MacroPot Programming Guide

The most recent version of this document is located here:
https://docs.google.com/document/d/1nPlYxyJ24XFE5UrgwzTidY-a2B3J3utPwMfMKG4Hb1w/edit?usp=sharing

Please, note basic firmware update steps as well as default macro programming are covered in the Assembly Guide: https://docs.google.com/document/d/1nHQZJlmbKfMoiNE-2jQ2WAEOWn6yFhDtXKQVzbqB8Ts/edit?usp=sharing

Deftaudio MacroPot controller is multifunctional programmable MIDI controller based on the Teensy microcontroller. MacroPot firmware is open and available for your own modifications. This allows you to integrate it with Arduino environment and leverage community to expand and develop additional functionality.
All configuration of MacroPot parameters are done by using SysEx commands. There is no built in setup menu inside MacroPot, it's designed to be simple to operate by switching pre-defined scenes or in this terms - Macros.

Hardware features:

- Rotary dual color (with brightness control) encoder with a push button
- OLED display for a visual feedback and preset information
- Up/Down buttons for a preset management
- 1 x MIDI Input (3.5mm TRS MIDI A or B)
- 3 x MIDI Output (3.5mm TRS MIDI A or B)
- USB MIDI Class compliant with 4 x MIDI devices under OS (1 physical port)
- 1 x Audio/CV/trigger/sync output (via PWM or DAC)
- Programmable activity LED
- Standalone operation from a power bank or power supply.

Software features:

MacroPot is based around the idea of controlling multiple parameters at a time and quick switching between presets (macro). Within a current macro it's possible to assign up to 5 simultaneous active controllers for transmission as well as a push button behavior. This allows to create layers or splits in values, assign controllers to different events, messages and MIDI ports, implement curves. MacroPot is also a 1x3 MIDI interface device with merging capability, which enables it at any place in the setup — as a USB controller for any class compliant device or pass through HW MIDI input for your gear.

- **Number of Macros:** 10
- **Number of controllers per Macro:** 5
- **Controller parameters:**
    - USB output port number 1-4 to send a data out
    - MIDI port number 1-3 to send a data out

    - Type of the transmitted data: CC, Note, Program Change, Pitch Band (Synclavier mode), NRPN(future FW releases)
    - MIDI Channel 1-16
    - Programmable CC number, MIDI Note number or Program Change number
    - Controller range: Low value, High value
    - Controller mapping onto encoder position: Map start, Map end
    - Controller Curve for creating curves for CC messages
    - Programmable MIDI Note number for Note ON messages
- **Push Button parameters:**
    - USB output port number 1-4 to send a data out
    - MIDI port number 1-3 to send a data out
    - Type of the transmitted data: CC or Note
    - MIDI Channel 1-16
    - Programmable CC number or MIDI Note number
    - Programmable CC value or MIDI Note velocity
    - Different action behavior supported: Push button or Toggle(for Note)
- MIDI Merge capability: configurable per Macro HW MIDI input can pass through to any HW of 1-3 outputs
- Assignable Macro name: up to 7 characters on the display
- Programming: SysEx over USB

SysEx programming:

MacroPot programming is done under SysEx mode, which is a state where it can receive multiple SysEx configuration messages at a time, validate them and save into the non-volatile memory. To enter this mode UP and DOWN buttons have to be pressed simultaneously.
There are multiple utilities that can generate and send SysEx commands, all examples below are using a cross-platform command line utility by Geert Bevin.
https://github.com/gbevin/SendMIDI
This provides a readable output, easy to integrate into scripts. You should consider defaut scripts as the first start: https://github.com/Deftaudio/MacroPot/SysEx%20scripts/
In the meantime GUI editor development is ongoing and in future it will be available in addition to SendMIDI for programming MacroPot without dealing directly with HEX parameters.
As utility is cross-platform (Windows, OS X, Linux) all examples below will work without any change. Data is expected to be transmitted over USB MIDI port. Any of four ports can be used for that, but typically first port is easier for naming. Use "*sendmidi list*" to identify MacroPot first.

Default syntax for SendMIDI is following:

sendmidi dev "MacroPot Port 1" hex syx [sysex data here]

Note: You don't need to specify F0/F7 start and closing bytes with SendMIDI utility. It always adds this on it's own. So, only actual data starting from Dev ID (7D) is specified.

SysEx structure.

SysEx command structure replicates the software feature hierarchy. Programming is done in small chunks, where you select a feature to program (example Macro #1, Controller #1) and then specify its configuration settings. All parameters are translated in Hex, and so decimal values have to be converted first.

