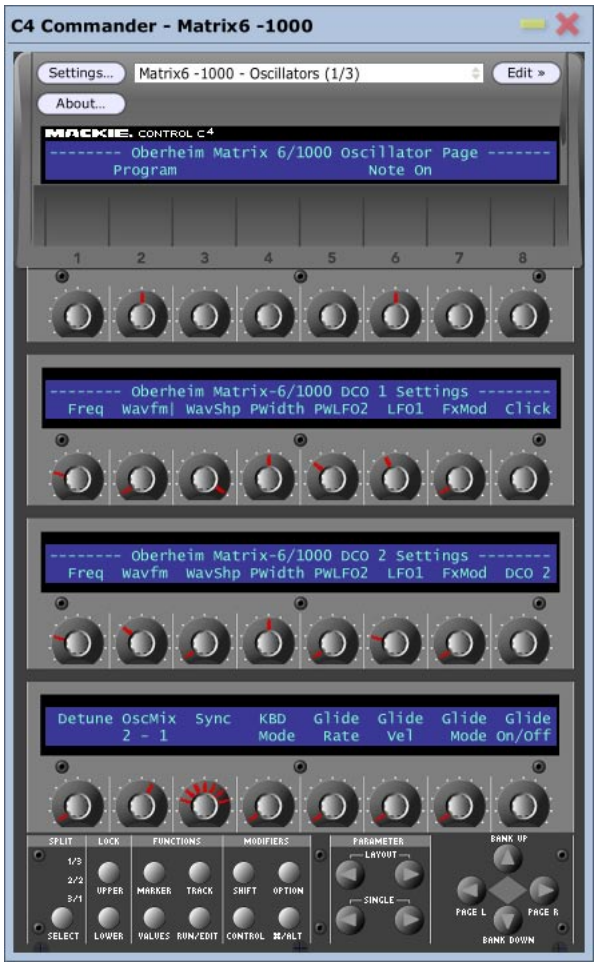
MACKIE.

PROGRAMMER'S GUIDE



C4 Commander

External MIDI Hardware Control Software

For Mackie Control C4

MACKIE.

C4 Commander Programmer's Guide

Contents

Introduction

The C4 Commander User's Guide covers the general operation of the C4 Commander software, while this Programmer's Guide provides detailed information on customizing the Instrument Definition files. You might want to customize an Instrument Definition file simply to change how a value appears in the display (ValueText), or you may need to create a new Instrument Definition file for a MIDI device that is not included in the Instruments folder provided with the software. Be sure to check for Instrument Definition file updates on our website (in the Settings window, click the Instruments tab and click on the link at the bottom of the window) to see if a new Instrument Definition file has been created for the MIDI device you have.

C4 Commander Files

C4 Commander deals with 2 types of files: Instrument Definition Files and Layouts.

Instrument Definition Files (.c4i) contain the parameters that allow C4 Commander to communicate with a particular MIDI hardware device.

Layout files (.c4l) allow the user to place controls in a specific order and to relabel control text.

The default Instrument Definition files for the C4 Commander were designed to provide the maximum amount of detail for each parameter. This detail may cause some instrument layouts to look cluttered or may cause them to be difficult to read. This can be corrected by simply using C4 Commander to move and rename parameters, hence creating an appropriate Layout file for a particular instrument. This does not affect the Instrument Definition file, but rather edits their position within the layout file. As such, it is possible to create several layouts for the same instrument, allowing maximum flexibility for a given application.

Instrument Definition Files (.c4i)

There may be instances where editing the Instruments Definition file is desirable. The main reason for this is to add ValueText messages to the definition file, which substitutes descriptive text in place of numerical values (e.g., Off/On instead of 0/1).

Another reason for editing the file is to create a new instrument file either for a MIDI device not included in the existing Instrument folder, or based on an existing definition.

