (e.g., instrument parameters, reverb parameters, and pans) it can be helpful to see on the GigPerformer screen which ones are actively linked to the knobs on the SL-MK3. You can change banks using the up/down arrow keys next to the control row on the SL-MK3 and the Indicator Widgets will reflect which bank is currently active

Indicator Widgets are named with the same format as Control and Parameter widgets but with “_i” in place of the _position.
e.g., sl_k_pan_i

Indicator Widgets are generally created as Text widget, most often with the text itself being blank or with the Text Color alpha set to zero so that the text itself does not appear on the Rackspace screen.

When the bank specified is “active” the extension will set the value of the widget to 1, which will raise its visibility on the Rackspace screen. When the bank is not active the value will be set to 0.3, which will reduce its visibility. A common use of these widgets is as an outline and background behind the associated set of Control Widgets.

The Caption of Indicator Widgets for knob and button banks controls the text that appears in the label areas on the right-most display on the SL-MK3, or temporarily in the Notify area when bank switching pad banks. Text for knob and button banks appears on two lines, which should be separated in the Caption by the “_” character. e.g., a Caption of “Solo_Mute” for a button bank would show the label “Solo” next to the top row of buttons, and “Mute” next to the bottom row.

Note – GigPerformer will remember Indicator Widget values when switching between Rackspaces/Variations/etc. As a result it will remember which banks you were controlling when you last used or saved a Variation, and when you return to it those same banks will be “active” again on the SL-MK3.

Note – the background color of an Indicator Widget is used for coloring the up/down bank select arrows on the SL-MK3. If you make each bank a different color, the up/down arrows will indicate which bank is next/previous as you bank select.

Parameter Widgets

“Parameter Widgets” are optional and can be used to specify how “Control Widgets” appear on the MCU.

The only Parameter Widget currently in use is of the form: mc_[f or k]_bankID_p

The Caption of these Parameter Widgets is displayed on the LCD when the bank is currently active on the MCU and being displayed on the LCD.

LCD Display

The MCU LCD displays are organized as two lines of text with seven characters each, aligned with each row of faders/knobs. On some units the displays of adjacent tracks run together; on others there is a gap. This can have a significant impact on readability. The extension was built for a unit with no gaps between displays, where it is actually two lines of 56 characters.

At any given time the extension will display captions of either the active Fader bank, Knob bank, or Songs/Rackspaces.

Standard button assignments are:

- EQ, Pan, Track – select Faders, Knobs, or Songs/Rackspaces on LCD display
- << >> - cycle through Fader banks
- < > - cycle through Knob banks
- Transport << and >> - cycle through Songs/Rackspaces in groups of 8
- Record – toggles in and out of Setlist mode
- Stop – stops the playhead
- Play – starts the playhead
- Jogwheel – adjusts tempo

Other button layouts can be selected from the MCU extension menu in GP. The purpose of the different layouts is to (try to) put the bank switching control buttons in positions that will be logical or easy to remember.

BankID Linking

If the same bankID is used for different control rows (e.g., Faders and Knobs) then when that bankID is selected to be active for one control row (e.g., Knobs or Faders) it will be selected for all rows that have a bankID of that name.

For example, if you want to be able to bank select between three separate 8 channel mixers plugins and have the entire control surface move together between them you would use Widget names like:

```
mc_f_mix1_0 … mc_f_mix1_7   and  mc_k_mix1_0 … mc_k_mix1_7   and  mc_mute_mix1_0   etc.
mc_f_mix2_0 … mc_f_mix2_7   and  mc_k_mix2_0 … mc_k_mix2_7   and  mc_mute_mix2_0   etc.
mc_f_mix3_0 … mc_f_mix3_7   and  mc_k_mix3_0 … mc_k_mix3_7   and  mc_mute_mix3_0   etc.
```

Name as such, when you bank select (using the << or >> Fader bank select keys) you will automatically also select the corresponding set of Knobs, Mute buttons, and any other button rows where you used that bankID.