All SysEx packages have a common structure:

Header	Dev ID	Macro number	Configuration subject	*Configuration parameters*	Footer
F0 - common for all Sysex Data	**7D** - unique device ID, MacroPot uses 7D	**[0-9]** - a macro number for programming, ten in total, start with zero.	**0** - Push Button configuration **[1-5]** - for Controllers 1 to 5 **64** - Global parameters per Macro	*Specific to the configuration subject, no determined size.*	**F7** - common for all Sysex Data

All incoming SysEx data is verified for the data integrity by checking a closing byte F7 and matching the length of the SysEx package.

Note: You don't need to do anything to save data into non-volatile memory, however it only saves after you exit SysEx mode (UP button) and switch a macro to any other. This provides an option to do multiple changes, validate parameters and only after that write it to a memory.

Button Configuration:

F0 7D [macro#] 00	[USBport#]	[MIDIport#]	[type]	[channel]	[number]	[value]	[push]
A header and macro # 0 to 9	0, 1 to 4	0, 1 to 3	0,1,2	1 to 0x10	0 to 0x7F	0 to 0x7F	0, 1

Description:

[USBport#]	USB 1-4 port # to send a data out, 0 means don't send to any of USB ports
[MIDIport#]	HW MIDI 1-3 port # to send a data out, 0 means don't send to any of MIDI ports
[type]	0=Disabled, 1=CC, 2=Note
[channel]	MIDI Channel 1-16, hex 0x0 - 0x10
[number]	CC controller # or Note # depends on the [type] mode, see note to values chart

[value]	Velocity for Note ON messages or CC value
[push]	1=push mode, 0=toggle for NOTE ON/OFF message

Note to values chart:

Octave	Notes											
	C	**C#**	**D**	**D#**	**E**	**F**	**F#**	**G**	**G#**	**A**	**A#**	**B**
0	0	1	2	3	4	5	6	7	8	9	A	B
1	C	D	E	F	10	11	12	13	14	15	16	17
2	18	19	1A	1B	1C	1D	1E	1F	20	21	22	23
3	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F
4	30	31	32	33	34	35	36	37	38	39	3A	3B
5	3C	3D	3E	3F	40	41	42	43	44	45	46	47
6	48	49	4A	4B	4C	4D	4E	4F	50	51	52	53
7	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F
8	60	61	62	63	64	65	66	67	68	69	6A	6B
9	6C	6D	6E	6F	70	71	72	73	74	75	76	77
10	78	79	7A	7B	7C	7D	7E	7F				

Examples:

sendmidi dev "MacroPot Port 1" hex syx 7d [macro#] 00 [USBport#] [MIDIport#] [type] [channel] [number] [value] [push]

Macro 0 button programming, output to USB#1, HW MIDI #1, note C-4 push mode at ch1

sendmidi dev "MacroPot Port 1" hex syx 7d 00 00 01 01 02 01 30 7f 01

Macro 1 button programming, output to USB#1, HW MIDI #1, note C-4 toggle mode at ch1

sendmidi dev "MacroPot Port 1" hex syx 7d 01 00 01 01 02 01 30 7f 00

Macro 2 button programming, USB#1, HW MIDI #2, CC#48 (hex 30) push mode at ch1

sendmidi dev "MacroPot Port 1" hex syx 7d 02 00 01 02 01 01 30 7f 01

Macro 3 button programming, USB#2, NO HW MIDI, CC#48 toggle mode at ch1

sendmidi dev "MacroPot Port 1" hex syx 7d 03 00 02 00 01 01 30 7f 00

Controller Configuration:

F0 7D [macro#]	*[controller#]*	[USBport#]	[MIDIport#]	[type]	[channel]	[number]	
A header and macro # 0 to 9	1 to 5	0, 1 to 4	0, 1 to 3	0,1,2,3, 5	1 to 16	0 to 0x7F	

	[lowvalue]	*[highvalue]*	*[mapstart]*	*[mapend]*	*[curve]*	*[velocity]*
	0 to 5	0, 1 to 4	0, 1 to 3	0,1,2,3, 5	1 to 0x10	0 to 0x7F