Many times when creating a new file for another instrument from the same manufacturer, the manufacturer will use the same SYSEX (System Exclusive) format (what we call a “schema”) between multiple devices. In this case, creating a new instrument may be as simple as changing the Device ID and the parameter numbers within the Instrument Definition file. Creating a new Instrument Definition file from scratch, however, will require much more work.

Some Useful Terminology

SYSEX is short for System Exclusive. This is a MIDI message specific to one device. The header includes the manufacturer's ID and the device ID. A SYSEX message can be lengthy compared to normal MIDI commands (i.e., MIDI Note or Program Change messages), because it contains all the data needed to define the parameters of a device.

Masks are used to hide or ignore specific selected data in a SYSEX message.

HEX is short for hexadecimal, a base-16 number system that represents every byte as two consecutive hexadecimal digits (e.g., the binary number 0011 1111 is represented by the HEX number 3F).

V-Pot is short for Virtual Potentiometer. These are the rotary controls on the C4, which are actually digital encoders but provide the function of an analog rotary pot.

New Instrument File Creation

What you will need:

Programming experience in XML and C
Familiarity with MIDI SYSEX protocols
An appropriate text editor

Some recommended text editors:
On the PC, use Notepad or WordPad.
On the Mac, use TextEdit.

If you don't know XML, **STOP RIGHT HERE**. The assumption here is that you already have programmming experience; hence the title "C4 Commander Programmer's Guide."

Steps in creating a new Instrument file:

- Gather the instrument's SYSEX documentation (see owner's manual or contact the synthesizer manufacturer for this information).
- Define masks based on the SYSEX messages used by the instrument.
- Define the parameters to be controlled via SYSEX.
- Optional — define ValueText messages and their usage by the instrument.
- Write the instrument file in a text editor.
- Test and debug.

C4 Commander XML schema

Each instrument controlled by the C4 must have an Instrument Definition file. This is a schema we have created that is used by the C4 to hook into functions of that instrument. C4 ships with definitions of many popular instruments.

Since new instruments are being introduced all the time, there are two options for obtaining Instrument Definition files. The first is to wait for someone else to write the definition, and the second is to write it yourself. If you are impatient and wish to give Instrument Definition writing a try, the following pages will show you how.

Basics

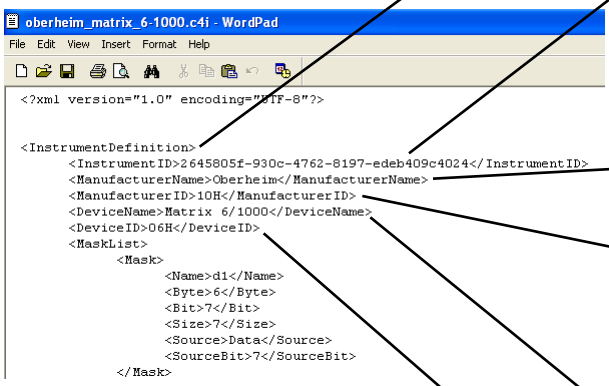
Since the format for the schema is XML, appropriate start and end tag syntax (<function> </function>) is required. Failure to do this will result in errors when the C4 Commander attempts to load your definition file.

Syntax

Please note the use of the "" marks in the Instrument Definition file's source code. These are not Microsoft Word begin/end quotation marks, as typed above. These are a text editor's straight quotation marks like this: " . If you use a word processor that substitutes open/closed (curly) quotes instead of the straight quote marks, it won't translate correctly, and the C4 Commander will return an error message.

Creating a new Instrument Definition

Every instrument Definition has a header. If you wish to create a definition for a new instrument, you will need to create the following file, or edit an existing file as a template. The header consists of the following information:



`<InstrumentDefinition>` This is the tag that defines the file as an instrument definition file and as an element contains all data necessary to communicate with and operate an instrument.

`<InstrumentID>` This is the specific instrument's MIDI ID as assigned by and registered in the MIDI Manufacturer's Association (MMA). A unique ID assigned to each instrument and stored in HEX.

