

CTR2-MIDI

Operation Manual

v1.01.00c



Last Revision: **May 12, 2024**

Copyright 2024 – Lynovation.com

All rights reserved

Updated to firmware **v1.01.00c**

Revised sections for this version are highlighted in yellow

Contents

Introduction	3
Legal Notice.....	4
How to use this manual	4
Change Log	5
What is CTR2-MIDI?	6
Background	6
A New Product is Born	6
Hardware	7
Buzzer Volume Control Option	7
Configuring the MIDI Firmware for First Use	8
Terminal Display.....	8
Device Names	8
Paddle Mode.....	9
Configuring Encoder Modes	9
Calibrating MFB ADC Counts.....	10
Connecting CTR2-MIDI to the App	11
SmartSDR and SDR-Control for iOS or MacOS	11
USB MIDI	11
Bluetooth-LE MIDI	12
Thetis on Windows.....	13
CTR2-MIDI Functions	14
Multi-Function Buttons (MFBs)	14
Encoder Mode Indications	15
Paddle Mode.....	15
Mapping in SmartSDR and SDR-Control for iOS and MacOS	16
Importing the CTR2-MIDI_XXXX.map file	16
Editing the MIDI Controller Map	16
Registering Controls.....	16
Mapping Functions to Controls	17
Using Paddles with the App's Keyer.....	18
Using a Straight Key or PTT Switch.....	18
Mapping Functions in Thetis	19
Advanced Options	21
Operation with N1MM Logger+	21
Troubleshooting	22
Appendix A: Using CTR2-MIDI Firmware on CTR2-Micro	24
Flashing the Firmware.....	24
Operating the <i>Micro</i> with <i>MIDI</i> Firmware	25
Appendix B: Installing or Updating CTR2-MIDI Firmware.....	26
Installing using Linux or Mac.....	28
Appendix C: Configuring Tera Term	29
Appendix D: Configuring Putty	30
Serial Session.....	30
Appendix E: Change Log.....	32

Introduction

CTR2-MIDI (called the *MIDI* in this document) is a custom MIDI controller designed for ham operators. It supports both USB and Bluetooth-LE MIDI connections. It is an ideal companion to Marcus Roskosch's (DL8MRE) [SmartSDR](#) and [SDR-Control](#) apps for iOS and MacOS and is also at home working with [Thetis](#) or [SDR-Console](#) to control your Apache Labs ANAN or Hermes Lite 2 (among others) on your Windows computer.



MIDI commands can be used to change frequency, mode, volume, CW speed, control the keyer and PTT, and many other functions in the app.

MIDI commands are sent directly to the radio control app. The app manages the user interface and the remote connection to the radio. The *MIDI* can be powered from the computer's USB port, and iOS device's accessory jack using a OTG adapter, a 5 volt cell phone charger, or a USB battery pack making the *MIDI* an idea companion for portable operation.



Use USB MIDI when the *MIDI* is powered by the device running the app. If the *MIDI* is powered separately use Bluetooth-LE MIDI to connect it to the app.

This photo shows the *MIDI* being powered by an iPhone though a Lightning OTG adapter. A charger or USB battery can be connected to the OTG adapter's Lightning jack to power both devices.

The *MIDI* includes the following features:

1. USB MIDI and Bluetooth-LE MIDI connections are supported. Works with Windows (USB only), Mac, Linux, iOS, and Android (untested but should work) devices (USB or Bluetooth-LE).
2. Four encoder modes control a total of 8 "Wheel" or "Slider" parameters like tuning, volume control, etc. Each encoder mode has a primary and a secondary function. Primary functions are executed when you turn the encoder. Secondary functions are executed when you press and turn the encoder. Press and release the encoder for less than 1 second to change the encoder mode. Do not turn the encoder when pressing it as this sends secondary control functions. The current mode is indicated by two yellow LEDs in the upper left corner of the unit. The type of each encoder function is set in the *MIDI* using a terminal program like Putty or Tera Term.
3. Six dual-mode pushbuttons. One MIDI **Button** control is assigned to the short-press and another one to the long-press function.

4. Connect your paddles to the 3.5mm (1/8") stereo jack on the *MIDI* and use your paddles to control the keyer in the app (not all apps have keyers). The paddle input jack can also be used for straight key input and PTT control when in voice modes. Long-press the encoder (> 1 second) in the encoder's Home mode to toggle the paddle input function. You can use the straight key option to connect N1MM Logger+ to your app. See [Operation with N1MM Logger+](#). If your app doesn't have a built-in keyer you can use the paddle inputs to control other app functions.
5. Power the *MIDI* from any USB 5 volt power source; computer USB port, cell phone charger, USB battery, or through an OTG adapter on your mobile device.
6. Firmware updating is done through the USB-C port.

Configuring the *MIDI* is a two step process:

1. Use a terminal program such as Putty or Tera Term to set the [type of MIDI control](#) associated with each encoder function. The encoder supports eight different MIDI Wheel/Slider controls depending on the selected encoder mode and whether you just turn or press & turn the encoder.
2. In the app, assign the MIDI controls received from the *MIDI* to functions you want to control. This is called [mapping the controller](#) and *must be done by the user on their app*. I supply basic maps for SmartSDR, SDR-Control, Thetis, and SDR-Console in the firmware zip file available [here](#). These maps can be [imported](#) into your app.

IMPORTANT NOTICE: *CTR2-MIDI is not Marcus's product and he offers no support for it. He was kind enough to include MIDI control in his programs. Please do not waste his valuable time with troubleshooting questions or asking for additional features related to CTR2-MIDI. If you have questions, contact me. My email is good on QRZ.com.*

Legal Notice

What would a manual be without a legal notice? Here goes...

- This is a hobby endeavor. Nothing is guaranteed! Use this device at your own risk!
- I will do my best to make sure you receive functioning hardware if you buy the assembled unit and will work with you if there is a problem with your unit on arrival.
- I cannot guarantee or warranty the hardware supplied in the kit.
- I make no warranty that the firmware provided for CTR2-MIDI will perform up to your expectations or be suitable for your application. Software bugs are a fact of life and I try to find and correct all bug reports to the best of my ability ASAP.

How to use this manual

This manual should be used as a reference manual. An expanded Help system if you will. Items in the Table of Contents link to their write up in the manual. The main categories have short write ups describing the functions available in that section. I've tried to group things logically and have added hyperlinks so you can quickly jump to other sections.

As this document evolves, sections that have changed since the last update will be highlighted in yellow.

The version number of this manual will follow the latest released version number of the firmware.

Feel free to contact me if you have question about a certain feature or have ideas for future improvements. I love to get feedback on my work. My email address is good on [QRZ.com](https://www.qrz.com).

Change Log

v1.01.00b and v1.01.00c: May 10 and 12, 2024

- Rewrote parts in the manual to clarify the configuration requirements of the *MIDI* and the app.

v1.01.00a: May 8, 2024

- Updated the [Appendix B: Installing and Updating CTR2-MIDI Firmware](#) section to include new information about flashing firmware to the ESP32-S3 in CTR2-MIDI.

v1.01.00: April 29, 2024 – Major Update Release

- Added USB MIDI support
 - Once v1.01.00 firmware has been installed in CTR2-MIDI you'll need to follow **Step 1** in [Appendix B: Installing or Updating CTR2-MIDI Firmware](#) section for future updates.
- Select from MIDI **Button**, and two types of **Sliders** and **Wheels** for each encoder function in the terminal page
- Redesigned terminal page
 - Hotkeys select the MIDI function for each encoder function
 - Displays saved ADC count assigned to each MFB
 - Displays ADC count when an MFB is pressed
 - Displays MIDI channel, control, and value when a control is executed
- Added [Appendix C](#) and [D](#) with information on using Tera Term and Putty

Changes to previous versions can be found in [Appendix E](#).

What is CTR2-MIDI?

Background

CTR2-MIDI (referred to as *MIDI* in this document) was derived from [CTR2-Micro](#) (referred to as *Micro* in this document), a full featured radio controller and memory keyer in a very small form factor. The *Micro* can control a wide range of radios but it has a limited user interface and generally requires a computer, tablet, or cell phone to display its settings. It can be used with [CTR2-Voice](#) to provide an audible user interface for blind and sight-limited operators. The *Micro* has some limitations when used to control a Flex radio remotely because it doesn't support SmartLink and requires port 4992 on the Flex radio to be forwarded through the station's router, creating a security risk if not managed properly.

HINT: An internet controlled power switch is a great way to isolate your radio from the network when you have port 4992 forwarded.