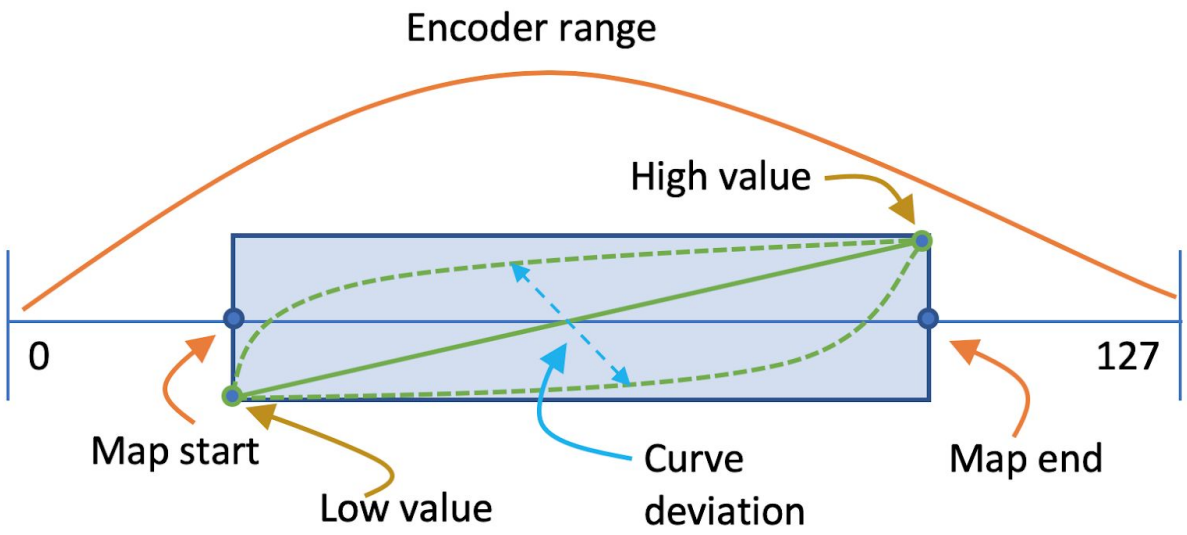
Description:

[controller#]	A controller for editing, one at a time
[USBport#]	USB 1-4 port # to send a data out, 0 means don't send to any of USB ports
[MIDIport#]	HW MIDI 1-3 port # to send a data out, 0 means don't send to any of MIDI ports
[type]	0=Disabled, 1=CC, 2=Note, 3=Program Change, 4=NRPN(to be implemented), 5=SynclavierMODE(can be the only in a macro, see a section)
[channel]	MIDI Channel 1-16, hex 0x0 - 0x10
[number]	CC controller #, Note # or Program Change # depends on the [type] mode
[lowvalue]	Low Value for CC range or main value for Note messages
[highvalue]	High Value for CC range, ignored if the type is Note
[mapstart]	Start position for mapped range in CC mode, main value for Note mode
[mapend]	End position for mapped range in CC mode, ignored if the type is Note
[curve]	For creating curves for CC messages deviation middle position is 0x40 have to be set for default
[velocity]	Velocity for Note ON messages

Note: NRPN mode to be implemented in future releases. Synclavier mode is covered in another section.

Understanding of CC ranges and mapping.

MacroPot is capable to produce multiple MIDI messages at the same type for a defined encoder move. That's called a controller. There are up to 5 controllers per macro that can send messages such as Control Changes, Notes, or Program Changes. They are very flexible but some configuration parameters may look confusing as they are unique to MacroPot. The illustration below is made to simplify this:



Consider whole encoder range from 0 to 127 (or 0x7F in hex). Within this range you can map a controller with Map Start, Map End, Low Value and High Value parameters. Map Start determines where controller is triggered with Low Value and it goes by some curve until it reaches Map End and it's where it has the maximum defined value - High Value. Curve Deviation (very similar to Velocity Curves in MIDI keyboards) is how linear its function. In the middle parameter (0x40) it's exactly linear, while going to upper or lower create a response shown on a chart.

As Map Start/End and Low/High Value parameters are per Controller, that allows to create overlapping controllers with different values and distributions weight. You also may decide to split the whole range between controllers or create fade effect between them by shaping curves and other parameters. Controller values can also be decrementing, so it starts high to goes down. This can be useful for some parameters such as envelope attack, where for instance you want to open a filter for staccato sounds.

Your imagination is the only limit.

Examples:

sendmidi dev "MacroPot Port 1" hex syx 7d [macro#] [controller# 1-5] [USBport#] [MIDIport#] [type] [channel] [number] [lowvalue] [highvalue] [mapstart] [mapend] [curve] [velocity]

Macro 0 controller 1 programming, output to USB#1, HW MIDI #1, CC #48 (0x30) starts from 0 to 127 (0x7F) with a range from 0 to 127 (0x7F). *(Basically maps an encoder 1:1 on a complete range.)*

sendmidi dev "MacroPot Port 1" hex syx 7d 00 01 01 01 01 01 30 00 7f 00 7f 40 7f

Macro 1 program controllers 1 to 4 from 0 to 127 (0x7F) all range to ALL USB and MIDI ports

sendmidi dev "MacroPot Port 1" hex syx 7d 01 01 01 01 01 01 30 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 01 02 02 03 01 01 31 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 01 03 03 03 01 01 32 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 01 04 04 04 01 01 33 00 7f 00 7f 40 7f

