INDEX

THE ARTISAN Octo-Flex CONTROL SYSTEM

Introducing the Artisan OctoFlex Control System

July 2019

Artisan Instruments, Inc.
6450 NE 183
Kenmore, WA 98028
(425) 486-6555
Website: http://www.artisanorgans.com/
Email: artisaninstruments@mac.com - Sales
(425) 486-6555
Email:
emarka@mac.com - Support
(425) 922-6810

Table of Contents

Hardware Installation

ARTISAN OCTO-FLEX

HARDWARE SETUP

1. The console computer (always numbered 50) and, if there is more than 40 feet between the console and the chambers, the chamber computers (numbered 51, 52, 53, 54, etc.) This numbering system refers to the network address used so that the different components can “talk” to each other.
2. A network switch (or hub) for the console. If there are more than two chamber computers you will also need one additional network hub for every chamber except the last chamber in the network chain. This means if you have only one chamber computer you will not need an additional network hub. If you have two chambers you will need one hub in the first chamber and none in the second.
3. A user interface panel with LCD screen and four function buttons. This comes with a 6 wire flat data cable and is always connected to Port 1 of the first 8-Port board in the console system.
4. A short network cable for each hub.
5. A longer network cable to connect to your laptop computer (for diagnostics and programming only – not needed for normal operation of the organ.)
6. A USB extension cable so that a memory stick can be inserted from the front of the console (usually located under the key-desk.
7. A USB memory stick.
8. A MIDI out/in board with 6 wire flat data cable
9. An Analog to Digital board and data cable (ribbon cable)
10. One or more 8-Port boards stacked atop the console computer board and addressed as board 1 (nearest the computer board in the stack) or 2,3,4,etc. by use of the small blue rotary switch near the center of the board. This addressing is usually done at the factory before shipping your system.

INDEX

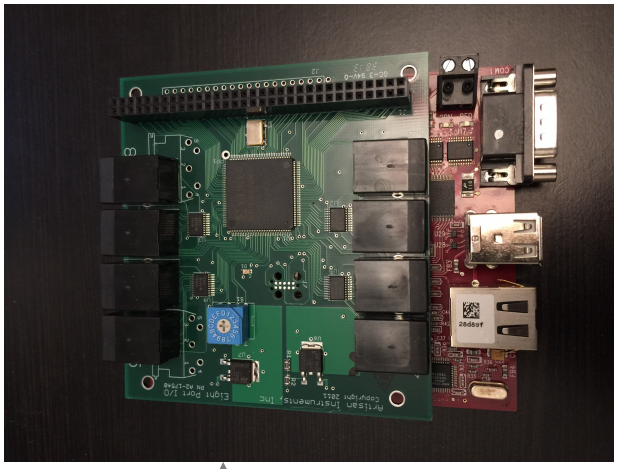


Illustration 1: Rotary Switch

Addressing Rotary Switch

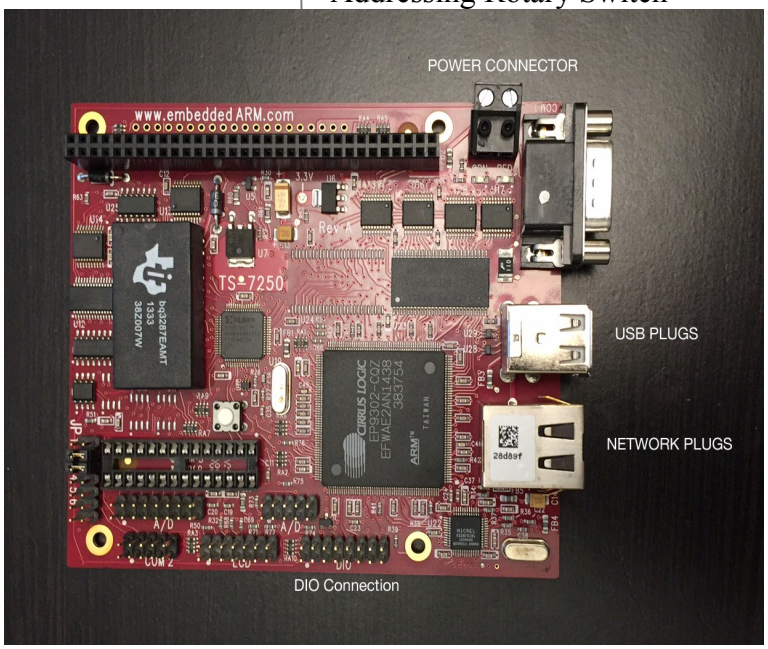


Illustration 2: Main Computer Board

For loading a program or doing system diagnostics you will connect your laptop (or desktop) computer via Ethernet cable through the supplied network switch (hub) and then open your browser and go to IP address 192.168.0.50

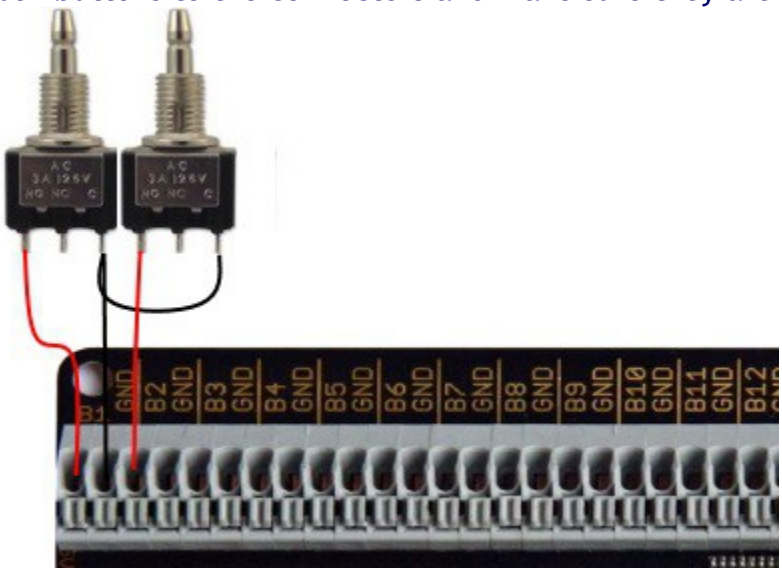
You can use any operating system you like, Windows, MAC, Linux, Unix, etc. Once connected you will see a screen in your network browser that displays the OctoFlex controls and allows you to edit and upload your main program.
Use Chrome web browser, or an up-to-date version of Firefox, Safari, IE 11, or Edge to connect to Octoflex

Connect your chamber system (if present) to the same network switch via an Ethernet cable.
Connect the A to D board using the 16-wire ribbon cable. One end goes to the DIO connector on the console computer board and the other end goes to the A to D board. This where you will connect all pots for expression, etc.

Connect The user interface screen using a 6-wire flat data cable to Port 1 of the first 8-Port board. There is a large number 1 printed on the circuit board immediately beside Port One.

Hints

1: Buttons on the LCD not working properly? Check the fine wires from the push buttons to the connectors and make sure they are properly secured.



2: If you have a USB memory stick mounted on the motherboard, don't disconnect it, it will store MIDI files that you have recorded and Registrations.

3: Save your configuration file before making any edits to the config file; same with the registration file.

4: After saving your new registrations ***wait a few minutes*** before powering down.

5: Use the Network Switch to connect the OctoFlex to your PC, and any chamber Octoflexes.

Illustration 3: Network Switch

using an Ethernet cable.

Illustration 4: Ethernet Cable

It is a good idea to label the cables, for easier service.
e.g. EIGHT_PORT_1
PORT 1

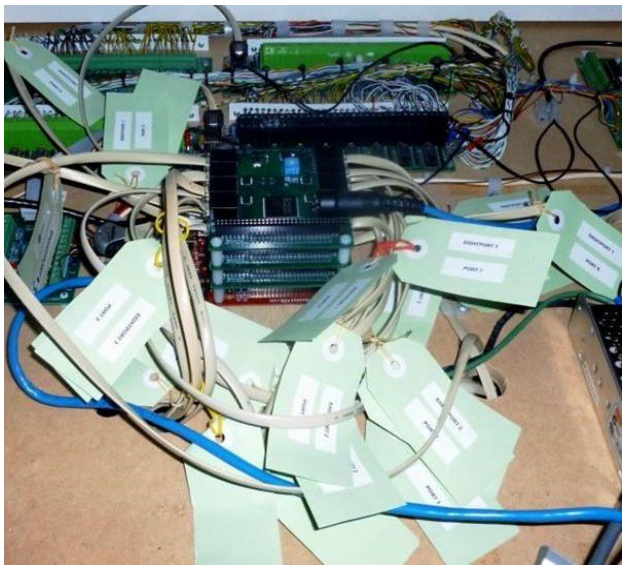


Illustration 5: Labels on Cables

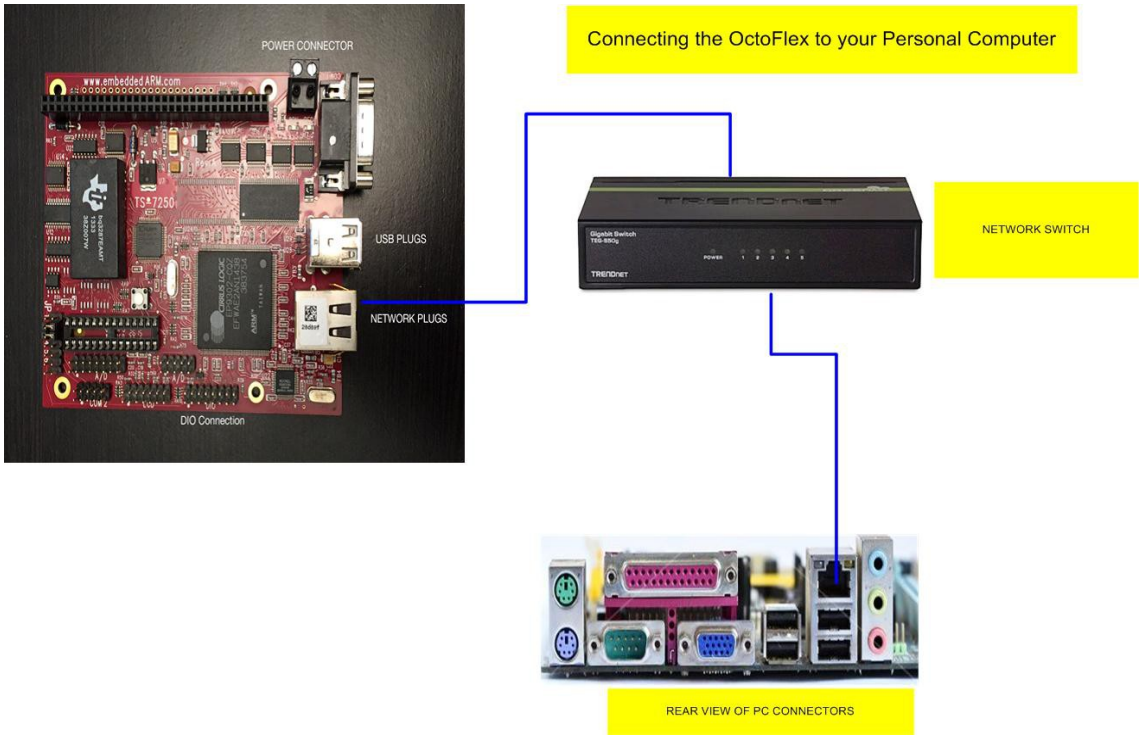
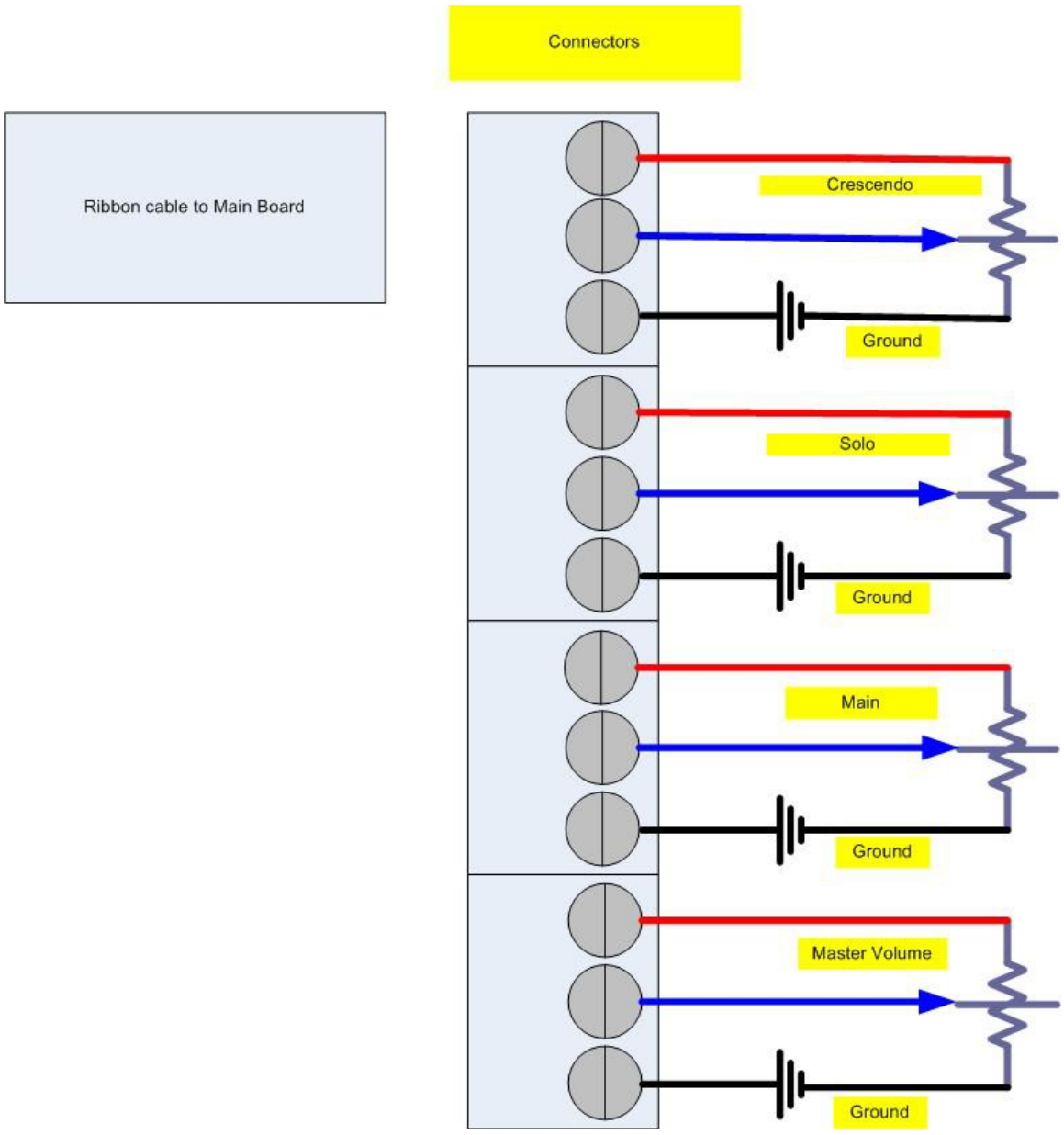


Illustration 6: Connection to Computer

Example: Connecting the swell and crescendo shoes to the Analogue to digital board.