Since the *Micro's* introduction many users asked if it was possible to connect the *Micro* directly to, and control, another radio control program such as Marcus Roskosch's SmartSDR or SDR-Control for iOS apps. Doing this would solve the security issues with the Flex radio and provide a better Flex and Icom user experience because of the more robust user interfaces.

A New Product is Born

Marcus's apps and several other radio control apps (such as Thetis and SDR-Console) support MIDI controllers. MIDI controllers are relatively inexpensive controllers used by musicians to control musical instruments. It made sense to me that a small MIDI controller the size of the *Micro*, designed for a ham's needs, would be useful with these programs.

The *Micro's* design seemed to be a natural fit since it was small and inexpensive. Unfortunately the ESP32-C3 controller I chose for the *Micro* doesn't support MIDI on its USB connection to the computer. However, it does support MIDI over Bluetooth LE (BLE), so that is good. The only problem was that when I added the BLE library to an already loaded, relatively slow (160 MHz) single-core processor it was more than the *Micro* could handle.

A better solution was to use the same form factor as the *Micro* but add three additional pushbuttons, more status LEDs, and use an ESP32-S3 processor. The -S3 variant supports USB MIDI in addition to Bluetooth-LE MIDI. It has dual-cores and runs at 240 MHz. CTR2-MIDI was born!

NOTE: You can run the [CTR2-MIDI firmware on a CTR2-Micro](#) using Bluetooth-LE MIDI.

Hardware

The *MIDI* is designed to use the same 60mm x 60mm x 20mm enclosure and aluminum knob that is used with the *Micro*. This enclosure is the perfect size for a busy operating desk or for portable operation.

As shown in the photo, the encoder knob is the predominant feature on the face of the unit. When oriented with the USB-C connector and paddle jack on the edge away from you, the two LEDs in the upper left corner show the encoder mode with the lower LED having a value of 1 and the upper LED having a value of 2. Both LEDs will be off when the encoder is in its Home mode. Short-pressing the encoder steps you through mode 1, 2, 3, and back to Home. The LEDs indicate the current mode. In the photo above the encoder is in mode 1.



The green LED flashes once every 2 seconds when the unit is first powered up. It changes to two flashes every second when the unit is connected via Bluetooth-LE to the app. **It is on when you enter the [Extended](#) paddle mode.**

The multi-function buttons (MFBs) are laid out around the circumference of the knob. MFB1 is on the upper left, just below the encoder mode LEDs. The buttons are numbered counter-clockwise around the knob ending with MFB6 on the upper right. Short and long-press functions allow you to control two MIDI **Button** functions with each button.

NOTE: The button layout is different from the buttons on the *Micro* where MFB1 is the bottom-left button.

You'll find the USB-C and 3.5mm (1/8") stereo Paddle Input jack on the top (or back) edge in the photo.

Buzzer Volume Control Option

CTR2-MIDI does not come with a volume control for the internal buzzer. Provision has been made on the PCB board to install one if you desire. Just cut JP1 and add RV1, a [50 kOhm potentiometer](#) to the board. You'll also need to drill a 1/8" hole in enclosure for access.

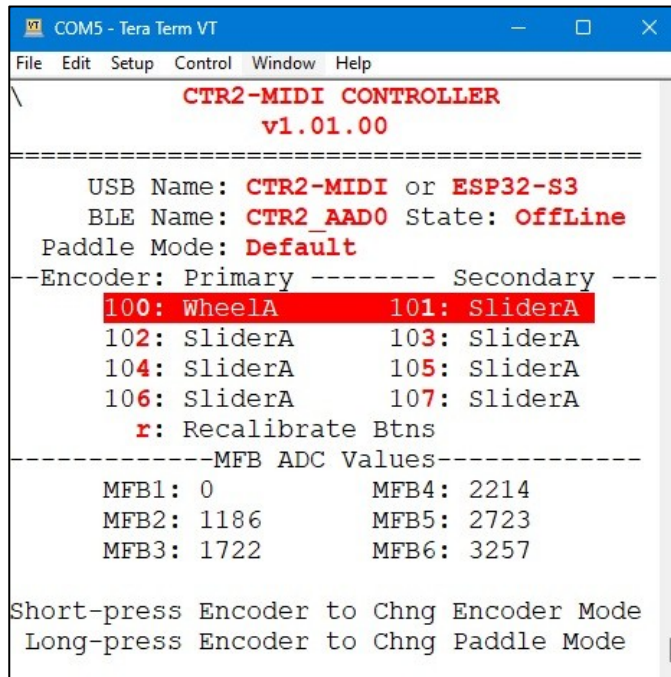
You can also just cut JP1 to disable the buzzer if you don't like it.

Configuring the MIDI Firmware for First Use

*MIDI*s come with the encoder functions pre-configured for use with SmartSDR or SDR-Control for iOS as shown in the screenshot here. A description of these functions can be found [here](#). If you want to change the MIDI control settings or are going to use it on any other app you'll need to change the encoder's configuration using a terminal program such as Putty or Tera Term.

To change these settings connect the terminal to the USB virtual serial port created for the *MIDI* when you connected it to your computer. Information on doing this can be found in [Appendix C](#) and [D](#).

Once you connect to the *MIDI* press any key on the terminal's keyboard to open the *MIDI*'s terminal display.



```
COM5 - Tera Term VT
File Edit Setup Control Window Help
CTR2-MIDI CONTROLLER
v1.01.00
-----
USB Name: CTR2-MIDI or ESP32-S3
BLE Name: CTR2_AAD0 State: OffLine
Paddle Mode: Default
--Encoder: Primary ----- Secondary ---
100: WheelA          101: SliderA
102: SliderA         103: SliderA
104: SliderA         105: SliderA
106: SliderA         107: SliderA
  r: Recalibrate Btns
-----MFB ADC Values-----
MFB1: 0              MFB4: 2214
MFB2: 1186           MFB5: 2723
MFB3: 1722           MFB6: 3257
Short-press Encoder to Chng Encoder Mode
Long-press Encoder to Chng Paddle Mode
```

After you've configured the *MIDI*'s encoder functions with a terminal you'll need to map the *MIDI*'s controls in your app. This is how you tell your app what you want each function on the *MIDI* to do. Mapping is covered in the [Mapping](#) sections below.

Terminal Display

The various fields in the terminal display are described next.

Device Names

When connecting the *MIDI* to the device running the radio control app using the [USB-C connector](#) it will show up on the device using one of two device names. On Windows it shows up as **CTR2-MIDI**. On Apple iOS and MacOS devices it shows up as **XIAO_ESP32-S3**.

When you [connect to an Apple device using Bluetooth-LE MIDI](#) the *MIDI* will show up as **CTR2_** plus a four digit hexadecimal number derived from the *MIDI*'s MAC address. This naming convention gives every *MIDI* a different Bluetooth-LE device name so you can use more than one *MIDI* on an Apple device (but not at the same time!).

The BLE **State** on the terminal display will show the connection status of the Bluetooth-LE MIDI connection. The green LED on the *MIDI* will flash twice once a second when the *MIDI*'s Bluetooth-LE is online.

NOTE: BLE MIDI is not supported on Windows.

Paddle Mode

The *Default* paddle mode sends MIDI controls 20 and 21 for the left and right paddles. These are usually mapped to the *Trigger CW Left paddle* and *Trigger CW Right paddle* functions in the app. When the *Extended* paddled mode is selected (by long-pressing the encoder in the encoder's Home mode) MIDI controls 30 and 31 are sent with the left and right paddles. These can be mapped to the *Trigger CW Straight Key* and *PTT Push* functions in the app.

Configuring Encoder Modes

The eight modes available on the encoder are shown next in the display. The primary mode controls (executed when you turn the encoder) are shown in the left column and are even numbered starting at 100 (the MIDI **Control Change** command number). The secondary controls (executed when you press and turn the encoder) are shown in the right column. They are odd numbered starting at 101.

NOTE: The highlight bar indicates the currently selected encoder mode.

Each encoder mode supports one of the following MIDI control types. To change the MIDI control type assigned to each encoder mode press the highlighted hotkey (the red digit in the control #) on the terminal's keyboard.

- **Button** – in this mode turning the encoder CCW or CW sends a MIDI **NoteOn {Button #}** for each encoder tick based on the table below. There are 24 encoder ticks per revolution. These commands can be used to change the frequency on SmartSDR and SDR-Control when they are mapped to the **Button Frequency Down** and **Frequency Up** in the app.

