# TeensyExpression2.0

## Specs

Buttons

LEDs

Display

Pedals

Rotary Controls

USB Midi Port

USB Serial Text

Serial Text

Serial Midi Port

## Overview

The device can send and receive Midi Sysex messages to send or receive the Configuration and Current Rig.

The Pedals and Rotaries are programmed via the Configuration.

The Configuration is loaded and saved to EEPROM.

Pedals can be named and send Midi CC Messages.

Rotaries can send Midi CC Messages.

The Buttons, LEDs, and Display, and everything else, are programmed via Rigs.

One Button is Special. A long press of THE\_SYSTEM\_BUTTON is reserved for going into Configuration Mode.

## License

## Credits

## Also See

# Electronics

# Build

# UI

## Screen Layout

## The Default LooperRig

## FTP Tuner

## FTP Sensitivity

# Firmware

## Installation

## Build and Install

## SD Card

# Rigs

## Rig Basics

### Rig Names

Rigs have names of up to 16 characters.

Rig Names can consist of A-Z, a-z, 0-9, and underscore.

There is a Default Rig name that is reserved for use by the system.

There will be a UI allowing the user to edit the Rig Name.

Editing a Rig Name will mark the Rig as Dirty.

### Rig Filenames

Rig filenames are the Rig Name an extension of .rig

Illegal Rig filenames (including the Default Rig Name) will be ignored by the Load UI

### Rig Files

Rig Files are regular text files.

Comments start with the pound # sign and go to the end of the line.

Rig files are parsed after all comments and white space are removed.

The language is case insensitive. Quoted Strings are case sensitive.

#### Initialization Section

Rig Files consist of an Initialization Section followed by LED and Button sections.

The Initialization Section consists of Listen definitions, String definitions, and Statements.

Listen definitions, String definitions, and Statements can exist in a Rig File in any order within the Initialization Section, but they will be written out in the Listen, String, Statements order, preserving the order of Statements.

#### LED Sections

LED Sections specify the LED to which they apply

LED Sections can have COLOR and BLINK subsections.

##### COLOR Subsection

The color subsection specifies a LedColorExpression that evaluates to one of the LED\_COLORS .

##### BLINK subsection

The blink subsection specifies a NumericExpression that causes the LED to blink if it evaluates to a non-zero value.

#### BUTTON Sections

Button Sections specify the Button to which they apply

Button Sections can have PRESS, CLICK, LONG, and RELEASE subsections.

Button Sections have a StatementList associated with them containing Statements that will be executed in order.

##### PRESS Subsection

Executed as soon as the button is Pressed

##### CLICK Subsection

Executed when the button Released if it is done before a certain time or there is no LONG section associated with the buton.

##### LONG Subsection

Executed when the button is held down for a certain amount of time.

##### RELEASE Subsection

Executed when the button is Released.

a Syntax COLOR COLON ColorExpression SEMI\_COLON

Buttons and the Initialization Section within Rigs can send CC, Program Change, Note On, and Note Off messages.

Rigs can receive CC messages to change Values.

#### Order of LED and Button Sections

LED and Button Sections can occur in any order in Rig Text files, but they will be written out in numeric order with LED(0) followed by BUTTON(0), the LED(1) followed by BUTTON(1), and so on.

### Loading and Saving

There is a Default Rig stored as constants in the program that has the Default Rig Name

There can be Rig files on the SD card.

The Default Rig, or one of the Rigs Files on the SD card, is Loaded when the device boots.

The loaded rig is called the Current Rig.

The Current Rig can be loaded from the Default, or from the SD card via the Load UI.

A Rig loaded from the Default or SD card starts out as Clean.

The Load UI presents the Default Rig Name followed by any legal Rig Names from the SD card sorted in alphabetical order.

The Load UI can be cancelled or accepted.

If accepted, and the Current Rig is Dirty, the user will be given a chance to Save the Current Rig before proceeding.

A Rig loaded via Sysex starts off as Dirty.

The Save UI will be disabled if the Rig is not dirty.

A dirty Rig will be shown in the title bar with an Asterisk.

The Save UI will prompt for a Verify or Cancel if the existing filename already exists on the SD Card.

When the Current Rig is saved to the SD card, it is marked as Clean in memory.

There will be a New Rig UI that will prompt for a Save if the current Rig is dirty and will create a dirty empty Rig, with the name Untitled.

Loading a Rig sets all Values to Zero and Clears all Strings before it is executed.

If there are any problems parsing a Rig File or Sysex message, the user will be notified via an Error Dialog and the system will otherwise be unchanged.

## Rig Editing

There will be a selection in the Configuration System called “Rig”.

Changes to the Rig are separate from changes to the configuration.

Under the Rig will options for the following:

- Initialization Statements

- Listens

- Strings

- LEDs

- Buttons

Each of these options will, in turn, present a UI that allows you to Edit the:

- The initialization - the 128 possible Listen slots

- the 128 possible String slots

- the 25 possible LED slots

- the 25 possible Button slots

In general, these UIs will include a OK and Abort button to allow you to Revert or Accept changes you make

The details of how the UI will look and behave in order to allow the user to edit Strings, including the Rig Name, Expressions, and Statement Lists have not yet been finalized.

The functionality will initially be implemented to Load a Rig File, then figure out how to make that the default, and then finally, design the User Interfaces.

## Rig File Syntax

The Initialization Section (anything before any LED or BUTTON sections) can contain LISTEN and STRING definitions and Statements in any order.