In contrast, if you want to use the same three mixer plugins but be able to independently control the Volumes from the first on the MCU faders, the pans from the second on the MCU knobs, and the mutes for the third on the MCU mute buttons, you must use different bankIDs for each widget row. A simple example would be:

```
mc_f_volume1_0 … mc_f_volume1_7   and  mc_k_pan1_0 … mc_k_pan1_7   and  mc_mute_mute1_0   etc.
mc_f_volume2_0 … mc_f_volume2_7   and  mc_k_pan2_0 … mc_k_pan2_7   and  mc_mute_mute2_0   etc.
mc_f_volume3_0 … mc_f_volume3_7   and  mc_k_pan3_0 … mc_k_pan3_7   and  mc_mute_mute3_0   etc.
```

NOTE: there are presently no independent MCU key assignments for cycling through button banks. This is because there are too many independent button rows to make this practical and no obvious keys in the layout for doing so.

In my Rackspaces I tend to have either A) only one bank for each button row, or B) button rows that utilize multiple banks having bankIDs exclusively aligned to Knob or Fader bank bankIDs.

It may be easy enough to get around this by holding down something like the “Shift” key and simultaneously pressing the first or last button of an individual button row to bank switch that row. However, because of the limited visual feedback, the plethora of buttons available on the MCU, uncertainty that different hardware will respond similarly to held buttons, and inability to reliably do the same through an OSC display I have not found this worth pursuing.

Song and Rackspace Selection Buttons

Song/Rackspace and Songpart/Variation selection can be automatically linked to different button rows using widgets named “mc_rackrow” and “mc_variationrow”. The caption of each widget will specify what button row you want them assigned to. The options are the same as the “_type_” fields of the widget names. i.e., solo, mute, rec, sel, fn, view.

When these options are utilized the Songs/Racks or Songparts/Variations can be changed by pressing the appropriate button on the row. The first Song/Rack is assigned to the leftmost button on the row and successive ones are assigned going left to right.

The << and >> buttons (by default) will shift the buttons to the next 8 Racks/Songs. The currently active Rack/Song or Variation/Songpart will have its button lit on the surface.

If these widgets are utilized they will override any other widgets assigned to those button rows.

I tend to put these in the global rackspace so that they don’t move around or vanish as I change songs, racks, etc.

MIDI Device Selection

The MCU extension will look for widgets named “mc_midiin” and “mc_midiout” to determine which devices to attach to. The captions in these widgets contain a comma separated list of MIDI device names that the extension will attempt to connect to. These names must match the names that show up in the GigPerformer “Options -> MIDI Port” configuration window.

I typically list my physical MCU units as well as OSC virtual MIDI ports so that I can use an OSC either instead of, or in conjunction with, a hardware control surface. The extension receives MIDI information on the “mc_midiin” ports and outputs to the “mc_midiout” ports.