Primary Encoder Mode	CCW Button #	CW Button #	Secondary Encoder Mode	CCW Button #	CW Button #
100	40	41	101	42	43
102	44	45	103	46	47
104	48	49	105	50	51
106	52	53	107	54	55

- **SliderA** mode emulates a potentiometer. It sends a MIDI **Control Change** (aka **CC**) command for each encoder tick to the app indicating the relative position of the slider. Turning the encoder CCW decrements the count value down to 0 and turning the encoder CW increments the count value up to 127. The *MIDI* remembers the last count position through power cycles. This mode is used for controls that would use physical knobs or sliders such as volume, squelch, power, etc.
- **SliderB** is similar to **SliderA** except it adds “audible detents”. The *MIDI* will beep once at the center (64 counts) and twice at the lower and upper limits of the control (0 and 127 counts). Encoder changes are blocked for 600 milliseconds after the center beep to give you time to stop turning. **SliderB** is useful when using the encoder for RIT and XIT tuning as the center beep indicates when RIT or XIT is turned off and the edge beeps indicate when you have reached the end of the tuning range.

- **WheelA** is similar to the **Slider** controls in that it uses a MIDI **CC** command to send a value to the app. This control has an imaginary center position at 64 counts. Turning the encoder CCW sends a value below 64 for each encoder tick. Turning it CW sends a value above 64 for each encoder tick. This control is speed sensitive. Turning the encoder slowly sends a value of +/- 1 count from center. Turning it moderately fast (one revolution every two seconds) sends a value of +/- 10 counts from center. Turning it fast (about 1 revolution per second) sends a value of +/- 50 counts from center. ***SmartSDR and SDR-Control for iOS and MacOS use this scheme for frequency control and the frequency change per count depends on the Tune Step selected in the app.***
- **WheelB** is a variant of **WheelA**. Unlike **WheelA** this control has no center position, it only tells the app which direction the encoder is being turned. Turning the encoder CCW sends a value of 126 for each encoder tick. Turning it CW sends a value of 1 for each tick. This control is not sensitive to encoder turning speed. ***Thetis and SDR-Console use this scheme for frequency control. Frequency change per tick is based on the Tune Step set in the app.***

Calibrating MFB ADC Counts

The *MIDI* uses a resistive ladder to derive which multi-function button (MFB) is pressed. The ranges are pretty wide and these should not need to be recalibrated in the field. In the event something changes beyond the normal range or the buttons don't respond properly you can recalibrate them.

Pressing [r] on the terminal keyboard opens the **MFB Recalibration** display. You will be prompted to press each button in sequence, starting with MFB1 (the upper most button on the left side of the encoder knob) and progressing counter-clockwise about the encoder knob. This allows the unit to record the voltage each button presents to the ADC (A/D converter).

NOTE: Each button must be pressed within 10 seconds or the process will fail.

These voltages are specific to your unit and the ADC counts are displayed in the **MFB ADC Values** section of the terminal display. When you press a button its ADC count will be displayed below the listed values. This gives you a way to check button calibration. On the *MIDI*, each button's value must be no greater than 250 counts above the calibrated value. When running the *MIDI* firmware on the *Micro* the allowable overvalue is only 60 counts.

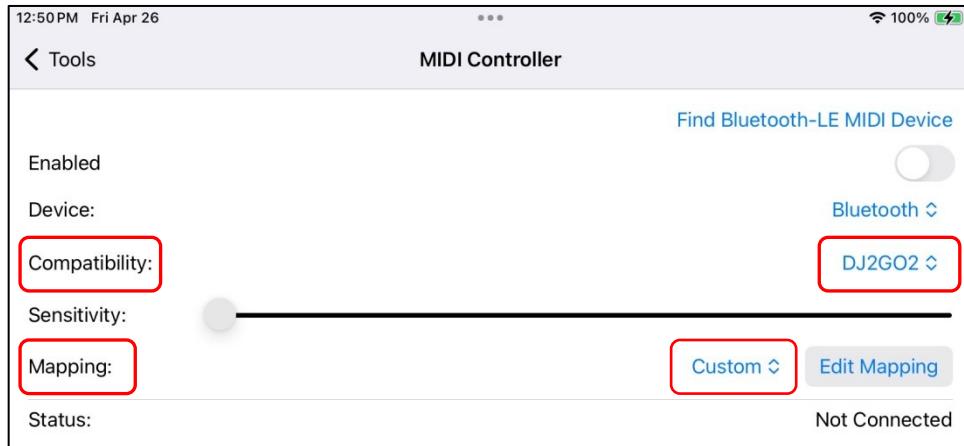
NOTE: If the *MIDI* loses the MFB calibration data it will sound "?" in Morse code when you press any MFB or press the encoder. To recalibrate the button voltages connect a terminal program to the *MIDI's* virtual USB serial port then press [r] to start the calibration process.

Connecting CTR2-MIDI to the App

To use the *MIDI* with your app you must first configure the app to use it.

SmartSDR and SDR-Control for iOS or MacOS

In these programs, open the **MIDI Controller** window in the **Tools** menu.



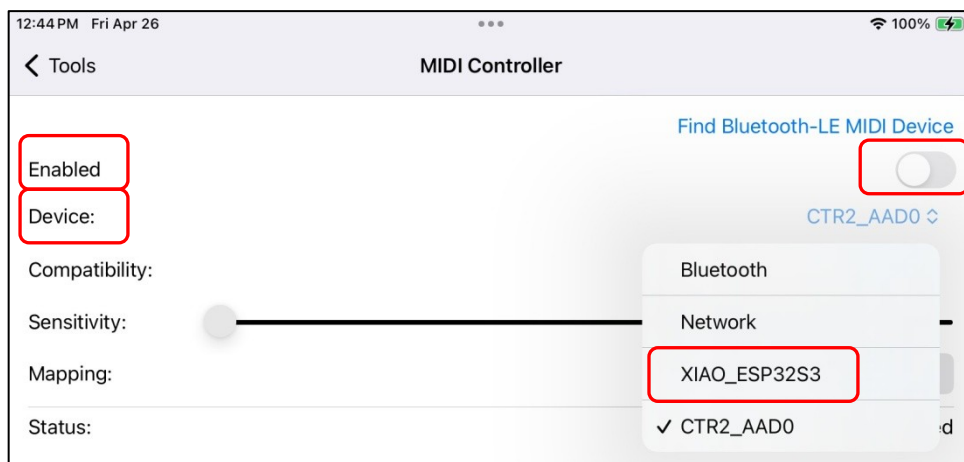
Change the **Compatibility** setting from **CMD Micro** to **DJ2GO2** and change the **Mapping** from **CMD Micro** to **Custom**. Without these changes the app won't connect to the *MIDI*.

USB MIDI

If using a USB connection you only need to select **XIAO_ESP32-S3** on the **Device** list and activate the **Enabled** button. The **Status** indication should then show **Connected**. Once connected you must [map the app's functions](#) to your *MIDI*.

NOTE: USB MIDI requires a USB-C data cable. USB-C power cables will not work!

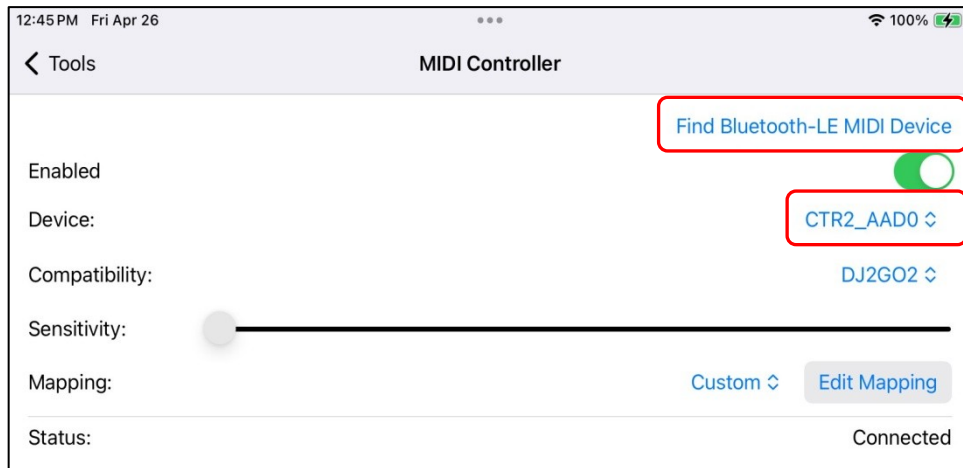
NOTE: iPad and iPhones require an OTG adapter in addition to the USB-C data cable.



Bluetooth-LE MIDI

Connecting with Bluetooth-LE is a little more involved.