Illustration 7: Connections To Analogue/Digital Board



Introduction

The config file consists of two parts:

- 1. Computer description
- 2. Connections

The computer description defines all the inputs and outputs of each computer, and some of the parameters specific to that computer. Connections are organ-wide parameters which may draw on resources from multiple computers.
In the text below, you will see many references to expressions.
Expressions are defined in Expressions

In the text below, you will see references to offsets. Let's tackle that one here. Without using any offsets – that is, if all offsets below are zero – the lowest note of a division will always play the bottom note of a rank. Offsets allow this behaviour to be modified. The general rule is that positive offsets will cause higher-pitched pipes to play when a given key is pressed on a keyboard, and negative offsets will cause lower-pitched pipes to play when a given key is pressed on a keyboard.

Let's look at couplers. If you have a 16-foot coupler but an 8-foot rank, you don't want the 8-foot pipe to sound, you want pipes one octave lower to sound. Thus, the proper offset for a 16-foot coupler is -12.
The offset in a *division is currently ignored by the software.
In order to create a configuration file, let's start with a template file.
Note that there are many features and options which are not mentioned in the template, so please read on to the following sections to learn about the rest!
Note that comments may be freely added to the configuration file by beginning them with a semicolon.

```
*computer 50 "Console" {
  *eight_port 1
    *port 1
      *hv64
        ; Enumerate inputs here. Here are some examples:
        1: *division pedal 0 24
```

```
        25: *input set
        26: *alt_action_input *sforzando
        27: *input trans_up
        28: *input trans_down
        29: *dual_mag_tab g_flute_4
      ; Add up to three more *hv64 boards on this port
    ; Add up to seven more ports
  ; Add more *eight_port modules as needed
; After all the inputs are tallied, you may have some
outputs
; Use the next available port
    *port 2
      *rank_driver
        ; Enumerate outputs here. These may be dual-magnet
tabs, lamps, etc. Examples:
        1: *lamp sforzando
        2: *mag_on g_flute_4
        3: *mag_off g_flute_4
; There is a shortcut for the preceding two lines:
    2: *mag_on_off g_flute_4
; After all the outputs are tallied, you may wish to add a
user interface.
; This includes an LCD display, ADC (analog) inputs, and
the interface for
; the built-in MIDI sequencer.
; Use the next available port
    *port 3
      *ui
        *lcd
          ; Put the definition for the LCD here
          ; Here is some sample code
          *line 1
            *if sfz
              "sforzando"
            *else
              "~b"
            *endif
           *line 2
              "Memory level" (mem_level)
           ; The following 4 lines and the matching *endif
should be present in order
           ; for the user interface to display its
information on the bottom two lines
```

```
            *if (_sbusy)
                *line 3 "~1"
                *line 4 "~2"
             *else
                *line 3
                   "line 3 text"
                *line 4
                    "line 4 text"
             *endif
          *midi_sequencer
            ; Put the definition for the MIDI mapping here
          *analog 1 crescendo 0 12
          ; This is an example of an analog input. More
information later.
;
; Put the details of the combination action here. Here is
an example:
*combination action set map 5 memory_select
;
; Put the details of the crescendo here. Here is an example:
*crescendo cres_set cres_map 12 crescendo
;
; Put information about the sforzando here
*sforzando sfz_set sfz_map sforzando
;
; If you want something to go "up" and "down", by pressing
two buttons, define that here
*increment_decrement transpose trans_up trans_down -6 6
;
; Each computer must always end with a "}"
}
;
; Usually, there will be one or more computers in chambers.
The chamber outputs
; could be connected to the first computer instead, if the
wire runs are short enough.
*computer 51 "Chamber" {
  *eight_port 1
    *port 1
      *rank_driver
        ; Describe ranks here. Example:
        1: *rank flute 85
    ; Add more ports and rank drivers as necessary
```

```
}
; More computers can be added here.
;
; Now it's time to make connections.
; Let's start with the couplers. Here are two samples
*coupler 1 swell to swell 0
*coupler great_to_swell_octave great to swell 12
; There must be at least one coupler for every division,
; otherwise no notes will sound on that division.
;
; Next, stops. Here is an example
*stop g_flute_4 great flute 12
;
; If you have a transpose control, list it:
*transpose transpose
; Of course, "transpose" must be defined somewhere.
;
; Definitions must always have the following at the end:
*end
```

Computer Description

The computer description consists of descriptions of what is connected to the computer, or functions performed by the computer.

The computer description begins with

```
*computer <computer_number>  "description" {
```

and ends with

```
}
```

Here, the <computer_number> connects the configuration with a specific computer. This is a number from 1 to 254, and corresponds with the last octet of the computer's IP address. By convention, the first computer is numbered 50, and is in the console. Chamber computers are numbered up from there. Any of the computers can be number 50, but in all cases, it must be powered on in order for the rest of the organ to function.

INDEX

Eight Port

An eight-port board description consists of descriptions of what is connected to its eight ports. It begins with

```
*eight_port <board_number>
```

The <board_number> is the number, from 1 to 7, that the switch on the board is set to.
After the *eight_port keyword, there may be up to eight port definitions.
For consistency, the eight-port board closest to the computer (on the bottom of the stack) is number 1 and any boards above that are 2-7 in order.
*Eight-port board ports 1 and 2 contain a larger input buffer than the other six ports. In applications in which a very large amount of MIDI information is flowing into the Octo-Flex system, it is preferable to use ports 1 and/or 2.

INDEX

Port

A port description consists of descriptions of the devices connected to that port. It begins with

```
*port <port_number>
```

The devices connected to a port may be

- An input64 board
- An hv64 board
- A lighted stop board
- A rank driver board
- A user-interface (display, buttons, ADC)
- MIDI input and/or output connection
-

INDEX

Input64

An input64 description begins with

```
*input64
```

and ends with the first keyword not describing inputs to that board. See hv64 for a description of those inputs.
Input64 refers to the older style Legacy input boards which operate by switching 5 volts to ground. HV64 refers to the modern input board which is selectable for positive or negative common inputs up to 33 volts.

INDEX

INDEX

HV64

An hv64 description begins with

```
*hv64
```

and ends with the first keyword not describing inputs to that board.

Inputs to HV64 and Input64 boards may be

- input
- alt_action_input
- division
- tab
- lit_tab
- dual_mag_tab
- piston
- reversible
- one_of_n_value
- binary_value

INDEX

Input

An input is used for any input switch which is not one of the special cases listed below. An input signal looks like

```
<input_number> : *input <name>
```

where <input_number> is the number of the input that is being described, and <name> is the name to be assigned to that input. For example:

```
*eight_port
*port 2
*hv64
2: *input transpose_up
```

Alt_action_input

This describes an input which is usually connected to a momentary switch. It creates a signal which alternates between on and off each time the switch is pressed. The description looks like

<input_number> : *alt_action_input <name>

where <input_number> is the number of the input that is being described, and <name> is the name to be assigned to that input. For example:

*eight_port

*port 2

*hv64

3: *alt_action_input sforzando

Division

This describes a group of inputs which are to be treated as a division. This is usually a keyboard. The description looks like

<input_number> : *division <name> <pitch> <count>

where <input_number> is the input number of the lowest note of the division, <name> is the name of the division, <pitch> is its pitch, and <count> is the number of consecutive inputs included in the division.

The <pitch> number allows a division to be offset in pitch from its normal status. In most cases, you should just make this zero. You could make the pitch of the pedal division -12, in which case, stops will sound one octave lower than otherwise identical stops on other divisions with the pitch set to 0.

Example:

*eight_port

*port 2

*hv64

1: *division swell 0 61

Tab

The tab description applies to an input which should be included in the combination action as a blind stop. The description looks like

<input_number> : *tab <name>

INDEX

Example:

*eight_port

*port 2

*hv64

1: *tab diapason_8

(This feature has not yet been tested)

INDEX

Lit_tab

The lit_tab description applies to an input which should be included in the combination action, and which has a light to indicate when it is engaged.

n: *lit_tab <name>

A push-on, push-off tab. The light needs to be explicitly added elsewhere as an output.

Optional

n: *lit_tab <name> *dual

takes two consecutive inputs (n and n+1) for an on and an off switch

INDEX

Dual_mag_tab

The dual_mag_tab description applies to an input which should be included in the combination action, and which has two magnets which allow it to be physically turned on and off. The description looks like

<input_number> : *dual_mag_tab <name>

Example:

*eight_port

*port 2

*hv64

1: *dual_mag_tab diapason_8

Note that for every *dual_mag_tab, there must be subsequent *mag_on and *mag_off definitions on a rank driver board.

INDEX

Piston

The piston description applies to an input, usually a momentary switch, which should be used by the combination action to select a preset. For "pistons" which are not combination action instigators, use the "input" keyword instead. The definition looks like

<input_number> : *piston <name>

where <input_number is the number of the input to which the piston is connected, and <name> identifies that piston for use elsewhere. Note that there is a special keyword for the general cancel input: *cancel

Example:

*eight_port

*port 2

*hv64

1: *cancel gen_cancel

2: *piston piston1

INDEX

Reversible

The reversible description applies to an input, usually a momentary switch, which should be used to turn a dual_mag_tab to its opposite position each time the switch is pressed. The definition looks like

<input_number> : *reversible <name>

where <input_number> is the number of the input to which the switch is connected, and <name> is the name of the dual mag tab that should reverse position when this switch is pressed.

INDEX

One_of_n_value

The one_of_n_value description applies to one or more consecutive inputs which are used to create a number, depending upon which input is pressed. Its definition looks like

<input_number> : *one_of_n_value <name> <num_inputs>

or

<input_number> : *one_of_n_value <name> <num_inputs> <min_value>

where <input_number> is the number of the first input, <name> is the identifier to be associated with the value, <num_inputs> is the number of inputs dedicated to this function, and <min_value> is the minimum value. Without <min_value>, the value of <name> will be between 0 and <num_inputs> - 1. With <min_value>, the value of <name> will be between <min_value> and <min_value> + <num_inputs> - 1.

Example:

*eight_port

*port 2

*hv64

1: *one_of_n_value transpose 12 -5

This defines a value named "transpose" which varies between -5 and 6, depending upon which input is activated.

If no input is activated, the number will be the most recent input activated, or zero if no input has ever been activated.

Binary_value

The binary_value description applies to one or more consecutive inputs which are used to create a number, depending upon which inputs are on at any time. The first input, if on, adds 1 to the number. The second adds 2, the third adds 4, and so on, with each subsequent input contributing twice the value of the previous input. A definition looks like

<input_number> : *binary_value <name> <count>

where <input_number> is the number of the fist input dedicated to this function, <name> is the identifier associated with the value, and <count> is the number of inputs dedicated to this function.

Example:

*eight_port

*port 2

*hv64

8: *binary_value memory_select 4

This defines a value named "memory_select" which varies between 0 and 15, depending upon which inputs are activated.

Rank driver board

A rank driver definition looks like

*rank_driver

and is followed by descriptions of what is connected to its outputs. This may include

- Rank
- Trap
- Mag_on
- Mag_off
- Lamp
-

Rank

A rank definition looks like

<output_number> : *rank <name> <pitch> <count>

where <output_number> is the number of the first output for the rank, <name> is the name of the rank, <pitch> is a pitch offset, and <count> is the number of consecutive outputs used by that rank.

Trap

A trap definition looks like

<output_number> : *trap <name>

where <output_number> is the number of the output for the trap, and <name> is the name of the trap

```
 *trap_stop (tab) div_name trap_name


Now define the trap under the *rank_driver, one of the
following:

5: *trap trap_name
5: *trap trap_name *pizz 100
5: *trap trap_name *reit 5

If you then do
6: *alternate trap_name
```

```
output 6 will alternate with output 5.
This only makes sense for *reit
But it might be interesting with *pizz: 5 comes on, then 6,
and 6 stays on until the trap goes away. Some kind of
 ker-bang effect?

If you have single inputs you wish to operate the trap too,
put
them in ():

5: *trap trap_name (toe_stud_5) (cheek_button_2) *reit 5
```


Mag_on

Mag_on is used to specify which output on a rank driver board is connected to the magnet which turns a dual-mag tab on. Mag_on looks like

<output_number>: *mag_on <name>

See mag_off for an explanation.

Mag_off

Mag_off is used to specify which output on a rank driver board is connected to the magnet which turns a dual-mag tab off. Mag_off looks like

<output_number>: *mag_off <name>

where <output_number> is the number of the output connected to the magnet coil which turns off a dual_mag_tab (or turns it on for mag_on), and <name> is the name of that dual_mag_tab. <name> must be the name of a dual_mag_tab defined elsewhere.

Mag_on_off

This is a shortcut for a *mag_on followed by a *mag_off.

<output_number>: *mag_on_off <name>

<output_number> is the number of the output connected to the on magnet, and <output_number>+1 is the output connected to the off magnet.

Lamp

The lamp command may be used to specify an output which is connected to a lamp, or any other device which doesn't fit another category. Lamp looks like

<output_number>: *lamp <expression>

where <expression> is an expression which defines when the lamp (or other output device) should be turned on. Examples might be

2: *lamp sforzando

3: *lamp (crescendo > 5)

*lamp is a synonym for *output

MIDI input and or output connection

A port can be used for MIDI input and/or output. There may be multiple MIDI ports. For each port, the MIDI map must be defined. That is, all MIDI messages that are to be generated and/or received must be enumerated. The MIDI definition begins with one or both of the following:

*midi_in

*midi_out

This is then followed by a list of the desired MIDI messages. The general form of this section is

<MIDI_message> <name>...

where <MIDI_message> is, for example, c1n23 for channel 1 note 23, c2c7 for channel 2 controller 7, or c3p4 for channel 3 program 4. This is followed by one or more identifiers for the items to be attached to that message. Identifiers after the first use the MIDI message that follow the message(s) used by the previous identifier. For example, divisions might look like this:

c1n36 swell

c2n36 great

Tab definitions might look like

c15n1 bourdon8 bourdon16 flute8 flute16

In this case, bourdon8 would be assigned to channel 15 note 1, bourdon16 would be assigned to channel 15 note 2, etc.