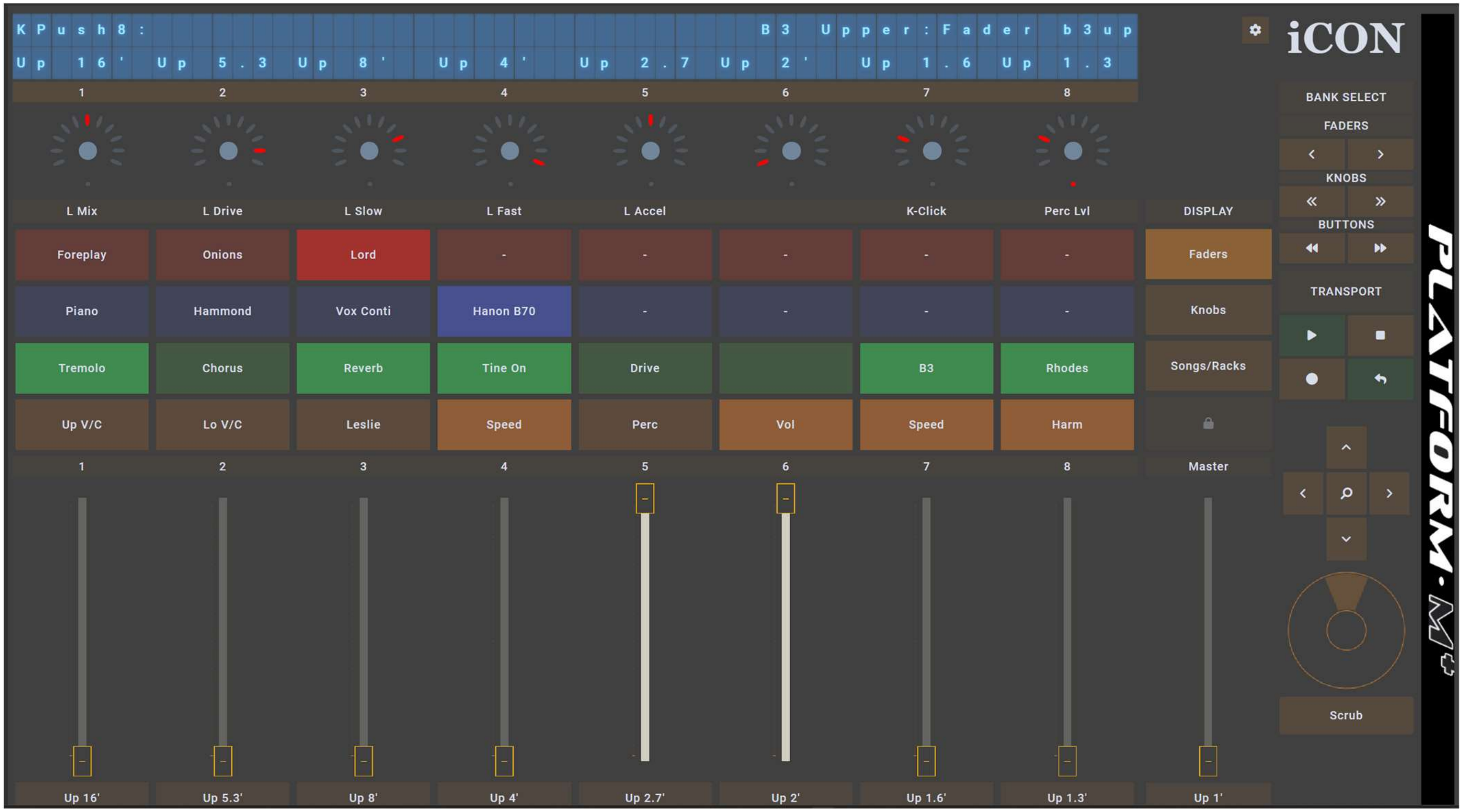
The MCU protocol is designed to accommodate 32 separate “channel strips” plus the master channel. In practice this means three “extension units” can be attached to a “master” MCU unit to give 32+1 faders, knobs, etc. This extension will only communicate using the data protocol associated with the first 8 channels plus the master.

Using with Open Stage Control (or OSC in general)

There are so many buttons on MCU units that I am prone to forgetting what I have assigned where. To address this I often use Open Stage Control to show what is assigned where. Sometimes use this on a touch screen instead of a physical MCU unit.