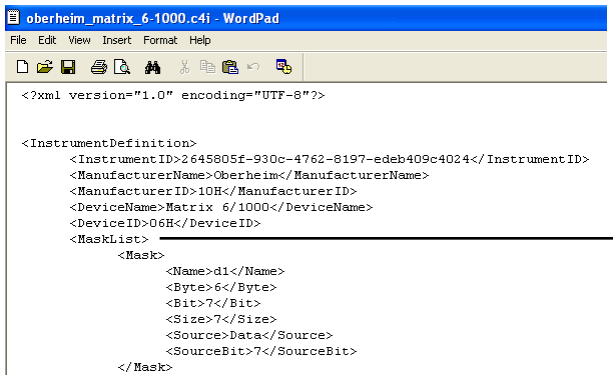
`<ManufacturerName>` Name of instrument manufacturer (e.g., Oberheim).

`<ManufacturerID>` This is the Manufacturer's MIDI ID as assigned by and registered in the MIDI Manufacturer's Association (MMA). A unique ID assigned to each manufacturer and stored in HEX.

`<DeviceName>` Name of the given instrument (e.g., Matrix 6). This is displayed in various file lists and menu trees on the C4 Commander

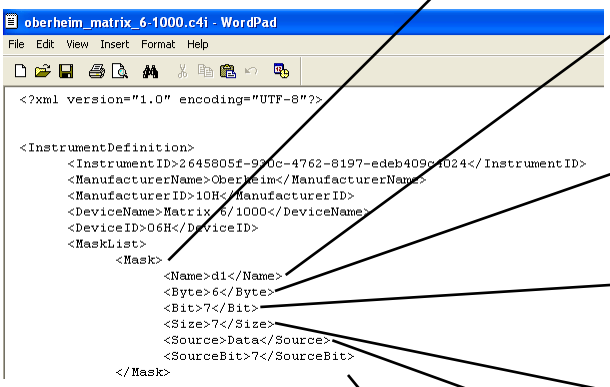
`<DeviceID>` This is the Manufacturer's Unique MIDI Device ID as assigned by and registered in the MIDI Manufacturer's Association (MMA). This is unique for each instrument model and stored in HEX.

Masks



Masks are used to enable the sending and receiving of complex MIDI messages. Since there is no "standard" for a MIDI SYSEX message, masks are the only way to ensure future compatibility with the C4 Commander and future instruments and devices.

`<MaskList>` This tag defines the beginning of the mask list section, and contains a list of all masks defined for the instrument. An instrument may have several different types of MIDI SYSEX messages, therefore multiple masks may be used. Masks are called on a per parameter basis, so an LFO message can have one mask, while a filter message may use another.



`<Mask>` This tag denotes a single mask element which specifies all data needed to apply a mask on a parameter message. This is the beginning of a single mask's definition, and each mask definition segment needs to be terminated by a corresponding </Mask>

`<Name>` The string identifier used to apply a particular mask on a parameter message (e.g., <Param mask="channel">), where Name would be "channel".

`<Byte>` An integer indicating which byte of the message (one-based, from the left) is to be masked.

`<Bit>` An integer indicating which bit of the message byte (one-based, from the right) is the left-most (starting) bit of the mask.

`<Size>` The number of bits this mask covers.

`<Source>` This value determines where the mask data is being pulled from. If a parameter ID is used, the value of that parameter will be used as the mask data. The following reserved keywords are used to indicate internal data sources:

Data — use the current V-Pot value
Channel — use the current V-Pot channel

`<SourceBit>` An integer indicating which bit of the data source (one-based, from the right) is the left-most (starting) bit of the mask data.