Each *MIDI* has a unique Bluetooth-LE device name that starts with **CTR2_** followed by the last four digits of that unit's MAC address. To connect the *MIDI* to your app you must first click the **Find Bluetooth-LE MIDI Device** button to tell the program to look for your *MIDI*. A popup window should indicate the *MIDI* has been found. Next, select **CTR2_xxxx** in the **Device** list then slide the **Enabled** switch to the right. You will need to do this each time you start the app or reboot the *MIDI*.



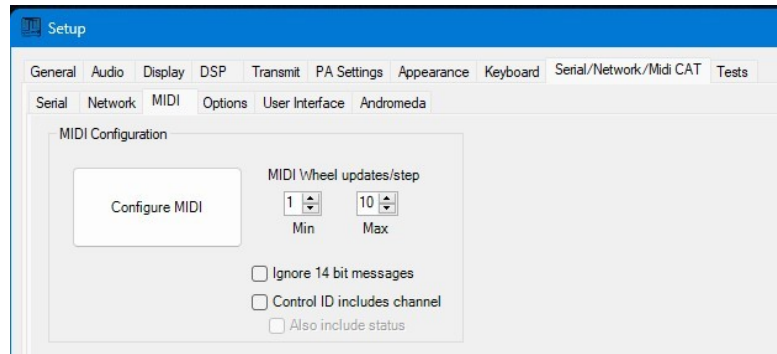
If **CTR2_xxxx** does not appear on the **Device** list, or it doesn't connect when turning the **Enabled** switch **On** press the **Find Bluetooth-LE MIDI Device** button again. You may need to press this button several times with the **Enabled** switch in both the **On** and **Off** position before **CTR2_xxxx** will appear in the **Device** list. It may also help to completely exit out of the app then restart it to get the *MIDI* to appear on the device list the first time.

If you can't get the app to connect to *MIDI*, download **MIDI-Wrench** for iOS or **Conji** for the Mac from the App store. They are free apps used to troubleshoot BLE MIDI issues. Connect to the *MIDI* with one of these apps first and then try to connect to the *MIDI* again in the radio control app.

Once **Connected** appears on the **Status** line press the **Edit Mapping** button to enter [mapping mode](#).

Thetis on Windows

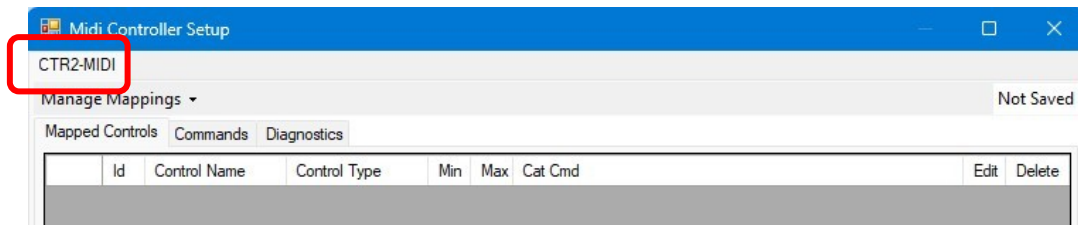
When connecting to any radio control app (such as Thetis or SDR-Console) running on Windows you must connect the *MIDI* using USB. Windows does not support Bluetooth-LE MIDI (at least not very well). Thetis will be used as an example here but connecting to and mapping controls in other apps should be similar.



NOTE: USB MIDI requires a USB-C data cable. USB-C power cables will not work!

Start by selecting the **Settings** menu on the main page then select the **Serial/Network/MIDI CAT** tab. Click the **MIDI** tab as shown above.

Next, click the **Configure MIDI** button to open the **Midi Controller Setup** page. The program should go through an initialization sequence and end up on this page if it finds the *MIDI* on USB. The tab at the top should be labeled **CTR2-MIDI** or possibly **ESP32S3**. If the initialization fails, double-check that you are using a USB-C data cable, not a charge cable and try again.



NOTE: You will need to open this window every time you start Thetis in order to initialize USB MIDI.

CTR2-MIDI Functions

The encoder functions were covered in the [Encoder Modes](#) section above. You must define the MIDI control you want to use for each encoder mode with a terminal program before you map the MIDI controls to your app.

NOTE: The *MIDI* always sends commands on MIDI Channel 1.

Multi-Function Buttons (MFBs)

MFBs are laid out counter-clockwise around the encoder knob starting with MFB1 on the upper left and ending with MFB6 on the upper right of the knob.

MFBs send the MIDI **NoteOn** command when the button is released.

Each button has two functions defined by the amount of time you hold the button down.

Short-pressing (i.e. less than 1 second) a button produces one beep when pressed and executes a MIDI **NoteOn** command when released. The **NoteOn** control # will be from 1 to 6 depending on which button was pressed.

Long-pressing (holding a button for longer than 1 second but less than 3 seconds) produces one beep when pressed and a second beep after the long-press timeout expires. Upon button release a **NoteOn** command is sent. The **NoteOn** control will be from 11 to 16 depending on which button was pressed.

NOTE: Holding a button longer than 3 seconds cancels the button press and sounds a third low frequency beep.









MIDI **Button** controls can be assigned on the app to toggle commands in the app's map such as *Band Up*, *Band Down*, *Mode Up*, *Mode Down*, etc.

NOTE: The *MIDI* does not receive updates from the controls it's mapped to in the app. If you change a value with the *MIDI*, then change it manually in the app, the next time you change it in the *MIDI* it will send the new value based on the last value the *MIDI* sent, not the value you manually set in the app.

Encoder Mode Indications



Encoder modes are selected by short-pressing (< 1 second) the encoder switch. The current encoder mode is indicated by the yellow LEDs at the top-left of the unit. Each encoder mode has a primary “turn” control and a secondary “press and turn” control. The table in the [Encoder Modes](#) section describes each option.

Long-pressing the encoder while in Mode 1, 2, or 3 will return you back to the encoder Home mode (all LEDs off) and sounds a single high frequency beep. Long-pressing the encoder in the encoder Home mode toggles the **Paddle Mode**.

Encoder Mode	LED	Beep Tone on Selection	Description
Home	 	Single high frequency beep	Encoder Turn = Control Change 100 Press & Turn = Control Change 101
1	 	Single low frequency beep	Encoder Turn = Control Change 102 Press & Turn = Control Change 103
2	 	Two low frequency beeps	Encoder Turn = Control Change 104 Press & Turn = Control Change 105
3	 	Three low frequency beeps	Encoder Turn = Control Change 106 Press & Turn = Control Change 107

Paddle Mode

The *MIDI* has two **Paddle Modes**, **Normal** and **Extended**. These modes allow you to map different functions to the paddle input jack on the *MIDI* depending on your needs. The **Paddle Mode** is indicated by the green LED. The table below explains the two modes.

Paddle Input	Green LED	Beep Tone on Selection	Description
Normal Mode		Two tone High>Low beep	Left paddle controls NoteOn 20 , Right paddle controls NoteOn 21 Map these to Trigger CW Left Paddle and Trigger CW Right Paddle
Extended Mode		Two-tone Low>High beep	Left paddle controls NoteOn 30 , Right paddle controls NoteOn 31 You can map these to Trigger CW Straight Key and PTT Push to give you a straight key and PTT control in this mode. You can map 30 and 31 the same as functions as 20 and 21 to disable the extended mode

To toggle **Paddle Mode**, long-press the encoder while in the Home encoder mode (both yellow LEDs off).

NOTE: The green LED flashes once every two seconds when the unit is first powered up. It changes to twice every second when the unit is connected to the app using Bluetooth-LE.

If your app doesn't support keyer or PTT input control, or you don't use them, you can map the paddle controls to any other function your app supports.

Mapping in SmartSDR and SDR-Control for iOS and MacOS

Each *MIDI* control must be mapped to a function in the app so it knows what to do with that control. This section will describe how this is done in SmartSDR and SDR-Control for iOS and MacOS.

Multi-function buttons (MFBs) can be mapped to change modes or bands, or enable features such as antenna tuners or filters. The eight encoder controls can be mapped to change values such as frequency, volume, RIT, XIT, RF Power, etc.

By default the configuration for the *MIDI* defines the first primary encoder control (**CC 100**) as **WheelA**. This control is typically used for frequency control. Other encoder functions are defined as MIDI **SliderA** (no center/edge beeps). You can [change these assignments](#) using a terminal program.

Importing the CTR2-MIDI_vxxx.map file

A copy of the map I use for SmartSDR and SDR-Control for iOS and MacOS is included in the *MIDI*'s firmware ZIP file. To import a map you must allow the app to use the iCloud drive on your device. Once this is done a folder for the app is created on the iCloud drive. Copy the *CTR2-MIDI_vxxx.map* file to this folder then click the **File** icon in the **Edit MIDI Controller Map** screen and select **Import Mapping from iCloud Drive**. You can also export your settings to this drive to save a backup copy.