MIDI I/O defined this way will only transmit and receive raw divisions. Thus, the data will represent the raw data from the division input device, without any modification by couplers or transpose.

*midi_port <port #>

*midi_div_out (<enable>) <division> <base_note> <offset> [(velocity)]

where <enable> is an expression for when to output the division, perhaps (1) for always
<division> is the name of the division
<base_note> is something like c1n36
<offset> can be used to add an offset to the division; probably just 0
(velocity) is an optional expression used to control the velocity of the on notes.
(useful for a piano synthesizer that has different sounds at different velocities)
So, maybe
*midi_div_out (1) swell c1n36 0
Add information about
*midi_div_in
Used when a division is being defined by a MIDI input
*midi_div_out
Used to output a division to MIDI including the effects of couplers and transpose
*midi_tab_in
Used when a tab is being defined by a MIDI input.
*midi_note_in
*midi_control_in
A controller is being defined by a MIDI input.

*midi_div_in
Used when a division is being defined by a MIDI input
*midi_div_out
Used to output a division to MIDI including the effects of couplers and transpose
*midi_tab_in
Used when a tab is being defined by a MIDI input.
*midi_note_in
Why?
*midi_control_in
A controller is being defined by a MIDI input.

UI

The *ui keyword introduces the definition of the organ user interface. The user interface includes the LCD display and ADC inputs. The ADC inputs are defined by the *analog keyword, and the lcd display by the *lcd keyword. The UI section also contains the MIDI sequencer, and the MIDI messages that are to represent the organ activity must be defined, as described in the MIDI section above.

Special characters:

~p is replaced by the name of the most recently pressed piston

~c crescendo level

~f File name from sequencer

~m Metronome setting

~q Eighth or Quarter

~r Metronome rate

~s Metronome beats

MIDI Sequencer

OctoFlex can send and receive MIDI messages on one or more ports, and in addition, it uses MIDI messages to record performances using the built-in MIDI sequencer. The MIDI sequencer, and any external ports used for MIDI, must each have the MIDI messages that will be used for that port defined. This allows a great deal of flexibility, in that messages which are not needed for a particular port need not be output to that port.

The MIDI sequencer is defined within the *ui section. In this section, all the inputs and parameters which define a performance should be listed. This includes all divisions, all tabs and other inputs, and controllers such as shoes and crescendo. It should NOT include combination action pistons, as their effect is recorded by the tabs which they affect. It should not include inputs for increment-decrement statements, rather the increment-decrement value itself should be recorded. A reversible input should not be recorded, rather the tab which it affects.

The MIDI sequencer definition starts with the line
*midi_sequencer
and is followed with definitions of the desired MIDI messages.

< insert information about all the raw messages here >

An Octo-Flex system may also have any number of MIDI inputs and/or outputs. The top eight-port board of the stack may have MIDI connectors on ports 5 and/or 8. In this case, these ports may be used only for output messages. A small PC board is available which can connect to any port on any eight-port board, and allows MIDI input and/or output**.

A MIDI port definition begins with
*midi_port <port_number>
For example,
*midi_port 8
If any raw messages are to be used in this port (as defined above for the midi sequencer), you should now include a line which shows what the intended use of the port is. Include one of the following:
*midi_in
*midi_out
*midi_in_out

This is followed with definitions of all the raw MIDI messages as defined above.

In some cases, you may wish to use additional commands for a MIDI port. The following commands can not be used in the midi sequencer section.

Include here *midi_div_in etc.

Analog

The analog description describes the analog signals connected to the UI board. It looks like

*analog <channel> <name> <min> <max>

where <channel> identifies the input number on the UI board, 1 through 8; <name> gives a name to this signal to be used elsewhere, and <min> and <max> give the minimum and maximum values that are to be assumed as the analog voltage increases from minimum to maximum. Example:

*analog 3 solo_swell 0 20

LCD

The LCD description describes the appearance of an LCD connected to the UI board. It starts with

*lcd

and is followed by one or more sub-elements:

- line command
- text string
- value
- backlight intensity
- contrast

The LCD description can also be modified with conditional statements, including *if, *else, and *endif.

Line

The line directive is used to position subsequent text and values. It looks like

*line <line_number>

where <line_number> is a number from 1 to 4.

Text

Text is used to write fixed characters to the display. It is done simply by putting the desired characters in quotes. Example:

"Crescendo "

Note the space after the letter o. Spaces can be used in text as needed to align text and values as desired.

Value

A value may be displayed. This is done by putting an expression in parentheses. For example,

(transpose – 6)

The way the value is displayed may be modified by one of the following keywords:

- Binary
- Decimal

INDEX

- Signed
- Unsigned

INDEX

Binary

After a *binary command, any values will be output as an 8-bit byte. This may be useful for displaying special characters.

INDEX

Decimal

*decimal is the opposite of *binary, and restores normal display of numbers in decimal. *decimal may optionally followed by the number 1, 2, or 3. This will reserve that many spaces for the number.

INDEX

Signed

After *signed, values are displayed with an extra space reserved for a – sign. If a number is negative, it will be displayed that way.

Unsigned

After *unsigned, no space is reserved for a minus sign.

Backlight

The backlight command sets the intensity of the backlight. Valid values are from 0 (off)

to 100 (full intensity). It looks like

*backlight (<expression>)

For example,

*backlight (80)

Contrast

The contrast command can sometimes help make the display more readable, especially if it is being viewed from an unusual angle. Valid values are from 0 to 31.

*contrast (<expression>)

For example,

*contrast (15)

Conditional LCD statements

Conditional LCD statements begin with *if and end with *endif. The *if line looks like

*if (<expr>)

and is followed by some items which should be displayed if the <expr> is true. If there is something that you want displayed instead if <expr> is not true, follow the true items with *else and then the items to display when <expr> is false. You might have something like

*if (sforzando)

"SFZ"

*else

" "

*endif

where the *else clause contains three spaces to erase the SFZ when sforzando is off. If there are multiple test items, you may want to use *elseif. For example,

*if (sforzando) "SFZ"

*elseif (crescendo) "CRZ"

*else " "

Here, if sforzando is on, SFZ is displayed. If not, but crescendo is non-zero, CRZ is displayed. If neither sforzando is on or crescendo is non-zero, three blank spaces are displayed. Any number of *elseif clauses may be included. *if statements may be nested.

Special LCD characters

There are some special characters which display various things. For example, if you display the string "~b", the rest of the LCD line will be filled with blanks. There is some boiler plate which should be included in your LCD definition in order for the user interface to work properly. It is

*if (_sbusy)

  *line 3 "~1"

  *line 4 "~2"

*else

;place here the desired contents of lines 3 and 4 when the user interface is not active

*endif

Here, _sbusy is a special variable which is true when the user interface has something to display, and ~1 and ~2 are the two lines of the user interface.

Special characters:

~p is replaced by the name of the most recently pressed piston

~c crescendo level

~f File name from sequencer

~m Metronome setting

~q Eighth or Quarter

~r Metronome rate

~s Metronome beats

Combination Action

Combination action is implemented by first defining the various inputs and outputs, and then declaring the combination action itself. The combination action is connected to a particular computer, and thus the combination action should be included in the definition of the console computer. The definition looks like this:

*combination_action <learn> <map> <levels> <level_select>

where <learn> is an expression which defines when a piston setting is to be learned, <map> is an expression used to map which tabs the piston affects, <levels> is an integer specifying how many memory levels the combination action should have, <level_select> is an expression which determines which memory level is active at any moment. An example might be

*combination_action set map 5 memory_level

where set and map are probably defined in an *input, and memory_level is defined by some numeric source, such as a *binary_value or *one_of_n_value.

The system will collect all the tab resources, which include *dual_mag_tab, *tab, *lit_tab, and *piston and implement the combination action.

Crescendo

A crescendo function may be added with the following definition:

*crescendo <learn> <map> <step_count> <step>

You may also have multiple crescendo definitions with the following definition:

*crescendo <learn> <map> <step_count> <step> <levels> <level_select>

Here, <learn> is an expression which determines when the crescendo is to be memorized, <map> is used to display which stops are affected by the crescendo, <step_count> is how many steps are present in the crescendo, and <step> is the expression which determines which step of the crescendo should be active at any moment. For the second variation, <levels> defines how many different crescendo definitions there should be, and <level_select> determines which should be active at any time.

The crescendo settings are saved along with the combination action settings.

Sforzando

A sforzando function may be defined as follows:

*sforzando <learn> <map> <sforzando_on>

where <learn> determines when the sforzando should be memorized, <map> specifies the input which allows the current sforzando setting to be displayed, and <sforzando_on> is an *input which determines when the sfz should be active. Multiple sfzs may be defined by using the following definition:

*sforzando <learn> <map> <sforzando_on> <levels> <level_select>

where <levels> determines how many sfzs should be defined, and <level_select> determines which definition to use at any time.

Note that <learn> and <map> may well be the same as the combination action <learn> and <map>s, since sfz definition is much like another piston. The sforzando settings are saved along with the combination action settings.

Piston Sequencer American

A piston sequencer allows one to memorize a sequence of piston pushes, and then play back that sequence by operating a special switch. Multiple sequences may be memorized. The sequences are saved along with the combination action settings.

The piston_sequencer specification looks like

*piston_sequencer <size> <learn> <next> <previous> <reset>

or

*piston_sequencer <size> <learn> <next> <previous> <reset> <levels> <level_select>

Here, <size> is an integer specifying the maximum number of piston-pushes which may be included in one sequence. <learn> is an expression (probably an input) which puts the piston sequencer into a mode where it memorizes a sequence of piston pushes. <next> is the input which is used to advance the sequence during playback. <previous> is the input which is used to back up in the piston sequence being played. <reset> is an input which resets the sequence back to the beginning, so that pressing <next> will "press" the first piston. If present, <levels> is an integer specifying how many different piston sequences should be memorized, and <level_select> is an expression which specifies which of the sequences is the current one.

Piston Sequencer European

In the configuration file, inside the *computer {}, probably next to
the *combination action, add

*natural_order_sequencer up down piston1 piston2 ...

where up is the name of an input which will increment the piston,
down is the name of an input which will decrement the piston,
and piston1, piston2, and more are the names of all the *pistons which you wish to be part of the sequence.
You can add line breaks as you wish to make it look pretty.

If you push any of the pistons listed, it will remember which one was pressed, along with what the current memory level is.
Pushing up will push the next piston, advancing to the next memory level if necessary, and wrapping around from the end to memory level 0.
Pushing down will do the opposite.

You may employ both types of piston sequencers on the same organ if desired.

Outputs

The items described above are primarily used for the console computer, which is mostly inputs. Here we will describe the items which are normally found in a chamber computer, which are primarily outputs.

n: *lit_tab <name> *dual takes two consecutive inputs (n and n+1) for an on and an off switch

Rank Driver

A rank driver board can driver up to 96 outputs. It is declared with a *rank_driver statement which is placed after a *port declaration. For example:

*computer 51

*eight_port 1

*port 2

*rank_driver

The *rank driver should be followed by descriptions of everything connected to its outputs.

Driver boards HD32, HD64, HD76, HD100

Rank

A set of outputs forming a rank should be declared with *rank. The general definition is

<output_number>: *rank <name> <count>

Here, <count> outputs starting at output number <output_number> are used to drive a rank of pipes. The <name> will be referenced in one or more *stop statements. The lowest pipe should be connected to the lowest output.

Shades

A set of outputs may be activated sequentially, as is required for operating shades. The general definition is

<output_number> *shades <exp> <count>

Here, <count> outputs starting at output number <output_number> are used to drive a set of shades. The <exp> is an expression which determines how many of the outputs should be active. If <exp> evaluates to 0, none of the outputs are activated. If <exp> evaluates to 1, the lowest numbered output will be active. And, if <exp> evaluates to a number which is <count> or greater, all of the outputs will be active.

Connections

Connections describe interactions which aren't necessarily limited to one computer. They include

- pseudo_division
- bit

- increment_decrement
- coupler
- melody_coupler
- bass_coupler
- midi_voice
- midi_trap
- swell
- stop
- pizz_stop
- reit_stop
- trap_stop
- mag_pulse_time
- transpose
- sustain
- sostenuto
- midi_reset

pseudo_division

At times it is convenient to create a division which is not associated with a physical keyboard. This is sometimes called a floating division. It is defined simply with

*pseudo_division <name> <offset>

bit

It is sometimes useful to give a name to some combination of inputs. This is done as follows:

*bit <name> <exp>

where <name> is the name of the bit, and <exp> is an expression that determines its value.

increment_decrement

A value may be defined by having two inputs, one of which increments the value and the other which decrements the value. This is sometimes used for a transpose feature. This looks like this:

*increment_decrement <name> <inc> <dec> <min> <max>

where <name> is the name to be assigned to the value, <inc> is the input which causes the value to increment, <dec> is the input which causes the value to be decremented, <min> is the minimum value, <max> is the maximum value. If it is desired for the value to jump from its minimum value to its maximum value when decremented from its minimum value, and vice versa, add the keyword *wrap after the increment_decrement definition. Example:

*increment_decrement transpose transpose_inc transpose_dec -5 6 *wrap

Coupler

At least one coupler must be defined for each division that is to play any notes. If the division is only used for MIDI output, then the coupler is not necessary. A coupler definition looks like:

*coupler (<exp>) <from_div> to <to_div> <offset>

where <exp> is an expression which determines when the coupler is active, <from_div> is the division which makes sounds, <to_div> is the division which is to cause the sounds to happen, and <offset> alters the pitch of the coupler. See Expressions for a description of expressions.

The default condition is for the lowest note of the division to play the lowest note of the rank to which it is eventually connected. Here, the <offset> can be used to modify this default. Negative numbers will cause lower sounds to be generated, and positive numbers will cause higher pitched sounds to be generated. Hence, <offset> should be -12 for a sub-octave coupler, and 12 for an octave coupler. Examples:

*coupler (1) swell to swell 0

*coupler (swell_octave) swell to swell 12

Offsets.

16' -12

8' 0

4' 12

2' 24

1' 36

Mutations.