ValueText

Valuetext are look-up tables that can be used for parameters to display text strings instead of numerical values. An example of this might be a reverb algorithm that has values from 0-15. Instead of displaying these values, a far more intuitive approach would be to display the words “Plate,” “Arena,” “Lg. hall,” etc., on the C4’s display. Other examples of where Valuetext can be useful are:

- LFO waveforms (SINE, SAW, S/H, TRI)
- Sample waveforms (Piano, Sax, vocal, strings)
- Envelope times (displaying the number of milliseconds for compressor attack and release)
- Keyboard modes – mono, poly, omni
- Amplifier selections – British, 2x10, 1x12, Metal, Jazz
- On/Off messages for portamento, arpeggiator settings, etc.

The Valuetext parameter is a very powerful and useful tool to simplify and clarify the readouts for the C4 controls.

`<ValueTextList>` This element contains a list of all value text lists defined. After you create this header, begin typing your valuetext statements.

`<ValueText name="xxxxxxx">` This element is used to substitute text in place of a V-Pot numerical value. So instead of seeing 0 and 1 for a V-Pot value, you can have on and off.

`<Text value = "n">VALUETEXT</Text>`
A single value text substitution. The tag value (VALUETEXT) determines what will be substituted in place of the numerical parameter value. If a value="n" attribute is present, the tag value will be substituted for the parameter value n. If min="a", max="b" attributes are used, the tag value will be substituted whenever the parameter’s value lies between a and b.

Value Text Example:

```
<ValueTextList>
    <ValueText name="onoff">
        <Text value="0">Off</Text>
        <Text value="1">On</Text>
    </ValueText>
    <ValueText name="alg">
        <Text value="0">Algorithm 1</Text>
        <Text value="1">Algorithm 2</Text>
        <Text value="2">Algorithm 3</Text>
        <Text value="3">Algorithm 4</Text>
    </ValueText>
    <ValueText name="MegaOnOff">
        <Text min="0" max="63">Off</Text>
        <Text min="64" max="127">On</Text>
    </ValueText>
</ValueTextList>
```

A single valuetext can be used for multiple parameters. In the example above, the “onoff” valuetext can be referenced to reflect any parameter that has a 0-1 setting, such as “glide on/off”, “Layer on/off”, etc.

Parameters

<ParamTable> This tag defines the beginning (and end) of the parameter table. All instrument parameters must be contained withint the begininning and ending ParamTable tags. ParamTable contains a list of all parameters (functions) that the C4 can control. The order of the parameters determines the corresponding default instrument layout order, so the first parameter value is placed on the first C4 V-Pot slot when dragging te entire instrument definition file onto the C4 Commander's screen.

```
<ParamTable>
      <Param mask="d1">
            <ParamID>1</ParamID>
            <FunctionName>DCO 1 Frequency</FunctionName>
            <Message>F0 10 06 06 00 00 F7</Message>
            <ValueRangeMin>0</ValueRangeMin>
            <ValueRangeMax>63</ValueRangeMax>
      </Param>
      <Param mask="d1">
            <ParamID>2</ParamID>
            <FunctionName>DCO 1 Freq. by LFO 1</FunctionName>
            <Message>F0 10 06 06 01 00 F7</Message>
            <ValueRangeMin>0</ValueRangeMin>
            <ValueRangeMax>127</ValueRangeMax>
      </Param>
```

<Param> This tag defines a single parameter element which specifies all data needed to control that function. If a mask attribute is present, the specified mask will be applied to the parameter message before it is sent to the instrument. If a valuetext attribute is present, the appropriate substitution will be made to the parameter value displayed in the C4 commander.

<ParamMask> This calls out which mask is to be used for this parameter. It is possible (but not very efficient) to have a different parameter mask for each parameter (see <Param> above).

<ParamID> The ID associated to the given function. It can be text or numerical. Normally this should reflect the device manufacturer's sysex message parameter ID, so that if one is looking up "parameter 34 – LFO" from the manufacturer's spec, a simple search for PARAMID 34 would result in the display of this function for editing.

<FunctionName> Name of the given function, e.g., Master volume. The FunctionName is displayed as the default name in the C4 Commander screen. Since the physical C4 is limited to displaying only 7-8 characters, the C4 will automatically truncate this name to fit its display. It is usually best to keep this parameter as descriptive as possible and use the Layout rename function to create good-looking onscreen text.