NOTE: There is a bug in the app. Once you have imported a map you must edit one of the control settings before the app will save the map settings otherwise the settings will be lost when you navigate away from the MIDI tool.

Editing the MIDI Controller Map

Pressing the **Edit Mapping** button in the **MIDI Controller** page opens the **Edit MIDI Controller Map** page.

The map will be empty when you open it the first time. If not, make sure **Mapping** is set to *Custom* on the MIDI Controller page.



Registering Controls

To register the MFBs on the *MIDI*, short-press and long-press each MFB to add it to the **Buttons** list.

Next, turn the encoder to register the primary event of the current encoder mode in the **Wheels/Sliders** list. Once this is done, press and turn the encoder to register the secondary event. Next, short-press the encoder to move to the next encoder mode (yellow LEDs will change) and register its primary and secondary events. Continue this process until you have registered all four encoder modes for a total of 8 slider controls.

When you're done you should have the short-press MFBs registered for **Buttons 1 to 6**, and long-press MFBs registered for **Buttons 11 to 16**. The 8 encoder functions should be registered to **Wheels/Sliders 100 to 107**.

Mapping Functions to Controls

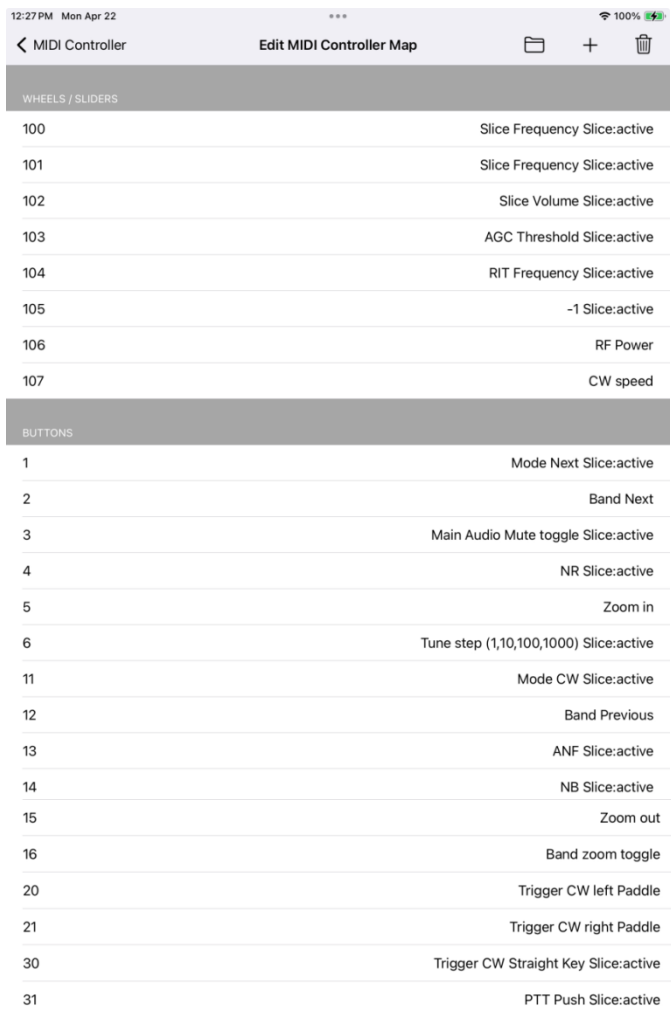
Next, map a function to each control. To do this, press or click on each control line and choose a function from the dropdown menu.

The screenshot at the right shows the mapping I'm currently using on my unit. I map similar items to the same button's short and long press.

NOTE: Your *MIDI* will not be able to control your app until these functions are mapped!

For instance: Short-pressing MFB1 on the *MIDI* sends the **Button 1** command. This is mapped to *Mode Text Slice: active* in SmartSDR which increments the mode on the active slice. Long-pressing MFB1 on the *MIDI* sends **Button 11**. This is mapped to *Mode CW Slice: active* in SmartSDR which tells it to set the radio's mode to CW.

Once you have your map configured press the folder button at the top of the display to save your map.



The screenshot shows the 'Edit MIDI Controller Map' interface. It is divided into two sections: 'WHEELS / SLIDERS' and 'BUTTONS'. Each section contains a list of controls with their corresponding mapped functions.

Control ID	Mapped Function
WHEELS / SLIDERS	
100	Slice Frequency Slice:active
101	Slice Frequency Slice:active
102	Slice Volume Slice:active
103	AGC Threshold Slice:active
104	RIT Frequency Slice:active
105	-1 Slice:active
106	RF Power
107	CW speed
BUTTONS	
1	Mode Next Slice:active
2	Band Next
3	Main Audio Mute toggle Slice:active
4	NR Slice:active
5	Zoom in
6	Tune step (1,10,100,1000) Slice:active
11	Mode CW Slice:active
12	Band Previous
13	ANF Slice:active
14	NB Slice:active
15	Zoom out
16	Band zoom toggle
20	Trigger CW left Paddle
21	Trigger CW right Paddle
30	Trigger CW Straight Key Slice:active
31	PTT Push Slice:active

NOTE: After mapping the **Paddle Modes** for **Buttons 30 and 31** make sure you turn **Extended Paddle Mode** off (long-press the encoder) if you want to use the normal paddle mode for keying the keyer.

Using Paddles with the App's Keyer

The SmartSDR for iOS app includes a keyer that you can control with the paddles connected to the *MIDI*.

NOTE: If you're using Bluetooth-LE MIDI you may notice a difference in how the paddles work with the app's keyer as opposed to a hardwired paddle/keyer. The latency in the Bluetooth-LE connection causes up to 15 milliseconds of delay. This will throw your "fist" timing off, especially in lmbic modes where an extra element may be added. It may take some practice to get use to the timing change. If this is a problem for you consider using USB MIDI instead.

To control the keyer, follow these steps:

- Map **Button 20** to **Trigger CW Left Paddle** and **Button 21** to **Trigger CW Right Panel**. You can change this mapping if they are wired backwards.
- Plug your paddles into the **Paddle In** jack on CTR2-MIDI.
- **Verify that the Paddle Mode is *in Normal mode*** on the *MIDI* (the green LED is off). To toggle this mode, long-press the encoder while in the Home encoder mode (both yellow LEDs off). You should hear two tones (high to low frequency) and the green LED will turn off. If **Paddle Mode** is in **Extended** mode the paddles will control MIDI **Buttons 30** and **31**.
- Set the radio's mode to CW
- Press the **View** menu in SmartSDR's Panadapter display and select **CWX Panel**. You can adjust speed and other setting here.
- Pressing either paddle will now activate the keyer and key the radio.

NOTE: There is a bug in the app that causes it to buffer paddle inputs when the radio is not in CW mode. This causes the radio to start sending the last paddle input (a Dit or Dah) repeatedly when you change to CW mode until you press that paddle again to turn it off.

Using a Straight Key or PTT Switch

If you want to use a straight key or an external PTT switch with the *MIDI*, map **Button 30** or **31** to **Trigger CW Straight Key** and **PTT Push** then wire the straight key and PTT switch to the Tip and Ring of a stereo plug and insert it into the **Paddle In** jack on the *MIDI*.

Long-press the encoder while in the Home encoder mode (yellow LEDs off) to toggle the **Paddle Mode** between **Normal** to **Extended**. You should hear two tones (low to high frequency) and the green LED will light when in **Extended** paddle mode.

Mapping Functions in Thetis

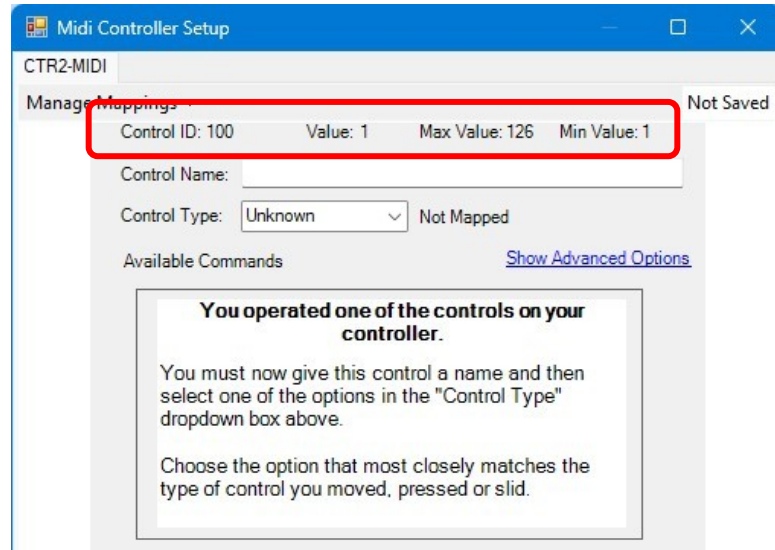
Thetis is a little different than SmartSDR and SDR-Control for iOS and MacOS in that it uses **WheelB** instead of **WheelA** for frequency tuning. You will need to use a terminal program to change the **WheelA** command to **WheelB** for **CC 100** (or or the **CC** you want to use for tuning).