5 1/3' 7

3 1/5' 16

2 2/3' 19

1 3/5' 28

1 1/3' 31

melody_coupler

A melody coupler works just like a coupler, except that only the top note of the "To" division is played. Example:

*melody_coupler (solo_to_swell_mel) solo to swell 0

bass_coupler

A bass coupler works just like a coupler, except that only the lowest played note of the "to" division is played. Example:

*bass_coupler (solo_to_pedal_bass) solo to pedal 0

midi_voice

A MIDI voice is an external synthesized rank. I think. Check back later for developments.

midi_trap

A MIDI trap is a trap which is implemented by some external MIDI device. When the first note of the selected division is played, the selected MIDI note on message is transmitted. Its format is:

*midi_trap (<expression>) <source_division> <MIDI_note>

where <expression> defines when the trap should be active, source_division is the division which should activate the trap, and <MIDI_note> is the note which should be transmitted in CiNj format, such as C12N23 for channel 12 note 23. Example:

*midi_trap (midi_siren) solo C12N23

Stop

The stop keyword is used to connect divisions and pseudo-divisions to ranks. It looks like

*stop (<exp>) <division> <rank> <offset>

or

*stop (<expr>) <division> <rank> <offset> <min> <max>

where <expr> is an expression which determines when the stop should be active, <division> is the source of the notes to play, <rank> is the name of the rank which should play, <offset> allows pitch alteration, and <min> and <max> allow a subset of the rank to be controlled by the stop.

See Expressions for a description of <exp>

<offset> allows the pitch of the stop to be altered. By default, the lowest note of the division will play the lowest note of the rank, assuming a unison coupler. Making <offset> negative will cause lower pipes to be played, and making it positive will cause higher pipes to be played.

If the <min> and <max> values are included, only the rank outputs between <min> and <max> will be controlled by this stop. For example, if <min> and <max> are 13 and 24, only the second octave of the rank will play.

Examples:

*stop (flute8) great flute 0

*stop (flute16) great flute -12 1 25

pizz_stop

A pizz_stop is just like a stop, except that whenever a note is sounded, it only remains on for a limited amount of time. The length of time the note is on can vary from the lowest note to the highest note, if desired. One might want lower notes to sound for a longer time than the higher notes. The format is:

*pizz_stop (<expr>) <division> <rank> <offset> <low_ms> <high_ms>

where<expr> is an expression which determines when the pizz_stop should be active, <division> is the source of the notes to play, <rank> is the name of the rank which should play, and <offset> allows pitch alteration. <offset> allows the pitch of the stop to be altered. By default, the lowest note of the division will play the lowest note of the rank, assuming a unison coupler. Making <offset> negative will cause lower pipes to be played, and making it positive will cause higher pipes to be played.

<low_ms> defines how many milli-seconds the lowest note should sound for, and <high_ms> how many milliseconds the highest note should sound for. The notes between will be scaled linearly. Example: *pizz_stop (flute_pizz) solo flute 0 500 400

reit_stop

A reit_stop is similar to a pizz_stop, except that the note repeats. Similar to pizz_stop, one might want the lower notes to repeat at a slower rate than the high notes. Its format is

*reit_stop (<expr>) <division> <rank> <offset> <low_rate> <high_rate>

where low_rate defines how many times per second the lowest note should repeat, and <high_rate> defines how many times per second the highest note should repeat. The duty cycle will be 50%.

trap_stop

A trap_stop connects a division to a trap. When a note on the division is played, the trap is operated. Its format is:

*trap_stop (<expr>) <division> <trap>

where <expr> is an expression that determines whether the stop is active, <division> is the controlling division, and <trap> is the name of the trap. Example:

*trap_stop (solo_snare) solo snare

There are several modifiers which may be added to a trap_stop. They are downbeat, upbeat, reit, and pizz.

Downbeat

Normally, a trap will operate when the first note of a division is played. A trap_stop may be modified by adding a *downbeat declaration, which changes this. The format is:

*downbeat <count>

where <count> is the number of notes which must be played on the division before the trap is operated. For example, if

*downbeat 2

is added after a *trap_stop, the trap will not operate until the second note of a division is played, and will operate until one or less notes are played.

Upbeat

If *upbeat is added after a *trap_stop definition, the trap will operate when the last note of a division is released. All *upbeat trap_stops require a pizzicato time, since there is no other way to determine how long the trap should be held. The format is

*upbeat <time_ms>

where time_ms is the number of milliseconds that the trap should operate.

Reit

Adding a *reit declaration to a trap stop will cause the trap to reiterate. The format is

*reit <speed>

where <speed> is the number of times per second that the trap should operate. The duty cycle will be 50%.

Pizz

Adding a *pizz declaration to a trap stop will cause the trap to operate for a fixed duration of time. Its format is

*pizz <time_ms>

where <time_ms> is the time in milliseconds that the trap should operate. Note that specifying *upbeat already makes the trap pizzicato, and *pizz should not be added.

If the last note (or the note determined by *downbeat) is released before <time_ms> has elapsed, the trap will be released at that time.

mag_pulse_time

When dual magnet tabs are included, mag_pulse_time specifies how long the output pulses should be to operate the tab. Its format is

*mag_pulse_time <time_ms>

where <time_ms> is the number of milliseconds that the magnets should be energized. The default is 100 milliseconds.

```
*computer 50 "Console" {
}  ; End of computer definition
*mag_pulse_time 100
```

Notes:
(Units, mSec
Default, 100
Location, Outside of any *computer { }
Limits, 1 to 512)

INDEX

Transpose

A transpose may be implemented as follows:

*transpose <exp>

where <exp> is the expression which determines how many semitones the transpose should be raised.

Sustain

*sustain allows a division to be held by operating an input. While the sustain is active, any note which is played on the division is held, and not released until the sustain is released. Its format is

*sustain <division> (<expr>)

where <division> is the name of the division which should be sustained, and <expr> is the expression which determines when the sustain should be active.

Sostenuto

*sostenuto allows a set of notes to be held, but not modified, as long as the sostenuto is in effect. Its format is

*sostenuto <division> (<expr>)

where <division> is the division to be modified, and <expr> specifies then the sostenuto should be in effect. Example:

*sostenuto solo (solo_sost)

When a sostenuto is first turned on, the notes which are currently active on the division are remembered. Those notes are held on during the sostenuto, while other notes can be played normally.

midi_reset

Occasionally, the MIDI system may have a note or stop stuck on. The midi reset function can be used to reset all MIDI notes within the controller. It is defined this way:

*midi_reset <reset>

where <reset> names an input to be used. It's usually convenient to just connect this to the cancel piston. Example:

*midi_reset gen_cancel

Expressions

In the above text there are many references to expressions. In the simplest case, an expression may just be a number or the name of an input. However, in many cases, it is desirable to combine several inputs or modify them in some way. For example, you might want a coupler to be active unless the input unison_off is true. You might type

*coupler (!unison_off) swell to swell 0

Here, ! is the symbol for "not", so the coupler will be active whenever unison_off is not true.

When an expression is called for, you may enter an integer or a simple name. If, however, any of the operators mentioned below are to be applied, the expression must be enclosed in parentheses.

Here are the allowed operators:

!	Not
~	Binary complement
*	Multiply
/	Divide
%	Modulo
+	Add
-	Subtract
min	Minimum
max	Maximum
shl	Shift left
shr	Shift right
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to (note two equal signs in a row)
!=	Not equal to
&	Binary and
&&	Logical and
^	Binary exclusive or
|	Binary or
||	Logical or
? :	Inline if
(..)	Parentheses force evaluation order

Some details which may help in forming expressions:

- A bit name, such as an input bit or a dual_mag_tab has the value 0 when off, and 1 when on.
- The ? : operator might need some explanation. It is a three-operand operator, for example "a ? b : c". In this case, the expression a is evaluated. If it is true (non-zero), then b is evaluated and it is the value of the expression. Otherwise, c is evaluated, and that is the value of the expression.
- When an expression is used for an on-off decision, it is on whenever it is non-zero, and off when it is zero.
- Be careful when combining things with &. For example, if my_switch is an input bit, and crescendo is a value, (my_switch & crescendo) is true whenever the binary and of my_switch and crescendo is non-zero, that is when my_switch is true and crescendo is odd. Perhaps you mean (my_switch && crescendo), which would be true whenever my_switch is true and crescendo is non_zero.
- Min and max are dyadic: They take two arguments, one to the left and one to the right. For example, (shoe1 max shoe2).

Using the function buttons

The three Function buttons on the user interface can be used for other purposes when the menu system is not active. This can be done using the special names _funcion1, _function2, and _function3.

As an example, the following code allows the buttons to be used as an increment-decrement controller, for example for a memory select.

First, in the *lcd definition, we prompt the user

*line 4 "Memory DOWN UP" (mem_level+1) "~b"

Then, usually toward the bottom of the computer, we implement the increment-decrement:

*increment_decrement mem_level _function3 _function2 0 24 *wrap

In line 4 of the display, we have put spaces so that DOWN is near the third button (_function2) and UP is near the fourth button (_function 3). Since mem_level has values from 0 to 24, we have made it more user-friendly by adding one, and displaying 1 through 25. The "~b" is a special code which fills the rest of line 4 with blanks.

When the user presses the right-most button, this makes _function3 go true, and mem_level is incremented. When the user presses the third button, _function2 goes true, and mem_level is decremented. The value will wrap; that is, a decrement from 0 will go directly to 24, and an increment from 24 will go to 0.