Note: Updating one controller doesn't erase settings of another controller. There are some conditions where you want simply disable a controller, such as in the example below for controller #5:

sendmidi dev "MacroPot Port 1" hex syx 7d 01 05 01 01 00 01 00 00 7f 00 7f 40 7f

Macro 1 program split controllers 1 and 2, where controller 1 goes from 0 to 127(0x7F) on the range 0 to 64(0x40) and controller 2 goes from 0 to 127(0x7F) on the range 65(0x41) to 127 (0x7F).

sendmidi dev "MacroPot Port 1" hex syx 7d 02 01 01 01 01 01 30 00 7f 00 40 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 02 01 01 03 01 01 31 00 7f 41 7f 40 7f

Macro 3 program controllers 1 and 2 from 0 to 127(0x7F) on 1 and 2 USB and MIDI ports, while Note message is sent to port 3 and the end of the range

sendmidi dev "MacroPot Port 1" hex syx 7d 03 01 01 01 01 01 30 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 03 02 02 02 01 01 31 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 03 03 03 03 02 01 00 00 7f 7e 7f 40 7f

Macro 4 program controller 1 and 2 from 0 to 127(0x7F) for CC#48 (0x30) and CC#49 (0x31) on 1 and 2 USB and MIDI ports, adds a Curve on controller 2, while Program Change #64 message is sent to port 3 at the end of the range

sendmidi dev "MacroPot Port 1" hex syx 7d 04 01 01 01 01 01 30 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 04 02 02 02 01 01 31 00 7f 00 7f 7f 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 04 03 03 03 03 01 40 00 7f 70 7f 40 7f

Synclavier Mode.

It's a special mode designed to integrate MacroPot into Synclavier Pocket or GO iOS applications. The app expects to receive a pitchband information and change a parameter value to applied move of the pitchband wheel. MacroPot doesn't have a spring that returns an encoder to the zero position such as an original ORK or VPK keyboard or newer Synclavier controller. Instead of that it translates the speed of the encoder movement to the pitch band value. So while user rotates the encoder it changes a value at the rate of the rotation speed and direction.
As Synclavier mode is such a special, there is no start or end of the encoder values, it's only possible to run it alone within a macro and can't be combined with other controllers.

Example:
sendmidi dev "MacroPot Port 1" hex syx 7d [macro#] [controller# 1-5] [USBport#] [MIDIport#] [type] [channel] [number] [lowvalue] [highvalue] [mapstart] [mapend] [curve] [velocity]

Macro 9 Synclavier mode enabled on USB port 2. Other controllers to be erased.
sendmidi dev "MacroPot Port 1" hex syx 7d 09 01 02 00 05 01 00 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 09 02 01 01 00 01 00 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 09 03 01 01 00 01 00 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 09 04 01 01 00 01 00 00 7f 00 7f 40 7f
sendmidi dev "MacroPot Port 1" hex syx 7d 09 05 01 01 00 01 00 00 7f 00 7f 40 7f

Note. *Non related parameters such as Map start/end, Low/High value are ignored but they have to be specified for the consistency of the SysEx message size.*

Macro Configuration:

F0 7D [macro#]	MacroName [7]	MidiThru
A header and macro # 0 to 9	7 x ASCII characters	0 - 7

Description:

MacroName [7]	is a 7 byte sequence of the macro name which represents seven ASCII characters. It's convenient to use online tools to translate Text to Hex: https://online-toolz.com/tools/text-hex-convertor.php You'll need to separate the output by each ASCII code.
MidiThru	A binary code that determines if HW MIDI input is forwarded to one of the MIDI outputs. It's designed to work as per table below.

By default all the data that comes to MIDI Input is forwarded to USB Port 1 as well as all data from USB ports 1-3 is forwarded to MIDI Out 1-3. This allows using MacroPot as a MIDI interface. However there are situations, where MacroPot preferably to stay in the middle of the MIDI chain, such as working in standalone mode in front of the MIDI gear or synthesizer. For that you may decide to forward all incoming MIDI data from the input to one or many output ports:

MidiMerge value	Binary code	Description
0	000	No MIDI thru
1	001	MIDI In forwarded to Port 1
2	010	MIDI In forwarded to Port 2
3	011	MIDI In forwarded to Port 1 and 2
4	100	MIDI In forwarded to Port 3
5	101	MIDI In forwarded to Port 1 and 3
6	110	MIDI In forwarded to Port 2 and 3
7	111	MIDI In forwarded to all three ports

Examples:

Macro 5 forward MIDI Input to MIDI Output port 1 and change the name to "Merge" *(6d 65 72 67 65 20 20 in hex**)***
sendmidi dev "MacroPot Port 1" hex syx 7d 05 64 6d 65 72 67 65 20 20 01

Macro 6 orward MIDI Input to all MIDI Output ports and change the name to "MergALL" *(4d 65 72 67 41 6c 6c)*
sendmidi dev "MacroPot Port 1" hex syx 7d 06 64 4d 65 72 67 41 6c 6c 07