With the **Setup** ->

Serial/Network/Midi CAT window open select the **MIDI** tab and click the **Configure MIDI** button. The program will initialize **MIDI** using USB.

NOTE: If the program doesn't initialize the **MIDI** check to make sure you are using a USB-C data cable, not a USB-C charge cable.

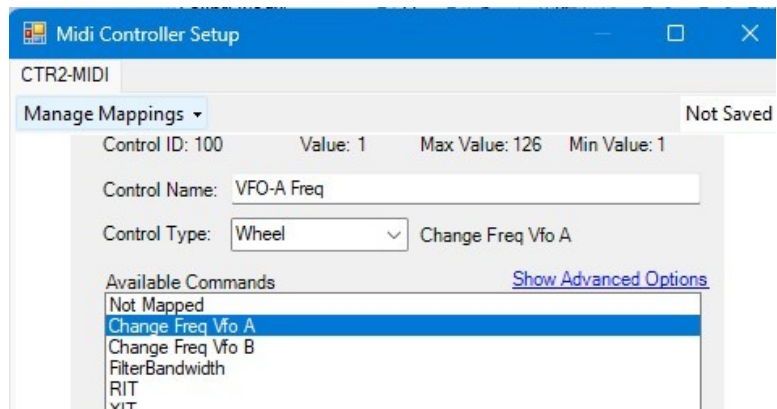
Press a button or turn the encoder and the window at the left will open.



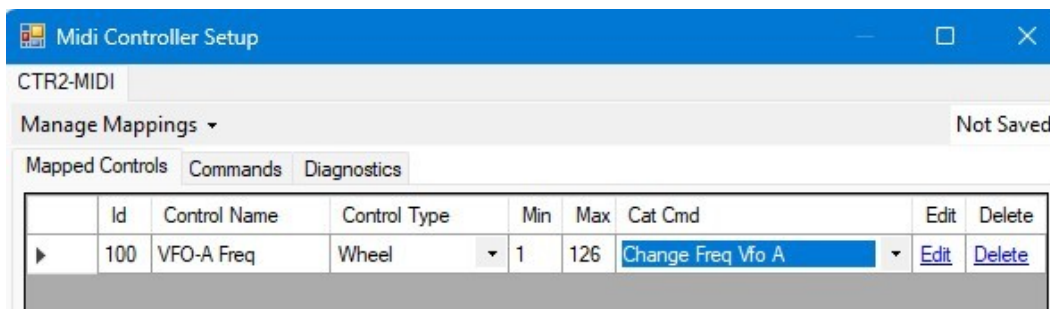
In this example I turned encoder when the **MIDI** was set to the Home encoder mode so it is controlling **Change Control (CC) 100**.

NOTE: You must turn the encoder fully CW and CCW to set the minimum and maximum values in this window. When set to **Slider** you need to turn CCW until you get to 1 and CW until you get to 127.

Next, I give the control a name. I'll call it **VFO-A Freq** and select **Change Freq VFO A** from the **Available Commands** list.



Click the **Done** and then the **Save** button to return to the **Manage Mappings** page.



You've mapped your first control! Now follow the same procedure to the other 7 encoder functions and the 12 button functions.

Here's a screenshot of the mapping I currently have in Thetis. Don't be afraid to experiment with the settings so you can find the combination that works best for you.

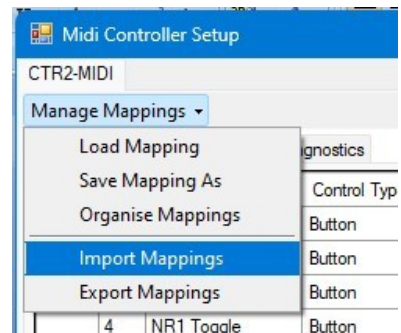
	Id	Control Name	Control Type	Min	Max	Cat Cmd	Edit	Delete
▶	1	Mode Up	Button	0	127	Rx1 Mode Next	Edit	Delete
	2	Band Up	Button	0	127	Band Up	Edit	Delete
	3	Mute	Button	0	127	Mute On Off	Edit	Delete
	4	NR1 Toggle	Button	0	127	Noise Reduction 1 On Off	Edit	Delete
	5	Swap VFOs	Button	0	127	Toggle Wheel to VFOA/VFOB	Edit	Delete
	6	Tuning Step Up	Button	0	127	Tuning Step Up	Edit	Delete
	11	Mode Down	Button	0	127	Rx1 Mode Prev	Edit	Delete
	12	Band Down	Button	0	127	Band Down	Edit	Delete
	13	Notch Toggle	Button	0	127	Auto Notch On Off	Edit	Delete
	14	NR2 Toggle	Button	0	127	Noise Reduction 2 On Off	Edit	Delete
	15	VFO Swap	Button	0	127	Vfo Swap	Edit	Delete
	16	Tuning Step Down	Button	0	127	Tuning Step Down	Edit	Delete
	100	VFO-A Freq	Wheel	1	126	Change Freq Vfo A	Edit	Delete
	101	Vol VFO-A	Knob or Slider	0	127	Volume VfoA	Edit	Delete
	102	VFO-B	Wheel	0	127	Change Freq Vfo B	Edit	Delete
	103	Vol VFO-B	Knob or Slider	0	127	Volume VfoB	Edit	Delete
	104	RIT	Knob or Slider	0	127	RIT	Edit	Delete
	105	XIT	Knob or Slider	0	127	XIT	Edit	Delete
	106	Drive	Knob or Slider	0	127	DriveLevel	Edit	Delete
	107	Zoom	Knob or Slider	0	127	Zoom	Edit	Delete

NOTE: Make sure you run the encoder to the full CCW and CW directions so the minimum and maximum values are recorded. This is especially important on **Slider** controls that have the full range from 0 to 127 counts.

When you're finished make sure to click the **Save** button in the **Midi Controller Setup** page and **Apply** before you leave the **Setup** page so these functions will be restored the next time you run Thetis.

You can export your map or import a map from someone else by clicking the **Manage Mappings** tab at the top of the **Midi Controller Setup** display. I've included a copy of my map in the firmware zip file.

NOTE: You will need to open the **Setup -> Serial/Network/MIDI CAT -> Configure MIDI** window every time you start Thetis in order to initialize USB MIDI.



Advanced Options

Operation with N1MM Logger+

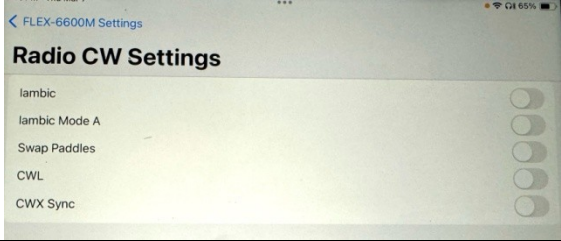
You can use the *MIDI* as an interface to use N1MM's CW keyer with SmartSDR or SDR-Control for iOS. To do this, follow the instructions on the [N1MM Interfacing Basics](#) webpage to build a key interface circuit. This will require a USB serial adapter with a keying interface circuit wired to its RTS output. Connect the output of the keying interface circuit to the TIP and SHIELD of the *MIDI*'s Paddle input jack. Next, map the Left Paddle input on the *MIDI* to **Trigger CW Straight Key**. With this setup you can control your radio using the N1MM Logger+ and use keyboard keying in N1MM Logger+.

Unfortunately this configuration doesn't allow you to use a set of paddles with the *MIDI* at the same time nor does it allow N1MM Logger+ to control your radio remotely.

If you're looking to have N1MM Logger control your radio remotely you might consider using the *Micro* to remotely control and key your radio with SmartSDR or SDR-Control providing the user interface. Instructions for this can be found in the [CTR2-Micro Operation Manual](#) under the **Advanced Options** section.