Octo-Flex configuration files

Moller

```
;Moller 11677

*computer 50
{
  *eight_port 1
    *port 1
      *ui
      *analog 1 solovol 10 127
        *analog 2 positifvol 10 127
        *analog 3 swellvol 10 127
        *analog 4 t_cresc 0 16
        *lcd
        *decimal 2
          *line 1
          *if (sfz) "   Sforzando On~b"
          *else
        "   Mark Andersen~b"
           *endif
            *line 2
              " Crescendo =  "
            *if (t_cresc) (t_cresc) *else "Off" *endif "~b"
          *if (_sbusy)
            *line 3 "~1"
            *line 4 "~2"
          *else
            *line 3 " Piston: "
            *if (pmap)
              *if (pset)
                  "Map Set~b"
                *else
                  "Map Display~b"
                *endif
              *else
                *if (pset)
                  "Set~b"
                *else
                  "~p"
                *endif
             *endif
      *line 4 "Memory  DOWN   UP" (mem_level) "~b"
     *endif

*midi_sequencer
        c5n36 Solo
        c4n36 Pedal
        c3n36 Positiv
        c2n36 Swell
```

```
        c1n36 Great

c15n1
        gGreat4         gBasson16         gMixtureIV
      gGedeckt4
        gBordun8         gGemshorn8       gTrompete8               gCymbale
        gSuperoctave2    gOvtave4         gPrincipal8
      gGemshorn16
        gChimes         pPositif4         pPositif16
      pCromorne8
        pClochesIII      pQuint113        pSpillflote4
      pErzahlerceleste8
        pHolzgedeckt8    pErzahler16      pTremulant
      pCoranglais8 pZimbelIV pSifflote1         pKleinoctave2  pOctave4
        pErzahler8 pPrincipal8  soTremulant  soFestival4
        soFestival8  soFestival16  soDoppelflote4  soDoppelflote8
        soCelesta4       soZimbelstern       soCarillon
      soSchalmey4
        soFrenchhorn8    soSchalmey8             soGambaceleste8
      soGamba
        soDoppelflote16 soHarp8                  soChimes

c16n1
        swTremulant      swChalumeau8         swHautbois8
      swPleinJeuIV
        swNasard223      swPrincipal4         swViola8
      swSwell16
        swVox8           swBasson16               swBlockflote2
      swWaldflote4
        swViolaceleste8 swRohrflote8         swSwell4
      swClairon4
        swTrompette8  swTierce135    swHohlflote4 swFlautoceleste8
        swFlauto8        swRohrbass16         pedHautbois8
      pedBasson16
        pedRohrflote4    pedGemshorn8         pedErzahler16
      pedGemshorn16
        pedFagot4        pedBombarde16        pedMixtureIV
      pedRohrflote8
        pedOctave8       pedRohrbass16        pedViolone32
      pedBombarde8
        pedBombarde32    pedChoralbass4       pedBourdon8
      pedSubbass16
        pedPrincipal16  g2ped8                    sw2ped8
      sw2ped4
        pos2ped8         pos2ped4           so2ped8
      sw2gt16
        sw2gt8           sw2gt4                   pos2gt16
      pos2gt8
        pos2gt4         gt2pos8                 sw2pos16          sw2pos8
        sw2pos4         so2gt8                  so2sw8
            so2pos8
        so2pos4
```

```
c7c7 swellvol
c8c7 positifvol
c9c7 solovol
c7c80 t_cresc

 *port 2
      *hv64
      1:  *division Solo '8' 61
      64: *input pmap
      *hv64
        1: *piston      Positif_4
        2: *piston      Positif_1
        3: *input     pset
        4: *piston      Positif_0
        5: *piston      Positif_2
        6: *reversible  pos2ped8
        7: *piston      Positif_3
        8: *piston      Positif_7
        9: *piston      Positif_6
        10: *piston     Positif_5
        ; 11: *piston   piston11
        12: *piston     Positif_8
        13: *cancel   Cancel
        14: *piston     General_7
        15: *piston     Great_7
       ; 16: *piston    piston16
        17: *piston     Great_4
        18: *piston     General_9
        19: *piston     Great_2
        20: *piston     Great_5
        21: *piston     General_12
        22: *piston     Great_8
        23: *piston     General_8
        24: *piston     Great_3
        25: *reversible g2ped8
        26: *piston     Great_1
        27: *piston     Great_0
        28: *piston     General_10
        29: *piston     General_11
        30: *piston     Great_6
       ; 31: *piston    piston31
        32: *piston     Swell_6
        33: *piston     Swell_8
        34: *piston     Swell_7
        35: *piston     Swell_4
        36: *piston     Swell_1
        37: *piston     General_5
        38: *piston     General_3
        39: *piston     General_6
       ; 40: *piston    piston40
        41: *piston     Swell_5
       ; 42: *piston    piston42
```

```
     43: *piston     General_2
     44: *piston     Swell_0
     45: *piston     Swell_3
     46: *piston     General_4
     47: *alt_action_input    sfz
     48: *reversible sw2ped8
     49: *piston     General_1
     50: *piston     Swell_2
     51: *piston     Solo_4
     52: *piston     Solo_6
     53: *piston     Solo_1
     54: *piston     Solo_5
     55: *piston     Solo_8
     56: *piston     Solo_3
     57: *reversible so2ped8
     58: *piston     Solo_7
     59: *piston     Solo_0
     60: *piston     Solo_2
    ; 61: *piston    piston61
    ; 62: *piston    piston62
    ; 63: *piston    piston63
    ; 64: *piston    piston64

*port 3
  *hv64
  1: *division Positiv '8' 61

  *hv64
  1: *division Great '8' 61

  *hv64
  1: *division Swell '8' 61

*port 4

  *hv64
  1: *reversible    so2ped8
    2: *piston      Toe_0
    3: *piston      Toe_7
    4: *piston      Toe_4
   ; 5: *piston     piston69
    6: *piston      Toe_3
    7: *piston      Toe_1
    8: *piston      Toe_5
    9: *piston      Toe_6
    10: *reversible sw2ped8
    11: *reversible g2ped8
    12: *piston     Toe_8
   ; 13: *piston    piston77
    14: *reversible soZimbelstern
   ; 15: *piston    piston79
    16: *alt_action_input Toe_Tutti
```

```
   17: *piston     Toe_2
   18: *piston     Toe_Gen_9
   19: *piston     Toe_Gen_8
   20: *piston     Toe_Gen_6
   21: *piston     Toe_Gen_11
   22: *piston     Toe_Gen_12
   23: *piston     Toe_Gen_7
   24: *reversible pedBombarde32
   25: *piston     Toe_Gen_10
  ; 26: *piston    piston90
   27: *alt_action_input Toe_Nave
   28: *piston     Toe_Gen_3
   29: *piston     Toe_Gen_4
 ;  30: *piston    piston94
 ;  31: *piston    piston95
   32: *piston     Toe_Gen_2
   33: *piston     Toe_Gen_1
   34: *piston     Toe_Gen_5
   35: *reversible pos2ped8
 ;  36: *piston    piston100
 ;  37: *piston    piston101
 ;  38: *piston    piston102
 ;  39: *piston    piston103
 ;  40: *piston    piston104

 *hv64
 1:  *division Pedal '8' 32

*port 5
 *rank_driver
1: *mag_on_off swTremulant
3: *mag_on_off swChalumeau8
5: *mag_on_off swHautbois8
7: *mag_on_off swPleinJeuIV
9: *mag_on_off swNasard223
11: *mag_on_off swPrincipal4
13: *mag_on_off swViola8
15: *mag_on_off swSwell16
17: *mag_on_off swVox8
19: *mag_on_off swBasson16
21: *mag_on_off swBlockflote2
23: *mag_on_off swWaldflote4
25: *mag_on_off swViolaceleste8
27: *mag_on_off swRohrflote8
29: *mag_on_off swSwell4
31: *mag_on_off swClairon4
33: *mag_on_off swTrompette8
35: *mag_on_off swTierce135
37: *mag_on_off swHohlflote4
39: *mag_on_off swFlautoceleste8
41: *mag_on_off swFlauto8
43: *mag_on_off swRohrbass16
```

```
     45: *mag_on_off pedHautbois8
     47: *mag_on_off pedBasson16
     49: *mag_on_off pedRohrflote4
     51: *mag_on_off pedGemshorn8
     53: *mag_on_off pedErzahler16
     55: *mag_on_off pedGemshorn16
     57: *mag_on_off pedFagot4
     59: *mag_on_off pedBombarde16
     61: *mag_on_off pedMixtureIV
     63: *mag_on_off pedRohrflote8
     65: *mag_on_off pedOctave8
     67: *mag_on_off pedRohrbass16
     69: *mag_on_off pedViolone32
     71: *mag_on_off pedBombarde8
     73: *mag_on_off pedBombarde32
     75: *mag_on_off pedChoralbass4

    *port 6
      *rank_driver
      1:  *rank octave 61
    *port 7
      *rank_driver
      1:  *rank fifteenth 61
    *port 8
      *rank_driver
      1:  *rank oboe 61

*eight_port 2
     *port 1
     *hv64
    1: *dual_mag_tab gGreat4
    2: *dual_mag_tab gBasson16
    3: *dual_mag_tab gMixtureIV
    4: *dual_mag_tab gGedeckt4
    5: *dual_mag_tab gBordun8
    6: *dual_mag_tab gGemshorn8
    7: *dual_mag_tab gTrompete8
    8: *dual_mag_tab gCymbale
    9: *dual_mag_tab gSuperoctave2
    10: *dual_mag_tab gOvtave4
    11: *dual_mag_tab gPrincipal8
    12: *dual_mag_tab gGemshorn16
    13: *dual_mag_tab gChimes
    14: *dual_mag_tab pPositif4
    15: *dual_mag_tab pPositif16
    16: *dual_mag_tab pCromorne8
    17: *dual_mag_tab pClochesIII
    18: *dual_mag_tab pQuint113
    19: *dual_mag_tab pSpillflote4
    20: *dual_mag_tab pErzahlerceleste8
    21: *dual_mag_tab pHolzgedeckt8
    29: *dual_mag_tab pErzahler16
```

```
22: *dual_mag_tab pTremulant
23: *dual_mag_tab pCoranglais8
24: *dual_mag_tab pZimbelIV
25: *dual_mag_tab pSifflote1
26: *dual_mag_tab pKleinoctave2
27: *dual_mag_tab pOctave4
28: *dual_mag_tab pErzahler8
30: *dual_mag_tab pPrincipal8
31: *dual_mag_tab soTremulant
32: *dual_mag_tab soFestival4
33: *dual_mag_tab soFestival8
34: *dual_mag_tab soFestival16
35: *dual_mag_tab soDoppelflote4
36: *dual_mag_tab soDoppelflote8
37: *dual_mag_tab soCelesta4
38: *dual_mag_tab soZimbelstern
39: *dual_mag_tab soCarillon
40: *dual_mag_tab soSchalmey4
41: *dual_mag_tab soFrenchhorn8
42: *dual_mag_tab soSchalmey8
43: *dual_mag_tab soGambaceleste8
44: *dual_mag_tab soGamba
45: *dual_mag_tab soDoppelflote16
46: *dual_mag_tab soHarp8
47: *dual_mag_tab soChimes

*port 2
 *rank_driver
1: *mag_on_off gGreat4
3: *mag_on_off gBasson16
5: *mag_on_off gMixtureIV
7: *mag_on_off gGedeckt4
9: *mag_on_off gBordun8
11: *mag_on_off gGemshorn8
13: *mag_on_off gTrompete8
15: *mag_on_off gCymbale
17: *mag_on_off gSuperoctave2
19: *mag_on_off gOvtave4
21: *mag_on_off gPrincipal8
23: *mag_on_off gGemshorn16
25: *mag_on_off gChimes
27: *mag_on_off pPositif4
29: *mag_on_off pPositif16
31: *mag_on_off pCromorne8
33: *mag_on_off pClochesIII
35: *mag_on_off pQuint113
37: *mag_on_off pSpillflote4
39: *mag_on_off pErzahlerceleste8
41: *mag_on_off pHolzgedeckt8
43: *mag_on_off pErzahler16
45: *mag_on_off pTremulant
47: *mag_on_off pCoranglais8
```

```
49: *mag_on_off pZimbelIV
51: *mag_on_off pSifflote1
53: *mag_on_off pKleinoctave2
55: *mag_on_off pOctave4
57: *mag_on_off pErzahler8
59: *mag_on_off pPrincipal8

*port 3
 *rank_driver
1: *mag_on_off soTremulant
3: *mag_on_off soFestival4
5: *mag_on_off soFestival8
7: *mag_on_off soFestival16
9: *mag_on_off soDoppelflote4
11: *mag_on_off soDoppelflote8
13: *mag_on_off soCelesta4
15: *mag_on_off soZimbelstern
17: *mag_on_off soCarillon
19: *mag_on_off soSchalmey4
21: *mag_on_off soFrenchhorn8
23: *mag_on_off soSchalmey8
25: *mag_on_off soGambaceleste8
27: *mag_on_off soGamba
29: *mag_on_off soDoppelflote16
31: *mag_on_off soHarp8
33: *mag_on_off soChimes
35: *lamp (t_cresc)
36: *lamp (sfz)

*port 4
 *hv64
1: *dual_mag_tab swTremulant
2: *dual_mag_tab swChalumeau8
3: *dual_mag_tab swHautbois8
4: *dual_mag_tab swPleinJeuIV
5: *dual_mag_tab swNasard223
6: *dual_mag_tab swPrincipal4
7: *dual_mag_tab swViola8
8: *dual_mag_tab swSwell16
9: *dual_mag_tab swVox8
10: *dual_mag_tab swBasson16
11: *dual_mag_tab swBlockflote2
12: *dual_mag_tab swWaldflote4
13: *dual_mag_tab swViolaceleste8
14: *dual_mag_tab swRohrflote8
15: *dual_mag_tab swSwell4
16: *dual_mag_tab swClairon4
17: *dual_mag_tab swTrompette8
18: *dual_mag_tab swTierce135
19: *dual_mag_tab swHohlflote4
20: *dual_mag_tab swFlautoceleste8
21: *dual_mag_tab swFlauto8
```

```
 22: *dual_mag_tab swRohrbass16
 23: *dual_mag_tab pedHautbois8
 24: *dual_mag_tab pedBasson16
 25: *dual_mag_tab pedRohrflote4
 26: *dual_mag_tab pedGemshorn8
 27: *dual_mag_tab pedErzahler16
 28: *dual_mag_tab pedGemshorn16
 29: *dual_mag_tab pedFagot4
 30: *dual_mag_tab pedBombarde16
 31: *dual_mag_tab pedMixtureIV
 32: *dual_mag_tab pedRohrflote8
 33: *dual_mag_tab pedOctave8
 34: *dual_mag_tab pedRohrbass16
 35: *dual_mag_tab pedViolone32
 36: *dual_mag_tab pedBombarde8
 37: *dual_mag_tab pedBombarde32
 38: *dual_mag_tab pedChoralbass4
 39: *dual_mag_tab pedBourdon8
 40: *dual_mag_tab pedSubbass16
 41: *dual_mag_tab pedPrincipal16
 42: *dual_mag_tab g2ped8
 43: *dual_mag_tab sw2ped8
 44: *dual_mag_tab sw2ped4
 45: *dual_mag_tab pos2ped8
 46: *dual_mag_tab pos2ped4
 47: *dual_mag_tab so2ped8
 48: *dual_mag_tab sw2gt16
 49: *dual_mag_tab sw2gt8
 50: *dual_mag_tab sw2gt4
 51: *dual_mag_tab pos2gt16
 52: *dual_mag_tab pos2gt8
 53: *dual_mag_tab pos2gt4
 54: *dual_mag_tab gt2pos8
 55: *dual_mag_tab sw2pos16
 56: *dual_mag_tab sw2pos8
 57: *dual_mag_tab sw2pos4
 58: *dual_mag_tab so2gt8
 59: *dual_mag_tab so2sw8
 60: *dual_mag_tab so2pos8
 61: *dual_mag_tab so2pos4

*port 6
  *rank_driver

*port 7
  *rank_driver
 1: *mag_on_off pedBourdon8
 3: *mag_on_off pedSubbass16
 5: *mag_on_off pedPrincipal16
 7: *mag_on_off g2ped8
 9: *mag_on_off sw2ped8
 11: *mag_on_off sw2ped4
```

```
     13: *mag_on_off pos2ped8
     15: *mag_on_off pos2ped4
     17: *mag_on_off so2ped8
     19: *mag_on_off sw2gt16
     21: *mag_on_off sw2gt8
     23: *mag_on_off sw2gt4
     25: *mag_on_off pos2gt16
     27: *mag_on_off pos2gt8
     29: *mag_on_off pos2gt4
     31: *mag_on_off gt2pos8
     33: *mag_on_off sw2pos16
     35: *mag_on_off sw2pos8
     37: *mag_on_off sw2pos4
     39: *mag_on_off so2gt8
     41: *mag_on_off so2sw8
     43: *mag_on_off so2pos8
     45: *mag_on_off so2pos4

    *midi_port 8
      *midi_out
        c5n36 Solo
        c4n36 Pedal
        c3n36 Positiv
        c2n36 Swell
        c1n36 Great

c15n1
        gGreat4        gBasson16        gMixtureIV
gGedeckt4
        gBordun8        gGemshorn8        gTrompete8    gCymbale
        gSuperoctave2   gOvtave4          gPrincipal8   gGemshorn16
        gChimes        pPositif4        pPositif16      pCromorne8
        pClochesIII     pQuint113 pSpillflote4 pErzahlerceleste8
        pHolzgedeckt8   pErzahler16             pTremulant
                    pCoranglais8
        pZimbelIV       pSifflote1           pKleinoctave2
pOctave4
        pErzahler8      pPrincipal8          soTremulant
soFestival4
        soFestival8     soFestival16        soDoppelflote4
soDoppelflote8
        soCelesta4      soZimbelstern       soCarillon
soSchalmey4
        soFrenchhorn8   soSchalmey8         soGambaceleste8     soGamba
        soDoppelflote16 soHarp8             soChimes

 c16n1
        swTremulant     swChalumeau8        swHautbois8
      swPleinJeuIV
        swNasard223     swPrincipal4        swViola8
      swSwell16
```

```
        swVox8          swBasson16              swBlockflote2
      swWaldflote4
        swViolaceleste8 swRohrflote8        swSwell4
      swClairon4
        swTrompette8    swTierce135         swHohlflote4
      swFlautoceleste8
        swFlauto8       swRohrbass16        pedHautbois8
      pedBasson16
        pedRohrflote4   pedGemshorn8        pedErzahler16
      pedGemshorn16
        pedFagot4       pedBombarde16       pedMixtureIV
      pedRohrflote8
        pedOctave8      pedRohrbass16       pedViolone32
      pedBombarde8
        pedBombarde32   pedChoralbass4      pedBourdon8
      pedSubbass16
        pedPrincipal16  g2ped8                  sw2ped8
      sw2ped4
        pos2ped8        pos2ped4          so2ped8
      sw2gt16
        sw2gt8          sw2gt4                  pos2gt16
      pos2gt8
        pos2gt4        gt2pos8                 sw2pos16          sw2pos8
        sw2pos4        so2gt8                  so2sw8
      so2pos8
        so2pos4

c7c7 swellvol
c8c7 positifvol
c9c7 solovol
c7c80 t_cresc

*increment_decrement mem_level _function3 _function2 0 9 *wrap

*combination_action pset pmap 10 mem_level

*crescendo pset pmap 16 t_cresc

*sforzando pset pmap sfz

}   ; end of computer definition
 *coupler 1 Solo to Solo 0
*coupler 1 Swell to Swell 0
*coupler 1 Great to Great 0
*coupler 1 Positiv to Positiv 0
*coupler 1 Pedal to Pedal 0

*stop 1 Solo oboe '8'

*end
```