### SEMI\_COLON

Listen and String Definitions, Color and Blink Expressions, and Statements are terminated with SEMI\_COLONS

### midi\_port

0 (zero) specifies the regular Midi Input or Output port

1 (one) specifies the Serial Input or Output Port.

### midi\_channel

The midi\_channel is a number from 1 to 15

### cc\_number

A cc\_number is from 0 to 127

### LISTEN(midi\_port, midi\_channel, in\_cc, out\_value\_index)

Listens on the given midi\_port for the given in\_cc control messages, and places their values into the Value memory location given by out\_value\_index.

LISTEN definitions can occur anywhere in the Initialization Section of a Rig.

LISTENs will be written out as the first thing in the Text File, in the order of their OUT\_VALUEINDEX.

### STRING(index, quoted string)

Defines a STRING that can be referenced by index.

Used only in DISPLAY statements.

In the definition the index must be a number from 0 to 127.

STRING definitions can occur anywhere in the Initialization Section of a Rig.

STRINGs may not contain double quotes.

STRINGs are limited to 32 characters and can contain Ascii characters from space (32) to lowercase z (122), with the exception of the double quote (Ascii character 34)

In references the index may be a NumericExpression.

STRINGS will be written out after the LISTENs in the order of their indexes.

### LED(led\_number) : [ Color Subsection ] [ Blink Subsection ]

LED\_NUMBER must be from 0 to 24.

#### COLOR : LedColorExpresion

LedColorExpression evaluates to one of the LED\_COLOR constants.

#### BLINK : NumericExpression

After the expression is evaluated, the LED will blink if the result is non-zero.

### BUTTON(button\_number) : Button Subsections

button\_number must be from 0 to 24.

The BUTTON subsections include the following:

- PRESS : StatementList

- CLICK : StatementList

- LONG : StatementList

- RELEASE : StatementList

### DISPLAY\_COLOR

BLACK

BLUE

RED

GREEN

CYAN

MAGENTA

YELLOW

WHITE

NAVY

DARKGREEN

DARKCYAN

MAROON

PURPLE

OLIVE

LIGHTGREY

DARKGREY

ORANGE

GREENYELLOW

PINK

### DisplayColorExpression

DISPLAY\_COLOR

NumericExpression ? DisplayColorExpression : DisplayColorExpression

### DISPLAY\_AREA

There are certain accessible areas of the screen.

The DISPLAY\_AREAs define the Justification and Font Size

#### AREA\_UPPER\_LEFT

Currently the short Patch Name area

1/2 screen width

Big Font

Left Justified

#### AREA\_UPPER\_RIGHT

Currently unused

1/2 screen width

Big Font

Right Justified

#### AREA\_SECOND\_LINE

Currently the long Patch Name

Full screen width

Small Font

Left Justified

#### AREA\_THIRD\_LEFT

Currently the Song Name

1/2 screen width

Small Font

Left Justified

#### AREA\_THIRD\_RIGHT

Currently the songMachine State

1/2 screen width

Small Font

Right Justified

#### AREA\_BOTTOM\_LEFT

Currently user programmable songMachine Area (0)

1/4 screen width

Big Font

Left Justified

#### AREA\_BOTTOM\_RIGHT

Currently user programmable songMachine Area (1)

1/4 screen width

Big Font

Center Justified

### LED\_COLOR

LED\_BLACK

LED\_RED

LED\_GREEN

LED\_BLUE

LED\_YELLOW

LED\_PURPLE

LED\_ORANGE

LED\_WHITE

LED\_CYAN

### LedColorExpression

LED\_COLOR

NumericExpression ? LedColorExpression : LedColorExpression

### NumericExpression

Number

VALUE[NumericExpression]

(NumericExpression)

NOT NumericExpression

NumericExpression + NumericExpression

NumericExpression \* NumericExpression

NumericExpression ? NumericExpression : NumericExpression

### StringExpression

#### STRING[ NumericExpression ]

### StatementList

Statement;

Statement; StatementList

### Statements

#### VALUE[NumericExpression] = NumericExpression

Sets the given value to the result of the NumericExpression.

#### sendCC( midi\_port, midi\_channel, cc\_number, NumericExpression)

Sends a Midi Control message to the given midi\_port, midi\_channel, and cc\_number with the value given by the NumericExpression.

#### sendPgmChange( midi\_port, midi\_channel, NumericExpression )

Sends a Midi Program Change to the given midi\_port and midi\_channel with the patch number given by the NumericExpression.

#### sendNoteOn( midi\_port, midi\_channel, NumericExpression, NumericExpression)

Sends a Midi Note On message to the given midi\_port and midi\_channel with a NumericExpression for the Note Number and another one for the Velocity.

#### sendNoteOff( midi\_port, midi\_channel, NumericExpression)

Sends a Midi Note Off message to the given midi\_port and midi\_channel with a NumericExpression for the Note Number.

#### sendAllNotesOff( midi\_port, midi\_channel ).

Sends a Midi All Notes message to the given midi\_port and midi\_channel.

#### DISPLAY(DISPLAY\_AREA, StringExpression)

Displays the given STRING in the given DISPLAY\_AREA

#### ftpTuner()

Pops up the ftpTuner dialog.

#### ftpSensitivity()

Pops up the ftpSensitivity dialog.

#### songMachine()

TBD

# Configuration

## Pedal Configuration Dialog

# Advanced Topics

## Serial Output

## Midi Monitoring

## Serial Midi Protocol

## Song Manager

## Serial SD File Management