

```
--
-- Called when the panel has finished loading
--
-- @type the type of instance beeing started, types available
in the CtrlrPanel
-- class as enum
--
-- InstanceSingle
-- InstanceMulti
-- InstanceSingleRestriced
-- InstanceSingleEngine
-- InstanceMultiEngine
-- InstanceSingleRestrictedEngine
--
myInit = function(--[[ CtrlrInstance --]] type)
    local dump_request =
L(panel:getLabel("LSYSEXREQ"):getText())
    if string.len(dump_request) == 0 then
        panel:getLabel("LSYSEXREQ"):setText("Enter in sysex
command for dump request")
    end

    _globVar =
        setmetatable(
            {},
            {
                __index = function(t, k)
                    local value = 0
                    if k == "startrecording" then
                        value = false
                    end

                    t[k] = value
                    return t[k]
                end
            }
        )
end
```

```
)

panel:getComponent("RECORD"):setProperty("uiButtonTextColourOn",
"ff000000",true)
panel:getComponent("RECORD"):setProperty("uiButtonTextColourOf
f","ff000000",true)
end

function myGetGlobalVariable(var)
    return _globVar[var]
end --function

function mySetGlobalVariable(var,value)
    _globVar[var]=value
end --function

-----
-- Called when a mouse is down on this component
--
start_recording = function(--[[ CtrlrComponent --]]
comp, --[[ MouseEvent --]] event)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true or
panel:getProgramState() == true then
        return
    end
    mySetGlobalVariable("startrecording",true)


    comp:setProperty("uiButtonTextColourOn","ffff0000",true)
    comp:setProperty("uiButtonTextColourOff","ffff0000",true)

    panel:getLabel("LSTATUSBAR"):setText("Waiting...")
end

-----
-- Called when a mouse is down on this component
```

```
--
save_to_computer = function(--[ [ CtrlComponent --
]] comp --[[ MouseEvent --]], event)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true or
panel:getProgramState() == true then
        return
    end
    if m:getSize() > 0 then
        saveToFile(m)
    else
        panel:getLabel("LSTATUSBAR"):setText("Nothing to save!")
    end
end
function saveToFile(dataToSave) -- save memoryBlock to .syx file
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true or
panel:getProgramState() == true then
        return
    end
    file = utils.saveFileWindow("Save file",
File.getSpecialLocation(File.userDesktopDirectory), "*.syx", true)
    if file:hasWriteAccess() then
        file:replaceWithData(dataToSave)
        panel:getLabel("LSTATUSBAR"):setText("Saved \187 " .. m:getSize() .. " bytes to sysex file ")
    end
end --function
-----
-- Called when the panel is created, no modulators will exist
at this point
-- consider this the panels constructor
```

```
--
constructor = function()
    m = MemoryBlock() -- failsafe initialization should be overridden in Loadstate
end
-----
-- Called when data is restored
--
load_state = function(--[ [ ValueTree --]]
stateData)
    savedHexString = stateData:getProperty("saveHexStringData")

    if savedHexString ~= nil and savedHexString ~= "" then
        m = MemoryBlock()
        m:loadFromHexString(savedHexString)
        panel:getLabel("LIN"):setText(savedHexString)
        
        panel:getLabel("LBUFF"):setText(m:getSize() .. " bytes")
    else
        m = MemoryBlock()
        panel:getLabel("LBUFF"):setText(m:getSize() .. " bytes")
        panel:getLabel("LSTATUSBAR"):setText("No data saved.
Initialised buffer")
    end
end
-----
-- Called when data needs saving
--
save_state = function(--[ [ ValueTree --]]
stateData)
    if m:getSize() > 0 then
        stateData:setProperty("saveHexStringData",
m:toHexString(1), nil)
    end
end
-----
```

```
-- Called when a mouse is down on this component
--
stop_recording = function(--[[ CtrlComponent --]]
comp, --[[ MouseEvent --]] event)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true or
        panel:getProgramState() == true then
        return
    end

mySetGlobalVariable("startrecording",false)

panel:getComponent("RECORD"):setProperty("uiButtonTextColourOn",
"ff000000",true)
panel:getComponent("RECORD"):setProperty("uiButtonTextColourOf",
f", "ff000000",true)

panel:getLabel("LSTATUSBAR"):setText("recording MIDI stopped")
end
-----
Called when a panel receives a midi message (does not need to
match any modulator mask)
-- @midi    CtrlrMidiMessage object
--
myMidiReceived = function(--[[ CtrlrMidiMessage --]] midi)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true
    or panel:getProgramState() == true then
        return
    end
    if (midi:getType() == 5) then -- it is a sysex message
        if myGetGlobalVariable("startrecording") == true then
```

```
        m:append(midi:getLuaData())
        panel:getLabel("LBUFF"):setText(m:getSize() .. " bytes")
        panel:getLabel("LIN"):setText(m:toHexString(1))
        panel:getLabel("LSTATUSBAR"):setText("Received
\\187 " .. m:getSize() .. " bytes")
    end
end
end
-----
-- Called when a mouse is down on this component
--
clear_buffer = function(--[[ CtrlComponent --]]
comp --[[ MouseEvent --]], event)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true
    or panel:getProgramState() == true then
        return
    end
    m = MemoryBlock() -- clear the MemoryBlock
    panel:getLabel("LBUFF"):setText("0 bytes")
    panel:getLabel("LIN"):setText("")
    panel:getLabel("LSTATUSBAR"):setText("Initialised buffer to 0 bytes")
end
-----
-- Called when a mouse is down on this component
--
upload_sysex_from_computer = function(--[[
CtrlComponent --]] comp --[[ MouseEvent --]],
event)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true or
    panel:getProgramState() == true then
```

```

        return
    end
    ret =
        AlertWindow.showOkCancelBox(
            AlertWindow.QuestionIcon,
            "LOAD SYSEX FILE TO PANEL FROM COMPUTER?",
            "Load sysex file to the panel for sending\nto
            machine\nPress SYSEX SEND to send.",
            "OK",
            "Cancel"
        )
    if ret == true then
        myLoadFromFile()
    end -- ret = true
end --function

myLoadFromFile = function()
    -- loads a sysex file and uploads
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true or
    panel:getProgramState() == true then
        return
    end

    local f = utils.openFileWindow("Open file",
    File.getSpecialLocation(File.userDesktopDirectory), "*.syx",
    true)
    fileRead = File(f)

    local buf = MemoryBlock()
    fileRead:loadFileAsData(buf)
    m = buf
    panel:getLabel("LBUFF"):setText(m:getSize() .. " bytes")
    panel:getLabel("LIN"):setText(m:toHexString(1))