Virtual Theatre Pipe Organ

```
; 3 manual Virtual Theatre Pipe Organ.


*computer 50 "Console" {

 *analog  1 hw4_volume 0 127
 *analog  2 mainswell 0 127
 *analog  3 soloswell 0 127
 *analog  4 crescendo 0 127

*eight_port 1
*port 1
 *ui

;****************************
;  DISPLAY SECTION
;****************************

*lcd

*unsigned    *decimal 2
*line  1 "Mem=" (memory)
*signed  *decimal 1
*if(transpose!=0)  " Tr=" (transpose)  *else  " TR=" (transpose) *endif

*if(gen_pianosustain)      " PS"   *else  "   "  *endif
*if(sol_lieblichflute_4)   " L4 "  *else  "   "  *endif

*line   2
 *if(sol_strings_4)     "S4 "  *else   "   "  *endif
 *if(sol_voxes_4)       "V4 "  *else   "   "  *endif
 *if(sol_lieblichflute_223)  "23 "  *else   "   "  *endif
 *if(sol_lieblichflute_2)    "L2 "  *else   "   "  *endif
 *if(sol_voxdolce_8)        "VD "  *else   "   "  *endif
 *if(sol_sleighbells)       "SB "  *else   "   "  *endif
 *if(gen_string_to_flauto)   "SF"   *else   "   "  *endif

; Boilerplate to make the display menus work
*if (_sbusy)
 *line 3 "~1"
 *line 4 "~2"
*else

 *line  3
       *if(gen_swellmaster)  "SM "   *else  "   " *endif
       *if(gen_airnoise)     "AN "   *else  "   " *endif
       *if(gen_dampers)      "DM "   *else  "   " *endif
       *if(acc_frenchhorn_8) "FH "  *else  "   "  *endif
       *if(acc_maracas)      "MA "  *else  "   "  *endif
```

```
       *if(acc_sandblock)    "SA "  *else  "   "  *endif
       *if(ped_stringbass_8) "ST "   *else  "   " *endif

; toggle line 4
      *line  4
       *if (owen_jones)
            *if(ped_tibiakinura_8) "PK "  *else  "   "   *endif
            *if(ped_timpani) "TM "        *else  "   "   *endif
            *if(ped_timpani_reit) "TR "   *else  "   "   *endif
            *if(ped_bassdrum_roll) "BR "  *else  "   "   *endif
            *if(ped_templeblock) "TB "    *else  "   "   *endif
            *if(gen_percreit) "RE~b"      *else  "~b"    *endif
      *else
            *if (set_button)
                  *if (map_button)
                        "SET/MAP~b"
                  *else
                        "SET PISTON~b"
                  *endif
            *else
                  *if (map_button)
                        "MAP DISPLAY~b"
                  *else
                        "piston: ~p"
                  *endif
            *endif
      *endif
*endif      ; Terminate the boilerplate

*midi_sequencer
 c1n36 ped
 c2n36 acc
 c3n36 grt
 c4n36 solo
 c5n36 acc2nd
 c6n36 grt2nd
 c16c7 hw4_volume
 c8c7 mainswell
 c9c7 soloswell
 c10c2 crescendo
c12c17 transpose

c11n1
      ped_contradiaphone_32 ped_contraviolone_32 ped_posthorn_16
ped_contrabourdon_32
      ped_tubahorn_16 ped_diaphone_16 ped_diaphonichorn_16
ped_tibiaclausa_s_16
      ped_violone_16 ped_oboehorn_16 ped_bourdon_16 ped_posthorn_8
      ped_tubahorn_8 ped_opendiapason_8 ped_horndiapason_8
ped_tibiaclausa_s_8
      ped_tibiapizz_8 ped_clarinet_8 ped_salicional_8 ped_concertflute_8
      ped_piano_16 ped_piano_8 ped_accomp_to_pedal ped_great_to_pedal
```

```
      ped_solo_to_pedal acc_tubamirabilis_8 acc_posthorn_8
acc_brasstrumpet_8
      acc_tubahorn_8 acc_diaphone_8 acc_opendiapason_8
acc_horndiapason_8
      acc_tibiaclausa_m_8 acc_clarinet_8 acc_oboehorn_8 acc_solostring_8
      acc_violdorch_8 acc_salicional_8 acc_quintadena_8
acc_lieblichflute_8
      acc_concertflute_8 acc_voxhumana_s_8 acc_voxhumana_m_8 acc_dolce_8
      acc_opendiapason_4 acc_octavehorn_4 acc_tibiaclausa_m_4
acc_solostring_4
      acc_violdorch_4 acc_salicional_4 acc_quintadena_4
acc_lieblichflute_4
      acc_concertflute_4 acc_voxhumana_m_4 acc_dolce_4
acc_twelfthflute_223
      acc_tibiaclausa_m_2 acc_concertflute_2 acc_piano_8 acc_marimba_sub
      acc_marimba acc_chrysoglott  acc_solo_to_accomp
grt_tubamirabilis_16
c11n65
      acc_orchharp acc_accompoctave_4 grt_posthorn_16
grt_brasstrumpet_16
      grt_trumpetd_16 grt_tubahorn_16 grt_opendiapason_16
grt_diaphonichorn_16
      grt_tibiaclausa_s_16 grt_tibiaclausa_m_16 grt_clarinet_16
grt_krumet_16
      grt_orchoboe_16 grt_musette_16 grt_saxophone_16 grt_solostring_16
      grt_stringsiv_16 grt_oboehorn_16 grt_voxhumana_s_16
grt_voxhumana_m_16
      grt_tubamirabilis_8 grt_posthorn_8 grt_brasstrumpet_8
grt_frenchtrumpet_8
      grt_trumpetd_8 grt_tubahorn_8 grt_opendiapason_8
grt_horndiapason_8
      grt_tibiaclausa_s_8 grt_tibiaclausa_m_8 grt_clarinet_8
grt_krumet_8
      grt_kinura_8 grt_orchoboe_8 grt_musette_8 grt_saxophone_8
      grt_orchstrings_8 grt_solostring_8 grt_violdorch_8
grt_salicional_8
      grt_frenchhorn_8 grt_oboehorn_8 grt_spitzflute_8 grt_quintadena_8
      grt_lieblichflute_8 grt_concertflute_8 grt_voxhumana_s_8
grt_voxhumana_m_8
      grt_dulciana_8 grt_tibiaclausa_m_513 grt_opendiapason_4
grt_tibiaclausa_s_4
      grt_tibiaclausa_m_4 grt_solostring_4 grt_violdorch_4
grt_salicional_4
      grt_lieblichflute_4 grt_concertflute_4 grt_voxhumana_s_4
grt_dulcet_4
      grt_tibiaclausa_m_315 grt_tibiaclausa_m_223 grt_twelfthflute_223
      c12n1
      grt_tibiaclausa_s_2
c12n2
      grt_tibiaclausa_m_2 grt_viol_2 grt_concertflute_2
grt_tibiaclausa_s_135
      grt_tibiaclausa_s_1 grt_piano_16 grt_piano_8 grt_harp_sub
```

```
c13n91
      ped_templeblock gen_percreit sol_lieblichflute_4 sol_strings_4
      sol_voxes_4 sol_lieblichflute_223 sol_lieblichflute_2
sol_voxdolce_8
c13n99
      sol_sleighbells gen_string_to_flauto gen_swellmaster gen_airnoise
      gen_dampers gen_pianosustain owen_jones aux_ls3_7

*port 2
*input64
01: *dual_mag_tab ped_contradiaphone_32
02: *dual_mag_tab ped_contraviolone_32
03: *dual_mag_tab ped_posthorn_16
04: *dual_mag_tab ped_contrabourdon_32
05: *dual_mag_tab ped_tubahorn_16
06: *dual_mag_tab ped_diaphone_16
07: *dual_mag_tab ped_diaphonichorn_16
08: *dual_mag_tab ped_tibiaclausa_s_16
09: *dual_mag_tab ped_violone_16
10: *dual_mag_tab ped_oboehorn_16
11: *dual_mag_tab ped_bourdon_16
12: *dual_mag_tab ped_posthorn_8
13: *dual_mag_tab ped_tubahorn_8
14: *dual_mag_tab ped_opendiapason_8
15: *dual_mag_tab ped_horndiapason_8
16: *dual_mag_tab ped_tibiaclausa_s_8
17: *dual_mag_tab ped_tibiapizz_8
18: *dual_mag_tab ped_clarinet_8
19: *dual_mag_tab ped_salicional_8
20: *dual_mag_tab ped_concertflute_8
21: *dual_mag_tab ped_piano_16
22: *dual_mag_tab ped_piano_8
23: *dual_mag_tab ped_accomp_to_pedal
24: *dual_mag_tab ped_great_to_pedal
25: *dual_mag_tab ped_solo_to_pedal
26: *dual_mag_tab acc_tubamirabilis_8
27: *dual_mag_tab acc_posthorn_8
28: *dual_mag_tab acc_brasstrumpet_8
29: *dual_mag_tab acc_tubahorn_8
30: *dual_mag_tab acc_diaphone_8
31: *dual_mag_tab acc_opendiapason_8
32: *dual_mag_tab acc_horndiapason_8
33: *dual_mag_tab acc_tibiaclausa_m_8
34: *dual_mag_tab acc_clarinet_8
35: *dual_mag_tab acc_oboehorn_8
36: *dual_mag_tab acc_solostring_8
37: *dual_mag_tab acc_violdorch_8
38: *dual_mag_tab acc_salicional_8
39: *dual_mag_tab acc_quintadena_8
40: *dual_mag_tab acc_lieblichflute_8
41: *dual_mag_tab acc_concertflute_8
42: *dual_mag_tab acc_voxhumana_s_8
```

```
43: *dual_mag_tab acc_voxhumana_m_8
44: *dual_mag_tab acc_dolce_8
45: *dual_mag_tab acc_opendiapason_4
46: *dual_mag_tab acc_octavehorn_4
47: *dual_mag_tab acc_tibiaclausa_m_4
48: *dual_mag_tab acc_solostring_4
49: *dual_mag_tab acc_violdorch_4
50: *dual_mag_tab acc_salicional_4
51: *dual_mag_tab acc_quintadena_4
52: *dual_mag_tab acc_lieblichflute_4
53: *dual_mag_tab acc_concertflute_4
54: *dual_mag_tab acc_voxhumana_m_4
55: *dual_mag_tab acc_dolce_4
56: *dual_mag_tab acc_twelfthflute_223
57: *dual_mag_tab acc_tibiaclausa_m_2
58: *dual_mag_tab acc_concertflute_2
59: *dual_mag_tab acc_piano_8
60: *dual_mag_tab acc_marimba_sub
61: *dual_mag_tab acc_marimba
62: *dual_mag_tab acc_chrysoglott
63: *dual_mag_tab acc_solo_to_accomp
64: *dual_mag_tab grt_tubamirabilis_16

*input64
01: *dual_mag_tab acc_orchharp
02: *dual_mag_tab acc_accompoctave_4
03: *dual_mag_tab grt_posthorn_16
04: *dual_mag_tab grt_brasstrumpet_16
05: *dual_mag_tab grt_trumpetd_16
06: *dual_mag_tab grt_tubahorn_16
07: *dual_mag_tab grt_opendiapason_16
08: *dual_mag_tab grt_diaphonichorn_16
09: *dual_mag_tab grt_tibiaclausa_s_16
10: *dual_mag_tab grt_tibiaclausa_m_16
11: *dual_mag_tab grt_clarinet_16
12: *dual_mag_tab grt_krumet_16
13: *dual_mag_tab grt_orchoboe_16
14: *dual_mag_tab grt_musette_16
15: *dual_mag_tab grt_saxophone_16
16: *dual_mag_tab grt_solostring_16
17: *dual_mag_tab grt_stringsiv_16
18: *dual_mag_tab grt_oboehorn_16
19: *dual_mag_tab grt_voxhumana_s_16
20: *dual_mag_tab grt_voxhumana_m_16
21: *dual_mag_tab grt_tubamirabilis_8
22: *dual_mag_tab grt_posthorn_8
23: *dual_mag_tab grt_brasstrumpet_8
24: *dual_mag_tab grt_frenchtrumpet_8
25: *dual_mag_tab grt_trumpetd_8
26: *dual_mag_tab grt_tubahorn_8
27: *dual_mag_tab grt_opendiapason_8
28: *dual_mag_tab grt_horndiapason_8
```

```
29: *dual_mag_tab grt_tibiaclausa_s_8
30: *dual_mag_tab grt_tibiaclausa_m_8
31: *dual_mag_tab grt_clarinet_8
32: *dual_mag_tab grt_krumet_8
33: *dual_mag_tab grt_kinura_8
34: *dual_mag_tab grt_orchoboe_8
35: *dual_mag_tab grt_musette_8
36: *dual_mag_tab grt_saxophone_8
37: *dual_mag_tab grt_orchstrings_8
38: *dual_mag_tab grt_solostring_8
39: *dual_mag_tab grt_violdorch_8
40: *dual_mag_tab grt_salicional_8
41: *dual_mag_tab grt_frenchhorn_8
42: *dual_mag_tab grt_oboehorn_8
43: *dual_mag_tab grt_spitzflute_8
44: *dual_mag_tab grt_quintadena_8
45: *dual_mag_tab grt_lieblichflute_8
46: *dual_mag_tab grt_concertflute_8
47: *dual_mag_tab grt_voxhumana_s_8
48: *dual_mag_tab grt_voxhumana_m_8
49: *dual_mag_tab grt_dulciana_8
50: *dual_mag_tab grt_tibiaclausa_m_513
51: *dual_mag_tab grt_opendiapason_4
52: *dual_mag_tab grt_tibiaclausa_s_4
53: *dual_mag_tab grt_tibiaclausa_m_4
54: *dual_mag_tab grt_solostring_4
55: *dual_mag_tab grt_violdorch_4
56: *dual_mag_tab grt_salicional_4
57: *dual_mag_tab grt_lieblichflute_4
58: *dual_mag_tab grt_concertflute_4
59: *dual_mag_tab grt_voxhumana_s_4
60: *dual_mag_tab grt_dulcet_4
61: *dual_mag_tab grt_tibiaclausa_m_315
62: *dual_mag_tab grt_tibiaclausa_m_223
63: *dual_mag_tab grt_twelfthflute_223
64: *dual_mag_tab grt_tibiaclausa_s_2

*port 3
*hv64
01: *dual_mag_tab grt_tibiaclausa_m_2
02: *dual_mag_tab grt_viol_2
03: *dual_mag_tab grt_concertflute_2