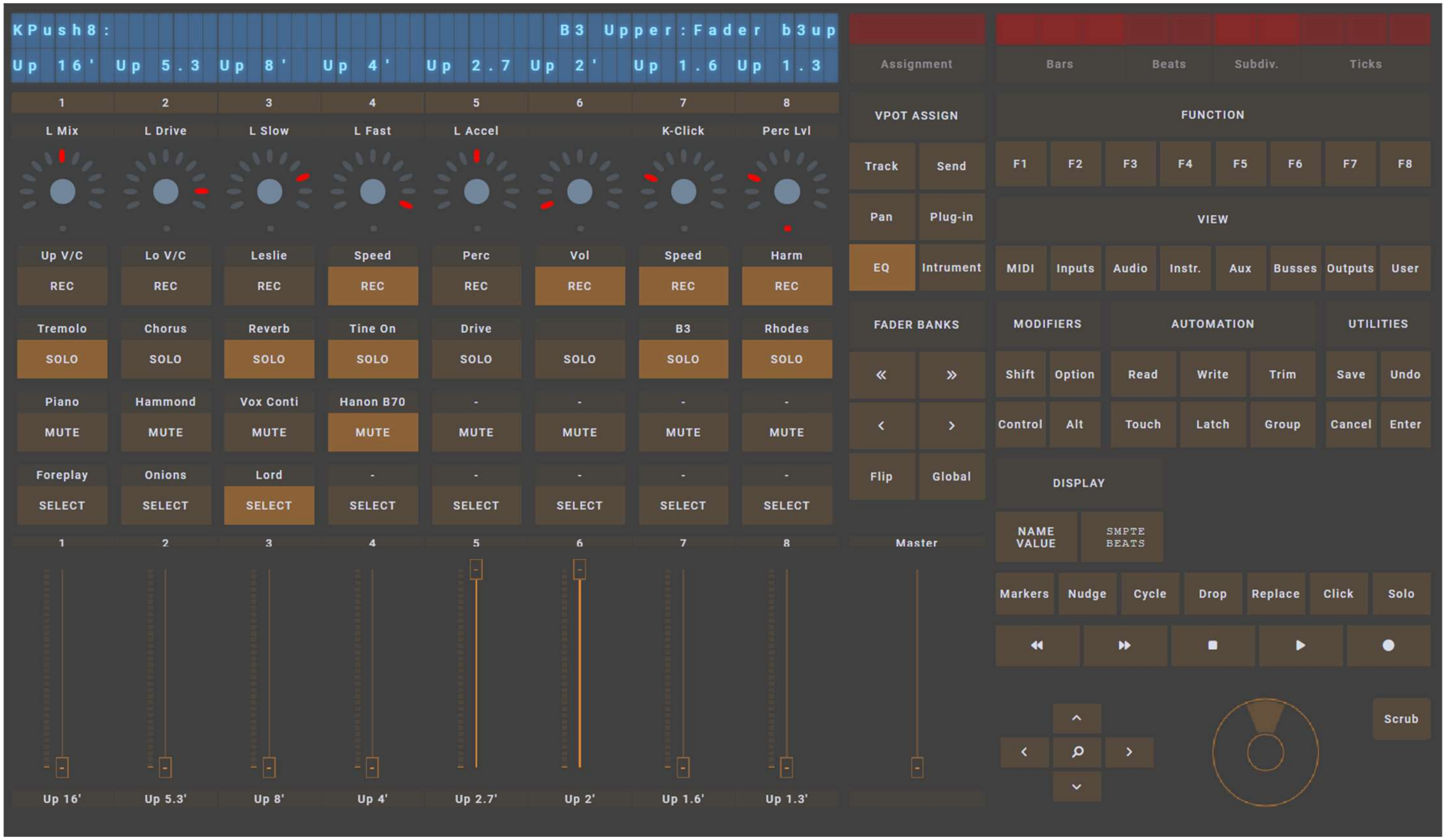
The MCU template created by the creator of Open Stage Control is available at:
https://github.com/jean-emmanuel/open-stage-control-mcu

The Touch OSC template is here: https://github.com/NicoG60/TouchMCU



To work with this extension (and these templates) you must also create appropriate “virtual midi ports” on your system. The MCU protocol is a MIDI protocol, so it has to run over MIDI.

I have modified the Open Stage Control template, and re-designed a version in the layout of an Icon Platform M+, to add text labels to the buttons, knobs, and faders. These modified versions are available in the same Git repository as this extension.



Other Comments

The OSC functionality of the extension requires using a Panel in the global rackspace with several dozen pre-defined widgets. This must be used for the OSC templates to work. Other than the MIDI In and MIDI Out widgets should not be user-modified.

Known Issues

The extension does not utilize the Bar/Beats/Ticks portion of the MCU display.

The MCU protocol calls for an initial “verification” exchange between the DAW and the MCU to allow it to function. It also calls for periodic “keep alive” messages to keep the connection working. This appears to have been part of a closely guarded “protection” scheme intended to only let licensed software communicate with official hardware. Current hardware does not generally use this, and this extension does not either as I have no mechanism to test it.

Many hardware devices claim to be “MCU compatible” or “HUI compatible” but it appears that definition varies widely.