Troubleshooting

Issue	Solution
I can't connect the app to the CTR2-MIDI using USB MIDI	<ol style="list-style-type: none"> 1) Make sure you are using a USB-C <u>data</u> cable. Many USB-C cables are <u>power only</u>. These will not work with USB MIDI. 2) If using a mobile device (iPhone, iPad, Android, etc) you must use an OTG adapter for your device. This adapter provides a USB connection for an external device.
I can't connect the app to the CTR2-MIDI BLE device	Follow the instructions in the Connecting the App to CTR2-MIDI section. You may need to click both the Find Bluetooth-LE MIDI Device and the Enabled switch several times in different combinations to get the app to find the <i>MIDI</i> . You also may need to install the MIDI-Wrench or Conji app to get CTR2-MIDI recognized. If all else fails, connect using USB.
I have to connect CTR2-MIDI every time I start the app.	CTR2-MIDI will stay connected the app on your iOS or Mc device with Bluetooth-LE as long as it remains powered up. If you remove power from CTR2-MIDI you'll need to open the Tools-MIDI Controller window and click the Find Bluetooth-LE MIDI Device button then click Enable to reconnect it to the app. When connecting with USB MIDI it should say connected even through a power cycle.
Radio starts sending a string of Dits or Dahs when I select CW mode	This seems to be a bug in the app. If you press the paddles when the app is not in CW mode but the CWX panel is open that paddle press is buffered in the app. When you change to CW mode from another mode CWX triggers the last paddle press you made and keeps sending it until you press the paddle again, turn off CW mode, or close the CWX panel.
The <i>MIDI</i> connects to my device but does not control anything	<ol style="list-style-type: none"> 1) Verify that you have mapped each MIDI control to a function in your app.
Frequency tuning is not working	<ol style="list-style-type: none"> 1) Use WheelA for SmartSDR and SDR-Control. Use WheelB for Thetis and other Windows apps. 2) Make sure you have the encoder mode set to the mode you programmed frequency control to. I use the encoder's Home mode for frequency tuning.
How can I tell when I'm at the center (0 Hz) when controlling RIT or XIT?	Use the SliderB control for RIT and XIT controls. This control beeps and pauses tuning for about ½ second when the control reaches its center position (64 counts) and beeps twice when reaching the upper or lower end of tuning. SliderA does not beep at center or at the edges. Its best for most other knob or dial controls.
Pressing an MFB executes the wrong parameter	First, verify that the MIDI mapping is correct. If it is, you will need to be recalibrate the MFB ADC counts. Connect a terminal to the unit's virtual serial port and select the [r] option to start the recalibration process.
Speed sensitive tuning is not working	The WheelA control mode used for frequency tuning in SmartSDR and SDR-Control supports speed sensitive tuning (tuning step changes as you tune faster). The WheelB control must be used on Thetis and other Windows apps. It does not support speed sensitive tuning.
Paddles don't key the radio	<ol style="list-style-type: none"> 1) Verify the radio is in CW mode 2) Verify the keyer is in Breakin mode and Sidetone in On in the P/CW panel 3) Verify the Paddle Mode is in Normal mode (green LED is off) 4) Verify MIDI Buttons 20 and 21 are mapped to Trigger CW Left Paddle and Trigger CW Right Paddle 5) Open the CWX Panel in the app's View menu

Paddles are reversed	Remap MIDI Buttons 20 and 21 to swap Left and Right paddle assignments
No sidetone in SmartSDR when keying	Open the left pop-out window in the Panadapter display on the app and click the Audio menu. Set the Local Audio Monitor slider to 0.
Slow response or timing issues with keyer	<p>Connecting multiple Bluetooth devices to your iOS device (i.e. CTR2-MIDI and a BT headset) may affect the app's keyer response to paddle input. To fix this problem, click the Flex 6xxx button on the bottom of the display, select the CW item to open the Radio CW Settings window and disable CWX Sync.</p> <p>You may also consider using USB MIDI with an OTG adapter on your mobile device.</p> 

Appendix A: Using CTR2-MIDI Firmware on CTR2-Micro

As mentioned above, the CTR2-MIDI firmware can be flashed to CTR2-Micro. This gives current *Micro* users access to the *MIDI* features without having to buy new hardware.

The terminal display for *MIDI* firmware running on the *Micro* is similar to the normal *MIDI* display except the **USB Name** is not shown because this mode is not available on the *Micro*.

There are several limitations to using the *MIDI* firmware on the *Micro*:

- CTR2-Micro hardware only supports Bluetooth-LE MIDI. The ESP32-C3 processor in the *Micro* doesn't have the hardware to support USB MIDI.
- Once you have flashed the *MIDI's* firmware on your *Micro* most of the features you use on the *Micro* will no longer be available. This includes the following:
 - WiFi is not supported so you can't control the Flex radio directly or use [CTR2-Voice](#)
 - Serial CAT is not supported so you can't control other radios with the *Micro*
 - The *Micro's* keyer is not supported so you can't use a terminal as a keyboard keyer or use any transmit message buffers you had set up in the *Micro*
 - Key and PTT Output is not supported
 - Favorite Frequency and Previous Frequency lists are not supported
 - The web browser interface is not supported
 - You must reflash the *Micro* firmware on your *Micro* to restore its normal features
- The *Micro* only has three buttons so you can only map 6 MIDI **Buttons** in the app
- The *Micro* only has one LED. This LED flashes to indicate the current mode of the unit

You can always reflash the *Micro* firmware on your *Micro* to return it to its normal functionality.

Flashing the Firmware

To flash the *MIDI* firmware on to your *Micro*, unzip the four binary files from the *MIDI* firmware distribution zip file into a unique folder then follow the instructions in [Appendix B](#).

You will need to complete the **MFB Calibration** on the *Micro* to save the MFB ADC counts in the *MIDI's* initialization file. This only needs to be done once.

NOTE: The *MIDI* firmware maintains its own file structure therefore the settings you had for the *Micro* firmware are not lost. Just reflash the *Micro* firmware to your *Micro* to restore your *Micro* to its normal operation.

```
COM13 - Tera Term VT
File Edit Setup Control Window Help
-----
CTR2-MIDI CONTROLLER
(running on CTR2-Micro)
v1.01.00
-----
BLE Name: CTR2 3AF8 State: Online
Paddle Mode: Default
--Encoder: Primary ----- Secondary ---
100: WheelA          101: SliderA
102: SliderA         103: SliderA
104: SliderB         105: SliderB
106: SliderA         107: SliderA
r: Recalibrate Btms
-----MFB ADC Values-----
MFB1: 2224
MFB2: 2376
MFB3: 2461
Short-press Encoder to Chng Encoder Mode
Long-press Encoder to Chng Paddle Mode
MIDI Ch: 1 Ctrl Chng: 100 Val:65
```


Operating the *Micro* with *MIDI* Firmware

The *Micro* operates exactly like the *MIDI* when running the *MIDI* firmware. You will need to go through the same steps to connect the *Micro* to your iPad and you will need to map the controls on the *Micro* to the app. You will only have 6 MIDI **Button** functions (two on each MFB) and all of the encoder modes.

The big difference between the *MIDI* and the *Micro* is with the LED indicators (or lack thereof). The *Micro*'s status is indicated by the flash sequence of the status LED.

The table below summarizes these flash sequences and the beeps associated with them.

Encoder Mode	Flash Sequence	Buzzer Tone when entering this mode	Description
Home	One long flash	One high frequency beep	Primary encoder sends Control Change (CC) 100 Secondary encoder sends CC 101
Mode 1	One short flash	One low frequency beep	Primary encoder sends CC 102 Secondary encoder sends CC 103
Mode 2	Two short flashes	Two low frequency beeps	Primary encoder sends CC 104 Secondary encoder sends CC 105
Mode 3	Three short flashes	Three low frequency beeps	Primary encoder sends CC 106 Secondary encoder sends CC 107

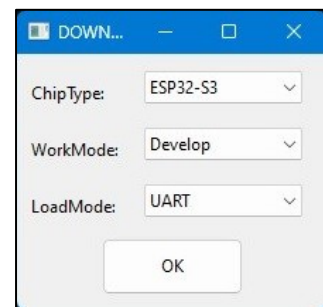
The **Paddle Mode** determines the rate of the LED flash.