<Message> The HEX message that will be sent to the C4 when a V-Pot action is triggered.

<ValueRangeMin> Minimum value for given parameter.

<ValueRangeMax> Maximum value for given parameter.

Instrument Definition Example

```
<InstrumentDefinition>
    <ManufacturerName>Yamaha</ManufactureName>
    <ManufacturerID>43H</ManufactureID>
    <DeviceName>DX100</DeviceName>
    <DeviceID>10H 12H</DeviceID>
    <MaskList>
        <Mask>
            <Name>ch</Name>
            <Byte>9</Byte>
            <Bit>7</Bit>
            <Size>4</Size>
            <Source>Channel</Source>
            <SourceBit>4</SourceBit>
        </Mask>
        .
        .
        .
    </MaskList>
    <ValueTextList>
        <ValueText name=”onoff”>
            <Text value=”0”>Off</Text>
            <Text value=”1”>On</Text>
        </ValueText>
        <ValueText name="alg">
            <Text value="0">Algorithm 1</Text>
            <Text value="1">Algorithm 2</Text>
            <Text value="2">Algorithm 3</Text>
            <Text value="3">Algorithm 4</Text>
        </ValueText>
        <ValueText name="MegaOnOff">
            <Text min="0" max="63">Off</Text>
            <Text min="64" max="127">On</Text>
        </ValueText>
        .
        .
        .
    </ValueTextList>
    <ParamTable>
        <Param mask=”ch,d1” valuetext=”alg”>
            <ParamID>17H</ParamID>
            <ParamName>Algorithm</ParamName>
            <Message>F0 22 20 33 17 00 00 F7</Message>
            <ValueRangeMin>0</ValueRangeMin>
            <ValueRangeMax>7</ValueRangeMax>
        </Param>
```

```
        <Param>
            <ParamID>39H</ParamID>
            <ParamName>Amplitude Mod Depth</ParamName>
            <Message>F0 22 20 33 39 00 00 F7</Message>
            <ValueRangeMin>0</ValueRangeMin>
            <ValueRangeMax>99</ValueRangeMax>
        </Param>

        .

        .

        .

    </ParamTable>
</InstrumentDefinition>
```

Layout files (.c4l)

A layout is defined as a mapping of all C4 virtual controls to a set of instrument parameters, i.e., a layout tells the C4 which knob controls what instrument functions. A layout may span across multiple pages. Several instruments can be mapped to from one layout and there is no restriction to duplicating the same instrument parameter on two different V-Pots. A user must create a layout before using the C4 Commander.

The <CustomLayout> element is used in two ways. First, it is used in each saved layout file for the purpose of remembering state. Second, there is a copy of the element in memory that represents the layout as the user makes changes in real time. When the layout is closed, the user is prompted to save the current state to a layout file mentioned earlier.

Layout files are created automatically when controls are positioned on a C4 Commander screen. As such, there is no need to manually edit these files. We are providing the following detail as a point of reference only, not as an invitation to modify layout files in a text editor. You could, but really, don't do it. Use the C4 commander app to generate layout files instead.

`<CustomLayout>` This element contains all data necessary for mapping instrument parameters to the C4.

`<LayoutID>` An ID used as a key to uniquely identify a particular layout. Most likely will be a generated GUI ID.

`<LayoutName>` A name specified by the user for the layout. This name appears in the top LED bar of the C4 Commander.

`<PageList>` Holds all names of the current layout's pages. The order of the child <Page> elements determines the corresponding page numbers.

`<Page>` A single page name.

`<DisplayGroups>` Holds data on all display groups.

`<DisplayGroup>` A single display group. There are two attributes associated with this element:

bars — setting to true will add horizontal dashes to the group name
pickets — setting to true will add vertical separators at the ends of the group name

`<StartControl>` The ID of the first control in the group.

`<GroupLength>` The number of V-Pots the display group spans.