04: *dual_mag_tab grt_tibiaclausa_s_135
05: *dual_mag_tab grt_tibiaclausa_s_1
06: *dual_mag_tab grt_piano_16
07: *dual_mag_tab grt_piano_8
08: *dual_mag_tab grt_harp_sub
09: *dual_mag_tab grt_xylophone_sub
10: *dual_mag_tab grt_chrysoglott_sub
11: *dual_mag_tab grt_glockenspiel
12: *dual_mag_tab grt_suboctave_16
13: *dual_mag_tab grt_unison_off_8
```

```
14: *dual_mag_tab grt_octave_4
15: *dual_mag_tab grt_solo_2_grt_16
16: *dual_mag_tab grt_solo_2_grt_8
17: *dual_mag_tab grt_solo_2_grtpizz_8
18: *dual_mag_tab grt_solo_2_grtmelody_8
19: *dual_mag_tab sol_posthorn_16
20: *dual_mag_tab sol_tubamirabilis_8
21: *dual_mag_tab sol_posthorn_8
22: *dual_mag_tab sol_brasstrumpet_8
23: *dual_mag_tab sol_frenchtrumpet_8
24: *dual_mag_tab sol_trumpetd_8
25: *dual_mag_tab sol_tubahorn_8
26: *dual_mag_tab sol_diaphone_8
27: *dual_mag_tab sol_opendiapason_8
28: *dual_mag_tab sol_tibiaclausa_s_8
29: *dual_mag_tab sol_tibiaclausa_m_8
30: *dual_mag_tab sol_clarinet_8
31: *dual_mag_tab sol_krumet_8
32: *dual_mag_tab sol_kinura_8
33: *dual_mag_tab sol_orchoboe_8
34: *dual_mag_tab sol_musette_8
35: *dual_mag_tab sol_saxophone_8
36: *dual_mag_tab sol_gambas_8
37: *dual_mag_tab sol_stringsiv_8
38: *dual_mag_tab sol_frenchhorn_8
39: *dual_mag_tab sol_lieblichflute_8
40: *dual_mag_tab sol_quintadena_8
41: *dual_mag_tab sol_voxhumana_s_8
42: *dual_mag_tab sol_tibiaclausa_s_4
43: *dual_mag_tab sol_tibiaclausa_m_4
44: *dual_mag_tab sol_harmonicflute_4
45: *dual_mag_tab sol_tibiaclausa_s_223
46: *dual_mag_tab sol_tibiaclausa_s_2
47: *dual_mag_tab sol_tibiaclausa_s_135
48: *dual_mag_tab sol_tibiaclausa_s_113
49: *dual_mag_tab sol_tibiaclausa_s_1
50: *dual_mag_tab sol_piano_8
51: *dual_mag_tab sol_harp_sub
52: *dual_mag_tab sol_vibraphone
53: *dual_mag_tab sol_orchharp
54: *dual_mag_tab sol_chrysoglott_sub
55: *dual_mag_tab sol_xylophone
56: *dual_mag_tab sol_chrysoglott
57: *dual_mag_tab sol_orchbell
58: *dual_mag_tab sol_chimes
59: *dual_mag_tab sol_suboctave_16
60: *dual_mag_tab sol_unison_off_8
61: *dual_mag_tab sol_octave_4
62: *dual_mag_tab sol_superoctave_2
63: *dual_mag_tab sol_seventeenth_135
64: *dual_mag_tab grt_bourdon_16
```

```
*port 4
*input64
01: *input gen_cymbal
02: *input gen_gong
03: *input gen_reset
04: *input gen_belltree
05: *input gen_birds
06: *input gen_trainwhistle
07: *input gen_noveltywhistle
08: *input clear_memory
09: *input clear_transpose
10: *input gen_cymbalroll
11: *input gen_toggle

*port 5
*input64
01: *dual_mag_tab gen_percunexpressed
02: *dual_mag_tab gen_stringscelestes_off
03: *dual_mag_tab gen_generalcelestes_on
04: *dual_mag_tab gen_lieblichflute_off
05: *dual_mag_tab gen_tibia_minor_on
06: *dual_mag_tab acc2t_tunedperc_to_acc2t
07: *dual_mag_tab acc2t_posthorn_8
08: *dual_mag_tab acc2t_brasstrumpet_8
09: *dual_mag_tab acc2t_trumpetd_8
10: *dual_mag_tab acc2t_tubahorn_8
11: *dual_mag_tab acc2t_diaphone_8
12: *dual_mag_tab acc2t_tibiaclausa_s_8
13: *dual_mag_tab acc2t_clarinet_8
14: *dual_mag_tab acc2t_tibiaclausa_s_4
15: *dual_mag_tab acc2t_piano_8
16: *dual_mag_tab acc2t_harp_sub
17: *dual_mag_tab acc2t_chrysoglott_sub
18: *dual_mag_tab acc2t_glockenspiel_octave
19: *dual_mag_tab acc2t_solo_to_acc2t
20: *dual_mag_tab acc2t_greatoctave_to_acc2t
21: *dual_mag_tab acc2t_traps_to_acc2t
22: *dual_mag_tab grt2t_solosub_to_grt2t
23: *dual_mag_tab grt2t_solo_to_grt2t
24: *dual_mag_tab grt_sostenuto
25: *dual_mag_tab grt_great_to_solo
26: *dual_mag_tab grt_greatoctave_to_solo
27: *dual_mag_tab ped_bassdrum
28: *dual_mag_tab ped_cymbal
29: *dual_mag_tab ped_brushcymbal
30: *dual_mag_tab ped_jazzcymbal
31: *dual_mag_tab ped_fingercymbal
32: *dual_mag_tab ped_triangle
33: *dual_mag_tab acc_snaredrum
34: *dual_mag_tab acc_rimshot
35: *dual_mag_tab acc_cymbal
36: *dual_mag_tab acc_brushcymbal
```

```
37: *dual_mag_tab acc_jazzcymbal
38: *dual_mag_tab acc_openhihats
39: *dual_mag_tab acc_tomtom
40: *dual_mag_tab acc_cowbell
41: *dual_mag_tab acc_castanets
42: *dual_mag_tab acc_tambourine
43: *dual_mag_tab acc_woodblock
44: *dual_mag_tab trems_main
45: *dual_mag_tab trems_solo
46: *dual_mag_tab trems_tibia
47: *dual_mag_tab trems_clarinet
48: *dual_mag_tab trems_strings
49: *dual_mag_tab trems_tubah_trumd
50: *dual_mag_tab trems_brasstrumpet
51: *dual_mag_tab trems_tubamirabilis
52: *dual_mag_tab trems_vibes_chrysoglott

*port 6
*hv64
01: *piston gen_6
02: *piston gen_7
03: *piston gen_8
04: *piston gen_9
05: *piston gen_10
06: *piston great_pp
07: *piston great_p
08: *piston great_mf
09: *piston great_f
10: *piston great_ff
11: *piston great_1
12: *piston great_2
13: *piston great_3
14: *piston great_4
15: *piston great_5
16: *piston great_6
17: *piston great_7
18: *piston great_8
19: *piston great_9
20: *piston great_10
21: *piston great_11
22: *piston great_12
23: *piston great_13
24: *piston greathwcancel15
25: *piston great_cancel
26: *input gen_memoryminus
27: *input gen_memoryplus
28: *piston gen_1
29: *piston gen_2
30: *piston gen_3
31: *piston gen_4
32: *piston gen_5
33: *piston acc_pp
```

```
34: *piston acc_p
35: *piston acc_m
36: *piston acc_f
37: *piston acc_ff
38: *input gen_transposeminus
39: *input gen_transposeplus
40: *input map_button
41: *input set_button
42: *piston gen_11
43: *piston gen_12
44: *piston gen_13
45: *input_auto_save
;45: *piston gen_14
46: *save_combination_action
;46: *piston gen_15
47: *piston solo_pp
48: *piston solo_p
49: *piston solo_mf
50: *piston solo_f
51: *piston solo_ff
52: *piston trems_on
53: *piston trems_off
54: *piston acc_3
55: *piston acc_4
56: *piston acc_5
57: *piston acc_6
58: *piston acc_7
59: *piston acc_8
60: *piston acc_9
61: *piston acc_10
62: *piston acc_11
63: *piston acc_12
64: *cancel gen_cancel

; pedalboard
*port 7
*hv64
01: *division ped '16' 32
33: *piston solo_1
34: *piston solo_2
35: *piston solo_3
36: *piston solo_4
37: *piston solo_5
38: *piston solo_6
39: *piston solo_7
40: *piston solo_8
41: *piston solo_9
42: *piston solo_10
43: *piston solo_11
44: *piston solo_12
45: *piston solo_13
46: *piston solo_14
```

```
47: *piston solo_cancel
48: *piston acc_13
49: *piston accomp_cancel

*port 8
*lighted_stop_8
01: *input acc_frenchhorn_8
02: *input acc_maracas
03: *input acc_sandblock
04: *input ped_stringbass_8
05: *input ped_tibiakinura_8
06: *input ped_timpani
07: *input ped_timpani_reit
08: *input ped_bassdrum_roll

*eight_port 2
*port 1
*rank_driver
01: *mag_on_off ped_contradiaphone_32
03: *mag_on_off ped_contraviolone_32
05: *mag_on_off ped_contrabourdon_32
07: *mag_on_off ped_posthorn_16
09: *mag_on_off ped_tubahorn_16
11: *mag_on_off ped_diaphone_16
13: *mag_on_off ped_diaphonichorn_16
15: *mag_on_off ped_tibiaclausa_s_16
17: *mag_on_off ped_violone_16
19: *mag_on_off ped_oboehorn_16
21: *mag_on_off ped_bourdon_16
23: *mag_on_off ped_posthorn_8
25: *mag_on_off ped_tubahorn_8
27: *mag_on_off ped_opendiapason_8
29: *mag_on_off ped_horndiapason_8
31: *mag_on_off ped_tibiaclausa_s_8
33: *mag_on_off ped_tibiapizz_8
35: *mag_on_off ped_clarinet_8
37: *mag_on_off ped_salicional_8
39: *mag_on_off ped_concertflute_8
41: *mag_on_off ped_piano_16
43: *mag_on_off ped_piano_8
45: *mag_on_off ped_accomp_to_pedal
47: *mag_on_off ped_great_to_pedal
49: *mag_on_off ped_solo_to_pedal
51: *mag_on_off acc_dolce_8
53: *mag_on_off acc_opendiapason_4
55: *mag_on_off acc_tibiaclausa_m_4
57: *mag_on_off acc_octavehorn_4
59: *mag_on_off acc_solostring_4
61: *mag_on_off acc_violdorch_4
63: *mag_on_off acc_salicional_4

*port 2
```

```
*rank_driver
01: *mag_on_off acc_tubamirabilis_8
03: *mag_on_off acc_posthorn_8
05: *mag_on_off acc_brasstrumpet_8
07: *mag_on_off acc_tubahorn_8
09: *mag_on_off acc_diaphone_8
11: *mag_on_off acc_opendiapason_8
13: *mag_on_off acc_horndiapason_8
15: *mag_on_off acc_tibiaclausa_m_8
17: *mag_on_off acc_clarinet_8
19: *mag_on_off acc_oboehorn_8
21: *mag_on_off acc_solostring_8
23: *mag_on_off acc_violdorch_8
25: *mag_on_off acc_salicional_8
27: *mag_on_off acc_quintadena_8
29: *mag_on_off acc_lieblichflute_8
31: *mag_on_off acc_concertflute_8
33: *mag_on_off acc_voxhumana_s_8
35: *mag_on_off acc_voxhumana_m_8
37: *mag_on_off acc_quintadena_4
39: *mag_on_off  acc_lieblichflute_4
41: *mag_on_off acc_concertflute_4
43: *mag_on_off acc_voxhumana_m_4
45: *mag_on_off acc_dolce_4
47: *mag_on_off acc_twelfthflute_223
49: *mag_on_off acc_tibiaclausa_m_2
51: *mag_on_off acc_concertflute_2
53: *mag_on_off acc_piano_8
55: *mag_on_off acc_marimba_sub
57: *mag_on_off acc_marimba
59: *mag_on_off acc_chrysoglott
61: *mag_on_off acc_orchharp
63: *mag_on_off acc_accompoctave_4

*port 3
*rank_driver
01: *mag_on_off grt_tubamirabilis_16
03: *mag_on_off grt_posthorn_16
05: *mag_on_off grt_brasstrumpet_16
07: *mag_on_off grt_trumpetd_16
09: *mag_on_off grt_tubahorn_16
11: *mag_on_off grt_opendiapason_16
13: *mag_on_off grt_diaphonichorn_16
15: *mag_on_off grt_tibiaclausa_s_16
17: *mag_on_off grt_tibiaclausa_m_16
19: *mag_on_off grt_clarinet_16
21: *mag_on_off grt_krumet_16
23: *mag_on_off grt_orchoboe_16
25: *mag_on_off grt_musette_16
27: *mag_on_off grt_saxophone_16
29: *mag_on_off grt_solostring_16
31: *mag_on_off grt_stringsiv_16
```

```
33: *mag_on_off grt_oboehorn_16
35: *mag_on_off grt_bourdon_16
37: *mag_on_off grt_voxhumana_s_16
39: *mag_on_off grt_voxhumana_m_16
41: *mag_on_off grt_tubamirabilis_8
43: *mag_on_off grt_posthorn_8
45: *mag_on_off grt_brasstrumpet_8
47: *mag_on_off grt_frenchtrumpet_8
49: *mag_on_off grt_trumpetd_8
51: *mag_on_off grt_tubahorn_8
53: *mag_on_off grt_opendiapason_8
55: *mag_on_off grt_horndiapason_8
57: *mag_on_off acc_solo_to_accomp
59: *mag_on_off grt_voxhumana_s_8
61: *mag_on_off grt_voxhumana_m_8
63: *mag_on_off grt_dulciana_8

*port 4
*rank_driver
01: *mag_on_off grt_tibiaclausa_s_8
03: *mag_on_off grt_tibiaclausa_m_8
05: *mag_on_off grt_clarinet_8
07: *mag_on_off grt_krumet_8
09: *mag_on_off grt_kinura_8
11: *mag_on_off grt_orchoboe_8
13: *mag_on_off grt_musette_8
15: *mag_on_off grt_saxophone_8
17: *mag_on_off grt_orchstrings_8
19: *mag_on_off grt_solostring_8
21: *mag_on_off grt_violdorch_8
23: *mag_on_off grt_salicional_8
25: *mag_on_off grt_frenchhorn_8
27: *mag_on_off grt_oboehorn_8
29: *mag_on_off grt_spitzflute_8
31: *mag_on_off grt_quintadena_8
33: *mag_on_off grt_lieblichflute_8