```

```

        panel:getLabel("LSTATUSBAR"):setText("Loaded \187 " ..
m:getSize() .. " bytes from sysex file")
    end --function
    -----
    -- Called when a modulator value changes
    -- @mod    http://ctrlr.org/api/class_ctrlr_modulator.html
    -- @value    new numeric value of the modulator
    --
    sendSysexBackToSynth = function(--[[ CtrlrModulator --]]
mod --[[ number --]], value --[[ number --]], source)
        if panel:getBootstrapState() then
            return
        end
        if panel:getRestoreState() == true or
        panel:getProgramState() == true then
            return
        end
        local t = {}
        if m:getSize() > 0 then
            local j = 0
            for i = 0, m:getSize() - 1 do
                if m:getByte(i) == 247 then
                    table.insert(t, m:getRange(j, (i - j) + 1):toHexString(1))
                    j = i + 1
                end
            end
        end
        panel:getLabel("LSTATUSBAR"):setText("Sent \187 " .. m:getSize() .. " bytes")

        for i, v in ipairs(t) do
            panel:sendMidiMessageNow(CtrlrMidiMessage(v))
        end
    else
        panel:getLabel("LSTATUSBAR"):setText("Nothing to send!")
    end
end
    -----

```

```

-- Called when a mouse is down on this component
--
send_dump_request = function(--[[ CtrlComponent --
]] comp --[[ MouseEvent --]], event)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true or
    panel:getProgramState() == true then
        return
    end
    local dump_request =
L(panel:getLabel("LSYSEXREQ"):getText())
    if string.len(dump_request) > 0 then

panel:sendMidiMessageNow(CtrlrMidiMessage(dump_request))
        end
    end
-----
--
-- Called when a mouse is down on this component
--

help = function(--[[ CtrlComponent --]] comp, --[[
MouseEvent --]] event)
    if panel:getBootstrapState() then
        return
    end
    if panel:getRestoreState() == true
    or panel:getProgramState() == true then
        return
    end

utils.infoWindow("GENERIC SYSEX DUMP RECORDER",

```

```

[[
Receive Bulk Dumps of sysex from any MIDI Device

(1) Press RECORD to go on standby for incoming MIDI

(2) Optionally Enter in a custom dump request message and
click on "DUMP REQ"
        (*Click once in the field to the right of the "DUMP REQ"
Button)

(3) Press STOP to end recording.

(4) Send saved sysex bulk dump message back to MIDI device by
clicking SYSEX SEND

(5) Bulk dump messages are saved between sessions.

(6) To clear the current buffer click CLEAR BUFFER

(7) Save/Load a copy of your message to disk by clicking on
        (1) "SAVE SYSEX TO PC"
        (2) "LOAD SYSEX TO PANEL"

-----
Released under The GNU General Public License
]]
)

end

```