`<Control>` This element represents an individual mapping from one instrument parameter to a C4 V-Pot.

`<ControlID>` The position of the V-Pot being mapped to. There are 32 V-Pots per page (if there is no split), so the 33rd V-Pot would be the upper leftmost V-Pot on page 2. If a mapping is not present for a given V-Pot, it will not be functional.

`<ManufacturerID>` Unique ID assigned to each manufacturer. Stored in HEX.

`<DeviceName>` Name of the given instrument (e.g., DX100).

`<Channel>` The MIDI channel on which the parameter message will be sent.

`<ParamID>` The parameter being mapped.

`<Display1>` A short name that will appear in the LED area above the V-Pot being mapped. The user can customize this name in the software. Default value will be the <InstrumentName> of the parameter being mapped.

`<Display2>` A short name that will appear in the LED area above the V-Pot being mapped. The user can customize this name in the software. Default value will be the <ParamName> of the parameter being mapped.

`<ValueRangeMin>` Minimum value for given parameter.

`<ValueRangeMax>` Maximum value for given parameter.

<RotaryStyle> Indicates the rotary type for the V-Pot. Default is 1.

0: Rotary action has no effect.
1: Dot — single LED light ranged from min to max.
2: Boost/Cut — multiple LEDs starting at the center top, traveling down the left or right side. Commonly used for pan effects.
3: Wrap — similar to Dot except multiple LEDs fill to the current position.
4: Spread — similar to Boost/Cut except LEDs on left and right are symmetric.

<RotaryStepSize> The amount to increment the parameter value for each step change of the rotary V-Pot. Default is 1.

<RotaryValue> The previous value of the parameter the V-Pot had when the layout was last used. Default will be half way between the min and max.

<ToggleResetValue> The value to set the param to upon "quick" toggle action of the V-Pot. Default will be half way between the min and max.

Layout File Format

```
<CustomLayout>
    <LayoutID>75a1e4445dcd311d</LayoutID>
    <LayoutName>line6 podtr</LayoutName>
    <Control>
        <ControlID>0<ControlID>
        <InstrumentID>7e5b1549-c3c6-4f59-a445-23bca5f06b85</InstrumentID>
        <ParamID>0</ParamID>
        <Channel>1</Channel>
        <Display1>Controllers</Display1>
        <Display2>Bank Select</Display2>
        <ValueRangeMin>0</ValueRangeMin>
        <ValueRangeMax>127</ValueRangeMax>
        <Value>0</Value>
        <ToggleResetValue>64</ToggleResetValue>
        <RotaryStyle>1</RotaryStyle>
        <ToggleStyle>0</ParamID>
        <RotaryStepSize>1</RotaryStepSize>
        <RotaryMegaStepSize>0</RotaryMegaStepSize>
        <RotaryMicroStepSize>0</RotaryMicroStepSize>
    </Control>
    <DisplayGroups>
        <DisplayGroup bars="true" pickets="true">
            <GroupName>Effects</GroupName>
            <StartControl>8</StartControl>
            <GroupLength>8</GroupLength>
        </DisplayGroup>
        .
        .
    </DisplayGroups>
    <PageList>
        <Page>Line 6 POD Amp</Page>
        <Page>FX Deep Editing</Page>
        <Page>Page 3</Page>
        .
        .
    </PageList>
</CustomLayout>
```

Console File (.c4s)

The Console file contains the default system parameters of the C4 Commander. This file is automatically generated by the C4 Commander and usually requires no editing or enhancement. It is mentioned here for reference only, and not as an enticement to modify its contents.

MACKIE.

16220 Wood-Red Road NE • Woodinville, WA 98072 • USA
United States and Canada: 800.898.3211
Europe, Asia, Central and South America: 425.487.4333
Middle East and Africa: 31.20.654.4000
Fax: 425.487.4337 • www.mackie.com
E-mail: sales@mackie.com

Part No. SW0371 Rev. A 06/06