35: *mag_on_off grt_concertflute_8
37: *mag_on_off grt_tibiaclausa_m_513
39: *mag_on_off grt_opendiapason_4
41: *mag_on_off grt_tibiaclausa_s_4
43: *mag_on_off grt_tibiaclausa_m_4
45: *mag_on_off grt_solostring_4
47: *mag_on_off grt_violdorch_4
49: *mag_on_off grt_salicional_4
51: *mag_on_off grt_lieblichflute_4
53: *mag_on_off grt_concertflute_4
55: *mag_on_off grt_voxhumana_s_4
57: *mag_on_off grt_dulcet_4
59: *mag_on_off grt_tibiaclausa_m_315
61: *mag_on_off grt_tibiaclausa_m_223
63: *mag_on_off grt_twelfthflute_223
65: *mag_on_off grt_tibiaclausa_s_2
```

```
67: *mag_on_off grt_tibiaclausa_m_2
69: *mag_on_off grt_viol_2
71: *mag_on_off grt_concertflute_2
73: *mag_on_off grt_tibiaclausa_s_135
75: *mag_on_off grt_tibiaclausa_s_1
77: *mag_on_off grt_piano_16
79: *mag_on_off grt_piano_8

*port 5
*rank_driver
01: *mag_on_off grt_harp_sub
03: *mag_on_off grt_xylophone_sub
05: *mag_on_off grt_chrysoglott_sub
07: *mag_on_off grt_glockenspiel
09: *mag_on_off grt_suboctave_16
11: *mag_on_off grt_unison_off_8
13: *mag_on_off grt_octave_4
15: *mag_on_off grt_solo_2_grt_16
17: *mag_on_off grt_solo_2_grt_8
19: *mag_on_off grt_solo_2_grtpizz_8
21: *mag_on_off grt_solo_2_grtmelody_8
23: *mag_on_off sol_posthorn_16
25: *mag_on_off sol_tubamirabilis_8
27: *mag_on_off sol_posthorn_8
29: *mag_on_off sol_brasstrumpet_8
31: *mag_on_off sol_frenchtrumpet_8
33: *mag_on_off sol_trumpetd_8
35: *mag_on_off sol_tubahorn_8
37: *mag_on_off sol_diaphone_8
39: *mag_on_off sol_opendiapason_8
41: *mag_on_off sol_tibiaclausa_s_8
43: *mag_on_off sol_tibiaclausa_m_8
45: *mag_on_off sol_clarinet_8
47: *mag_on_off sol_krumet_8
49: *mag_on_off sol_kinura_8
51: *mag_on_off sol_orchoboe_8
53: *mag_on_off sol_musette_8
55: *mag_on_off sol_saxophone_8
57: *mag_on_off sol_gambas_8
59: *mag_on_off sol_stringsiv_8
61: *mag_on_off sol_frenchhorn_8
63: *mag_on_off sol_lieblichflute_8
*port 6
*rank_driver
01: *mag_on_off sol_harmonicflute_4
03: *mag_on_off sol_tibiaclausa_s_223
05: *mag_on_off sol_tibiaclausa_s_2
07: *mag_on_off sol_tibiaclausa_s_135
09: *mag_on_off sol_tibiaclausa_s_113
11: *mag_on_off sol_tibiaclausa_s_1
13: *mag_on_off sol_piano_8
15: *mag_on_off sol_harp_sub
```

```
17: *mag_on_off sol_vibraphone
19: *mag_on_off sol_orchharp
21: *mag_on_off sol_chrysoglott_sub
23: *mag_on_off sol_xylophone
25: *mag_on_off sol_chrysoglott
27: *mag_on_off sol_orchbell
29: *mag_on_off sol_chimes
31: *mag_on_off sol_suboctave_16
33: *mag_on_off sol_unison_off_8
35: *mag_on_off sol_octave_4
37: *mag_on_off sol_superoctave_2
39: *mag_on_off sol_seventeenth_135
41: *mag_on_off sol_quintadena_8
43: *mag_on_off sol_voxhumana_s_8
45: *mag_on_off sol_tibiaclausa_s_4
47: *mag_on_off sol_tibiaclausa_m_4

*port 7
*rank_driver
01: *mag_on_off trems_vibes_chrysoglott
03: *mag_on_off trems_tubamirabilis
05: *mag_on_off trems_brasstrumpet
07: *mag_on_off trems_tubah_trumd
09: *mag_on_off trems_strings
11: *mag_on_off trems_clarinet
13: *mag_on_off trems_tibia
15: *mag_on_off trems_solo
17: *mag_on_off trems_main
19: *mag_on_off acc_woodblock
21: *mag_on_off acc_tambourine
23: *mag_on_off acc_castanets
25: *mag_on_off acc_cowbell
27: *mag_on_off acc_tomtom
29: *mag_on_off acc_openhihats
31: *mag_on_off acc_jazzcymbal
33: *mag_on_off acc_brushcymbal
35: *mag_on_off acc_cymbal
37: *mag_on_off acc_rimshot
39: *mag_on_off acc_snaredrum
41: *mag_on_off ped_triangle
43: *mag_on_off ped_fingercymbal
45: *mag_on_off ped_jazzcymbal
47: *mag_on_off ped_brushcymbal
49: *mag_on_off ped_cymbal
51: *mag_on_off ped_bassdrum

*port 8
*rank_driver
01: *mag_on_off grt_greatoctave_to_solo
03: *mag_on_off grt_great_to_solo
05: *mag_on_off grt_sostenuto
07: *mag_on_off grt2t_solo_to_grt2t
```

```
09: *mag_on_off grt2t_solosub_to_grt2t
11: *mag_on_off acc2t_traps_to_acc2t
13: *mag_on_off acc2t_greatoctave_to_acc2t
15: *mag_on_off acc2t_solo_to_acc2t
17: *mag_on_off acc2t_glockenspiel_octave
19: *mag_on_off acc2t_chrysoglott_sub
21: *mag_on_off acc2t_harp_sub
23: *mag_on_off acc2t_piano_8
25: *mag_on_off acc2t_tibiaclausa_s_4
27: *mag_on_off acc2t_clarinet_8
29: *mag_on_off acc2t_tibiaclausa_s_8
31: *mag_on_off acc2t_diaphone_8
33: *mag_on_off acc2t_tubahorn_8
35: *mag_on_off acc2t_trumpetd_8
37: *mag_on_off acc2t_brasstrumpet_8
39: *mag_on_off acc2t_posthorn_8
41: *mag_on_off acc2t_tunedperc_to_acc2t
43: *mag_on_off gen_tibia_minor_on
45: *mag_on_off gen_lieblichflute_off
47: *mag_on_off gen_generalcelestes_on
49: *mag_on_off gen_stringscelestes_off
51: *mag_on_off gen_percunexpressed

;keyboards
*eight_port 3
*port 1
*hv64
01: *division solo '8' 61
*hv64
01: *division grt  '8' 61

*port 2
*hv64
01: *division acc  '8' 61

*port 3
*hv64
01: *division acc2nd '8' 61
*hv64
01: *division grt2nd '8' 61

*port 5
*rank_driver
01: *rank orchbells 61 ; *bits=1,61

*port 6
*lighted_stop_8
01: *input ped_templeblock
02: *input gen_percreit
03: *input sol_lieblichflute_4
04: *input sol_strings_4
05: *input sol_voxes_4
```

```
06: *input sol_lieblichflute_223
07: *input sol_lieblichflute_2
08: *input sol_voxdolce_8

*port 7
*lighted_stop_8
01: *input sol_sleighbells
02: *input gen_string_to_flauto
03: *input gen_swellmaster
04: *input gen_airnoise
05: *input gen_dampers
06: *input gen_pianosustain
07: *input owen_jones
08: *input aux_ls3_7

;xxxxxxxxxxxxxxxxxxxxxxxxx
; MIDI output declarations
;xxxxxxxxxxxxxxxxxxxxxxxxx
*midi_port 8
  *midi_out
        c16c7 hw4_volume
        c8c7 mainswell
        c9c7 soloswell
        c10c2 crescendo

c11n1
      ped_contradiaphone_32 ped_contraviolone_32 ped_posthorn_16
ped_contrabourdon_32
      ped_tubahorn_16 ped_diaphone_16 ped_diaphonichorn_16
ped_tibiaclausa_s_16
      ped_violone_16 ped_oboehorn_16 ped_bourdon_16 ped_posthorn_8
      ped_tubahorn_8 ped_opendiapason_8 ped_horndiapason_8
ped_tibiaclausa_s_8
      ped_tibiapizz_8 ped_clarinet_8 ped_salicional_8 ped_concertflute_8
      ped_piano_16 ped_piano_8 ped_accomp_to_pedal ped_great_to_pedal
      ped_solo_to_pedal acc_tubamirabilis_8 acc_posthorn_8
acc_brasstrumpet_8
      acc_tubahorn_8 acc_diaphone_8 acc_opendiapason_8
acc_horndiapason_8
      acc_tibiaclausa_m_8 acc_clarinet_8 acc_oboehorn_8 acc_solostring_8
      acc_violdorch_8 acc_salicional_8 acc_quintadena_8
acc_lieblichflute_8
      acc_concertflute_8 acc_voxhumana_s_8 acc_voxhumana_m_8 acc_dolce_8
      acc_opendiapason_4 acc_octavehorn_4 acc_tibiaclausa_m_4
acc_solostring_4
      acc_violdorch_4 acc_salicional_4 acc_quintadena_4
acc_lieblichflute_4
      acc_concertflute_4 acc_voxhumana_m_4 acc_dolce_4
acc_twelfthflute_223
      acc_tibiaclausa_m_2 acc_concertflute_2 acc_piano_8 acc_marimba_sub
      acc_marimba acc_chrysoglott  acc_solo_to_accomp
grt_tubamirabilis_16
```

```
      sol_octave_4 sol_superoctave_2 sol_seventeenth_135 grt_bourdon_16
c12n66
      gen_cymbal gen_gong gen_reset gen_belltree
      gen_birds gen_trainwhistle gen_noveltywhistle clear_memory
      clear_transpose gen_cymbalroll gen_toggle
c12n77
      gen_percunexpressed gen_stringscelestes_off gen_generalcelestes_on
gen_lieblichflute_off
      gen_tibia_minor_on acc2t_tunedperc_to_acc2t acc2t_posthorn_8
acc2t_brasstrumpet_8
      acc2t_trumpetd_8 acc2t_tubahorn_8 acc2t_diaphone_8
acc2t_tibiaclausa_s_8
      acc2t_clarinet_8 acc2t_tibiaclausa_s_4 acc2t_piano_8
acc2t_harp_sub
      acc2t_chrysoglott_sub acc2t_glockenspiel_octave
acc2t_solo_to_acc2t acc2t_greatoctave_to_acc2t
      acc2t_traps_to_acc2t grt2t_solosub_to_grt2t grt2t_solo_to_grt2t
grt_sostenuto
      grt_great_to_solo grt_greatoctave_to_solo ped_bassdrum ped_cymbal
      ped_brushcymbal ped_jazzcymbal ped_fingercymbal ped_triangle
      acc_snaredrum acc_rimshot acc_cymbal  acc_brushcymbal
      acc_jazzcymbal acc_openhihats acc_tomtom acc_cowbell
      acc_castanets  acc_tambourine  acc_woodblock trems_main
      trems_solo  trems_tibia  trems_clarinet  trems_strings
      trems_tubah_trumd trems_brasstrumpet trems_tubamirabilis
c13n1
      trems_vibes_chrysoglott
c12c17
      transpose
c13n83
      acc_frenchhorn_8 acc_maracas acc_sandblock ped_stringbass_8
      ped_tibiakinura_8 ped_timpani  ped_timpani_reit ped_bassdrum_roll
c13n91
      ped_templeblock gen_percreit sol_lieblichflute_4 sol_strings_4
      sol_voxes_4 sol_lieblichflute_223 sol_lieblichflute_2
sol_voxdolce_8
c13n99
      sol_sleighbells gen_string_to_flauto gen_swellmaster gen_airnoise
      gen_dampers gen_pianosustain owen_jones aux_ls3_7

*midi_div_out (1) ped c1n36 '8'
*midi_div_out (1) solo c4n36 '8'
*midi_div_out (1) grt c3n36 '8'
*midi_div_out (1) acc c2n36 '8'
*midi_div_out (1) acc2nd c5n36 '8'
*midi_div_out (1) grt2nd c6n36 '8'

*midi_command  (clear_memory) c12c56 0
*midi_command  (clear_transpose) c12c17 6
*midi_command  (gen_pianosustain) c16c64 127
*midi_command  ((!gen_pianosustain)) c16c64 0
```

```
;*********************************
;     INC_DEC
;*********************************

*increment_decrement memory gen_memoryplus gen_memoryminus 0 23 *wrap

*increment_decrement transpose gen_transposeplus gen_transposeminus -5
6 *wrap

;xxxxxxxxxxxxxxxxxxxxxxxxx
;     COMBINATION ACTION
;xxxxxxxxxxxxxxxxxxxxxxxxx

*combination_action set_button map_button 24 memory

}  ; End of computer definition


; set the mag pulse time
; where <time_ms> is the number of milliseconds that the magnets should
; be energized. The default is 100 milliseconds.

*mag_pulse_time 100

;internal couplers no tabs
*coupler  (1) solo to solo    0
*coupler  (1) grt to grt      0
*coupler  (1) grt2nd to grt2nd      0
*coupler  (1) acc to acc            0
*coupler  (1) acc2nd to acc2nd      0
*coupler  (1) ped to ped            0
*coupler  (sol_superoctave_2)  solo to solo 24
*coupler  (sol_seventeenth_135) solo to solo 28
*coupler  (grt_solo_2_grt_16) solo to grt -12
*coupler  (grt_solo_2_grt_8)  solo to grt 0
*coupler  (grt2t_solosub_to_grt2t)  solo to grt2nd 0
*coupler  (grt2t_solo_to_grt2t) solo to grt2nd -12
*coupler  (ped_solo_to_pedal) solo to ped 0
*coupler  (ped_great_to_pedal) grt to ped 0
*coupler  (ped_accomp_to_pedal) acc to ped 0
*melody_coupler (grt_solo_2_grtmelody_8) solo to grt 0

;internal
*transpose   transpose
*stop (sol_orchbell) solo orchbells '8'
*stop (grt_glockenspiel) grt orchbells    '8'
*end
```