- When **Paddle Mode** is in **Normal** mode (controlling MIDI **Buttons 20** and **21**) the LED flashes the status at 2 second intervals
- When **Paddle Mode** is in **Extended** mode (controlling MIDI **Buttons 30** and **31**) the LED flashes the status at 1 second intervals

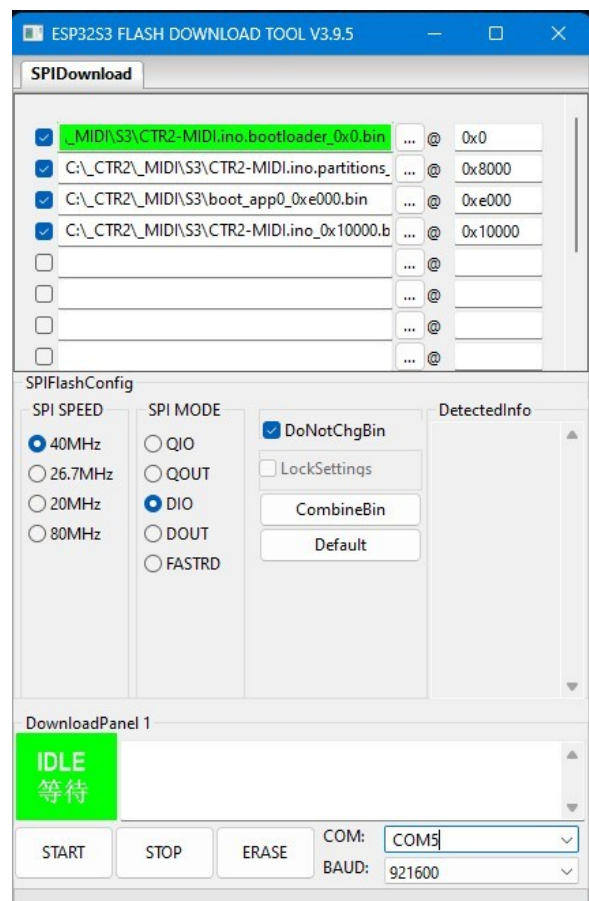
Appendix B: Installing or Updating CTR2-MIDI Firmware

Kits and assembled CTR2-MIDIs have the firmware already installed on them but as you might expect changes will be made to the program over time. To install the latest MIDI firmware on your device, follow these steps:

1. Determine which version of the firmware you need for your device. If your device is a CTR2-MIDI (with six MFBs) you'll need the **S3** firmware (for the ESP32-S3). If you are running CTR2-MIDI on a CTR2-Micro (with three MFBs) you'll need the **C3** firmware (for the ESP32-C3).
2. Download and unzip the appropriate CTR2-MIDI firmware from [my web site](#). Unzip it into a different folder than where you store the Micro's firmware update files.
3. Download and open the [Espressif Flash Downloader Tool](#). When it starts, select the **Chip Type** your unit is using (*ESP32-S3* for CTR2-MIDI or *ESP32-C3* for CTR2-Micro hardware). Leave **WorkMode** set to *Develop* and **LoadMode** set to *UART*.



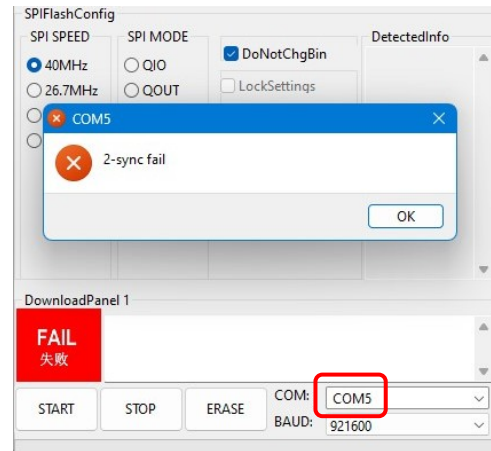
4. Map the four .BIN files in the CTR2-MIDI firmware distribution file into the downloader tool.
NOTE: The address for each file is embedded in its file name. Enter this address in the address field to the right as shown in this screenshot.
5. Select the checkboxes for the four files as shown.
6. Set the COM: port to the port assigned to CTR2-MIDI and set the Baud to 921600.



IMPORTANT NOTE FOR ESP32-S3

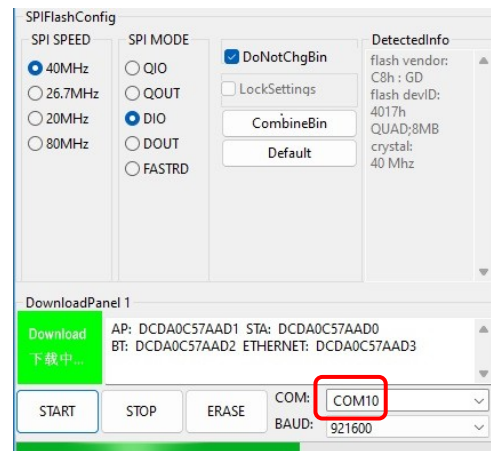
The ESP32-S3 processor in CTR2-MIDI has two USB UARTs. The first UART (COM 5 in this example) is active when you initially power up the unit and is the COM port you connect to with a terminal program. ***The Flash Download Tool will fail and pop up the window in this screenshot when you try to flash the program to this port.***

Press **OK** to clear the alarm window.



Next, drop down the **COM:** list again. You notice that the initial COM port (COM 5 in this example) is not on the list. You'll also notice a new COM port has been added to the list. On my computer it shows up as COM 10 (your computer will show a different port #). This is the second USB UART on the processor and is used for programming. Select this new COM port.

NOTE: The ESP32-C3 processor in CTR2-Micro only has one UART so the above steps are not required. Just select the COM port for the *Micro* when flashing CTR2-MIDI firmware to it.



7. Click the **START** button to start the download.
8. Once the download is complete, cycle the power on the unit to start the new MIDI firmware.

NOTE: The ESP32-S3 processor will revert back to its initial COM port (COM 5 in this example) after the reboot.

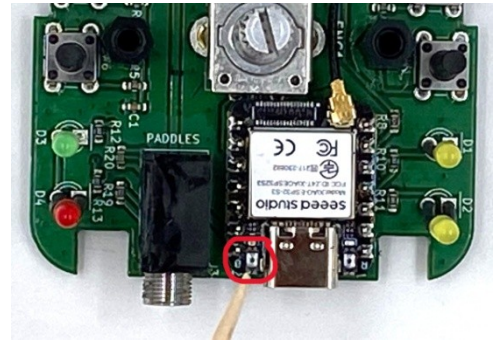
NOTE: Clicking the **ERASE** button will erase the entire flash memory including the setup file. This will reset the MFB ADC values saved on the unit and you will need to [recalibrate](#) the buttons when you start the program. If you are flashing the MIDI firmware to a CTR2-Micro (-C3 processor) erasing the flash will also delete the configuration files used with the normal *Micro* firmware.

Installing using Linux or Mac

A script file is also supplied in the firmware update zip file. This script file can be used in a Linux or Mac environment if you don't have access to a Windows computer.

Instructions for using this script file are include in the [CTR2-Micro Operation Manual](#) in **Appendix B**.

Unfortunately, there is no way to force the ESP32-S3 processor used in CTR2-MIDI to switch from the initial COM port to the programming COM port. This must be done by pressing and holding the **PROGRAM** button down on the processor board when applying power to the board. Yes, it's that little black dot on the square silver pad the toothpick is pointing to in this photo!



I added a small notch to the *MIDI's* enclosure that allows you to press this button without disassembling the enclosure. It's still hard to do especially if your eyes aren't as good as they use to be. Insert the toothpick only about 6mm (1/4") and gently press down. You will feel a light click. If you insert the toothpick too far it will rest on the chip LED in back of the switch and will not press the button. Once you have the button pressed power up the unit. If it's in programming mode the normal LED boot sequence will not run. Use a terminal program to find the program UARTs COM port assignment.

With all that said, it's probably easier to just find a Windows computer to do the update!

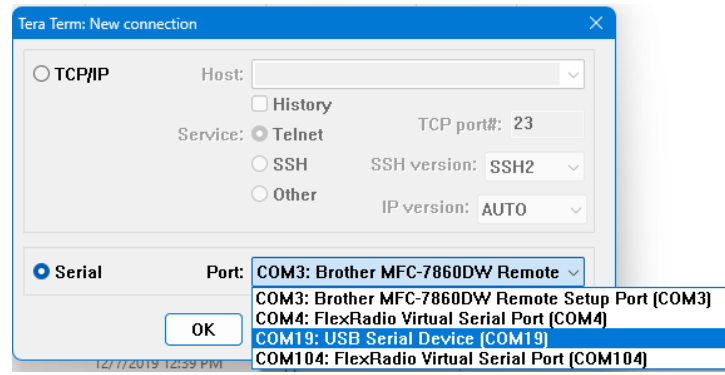


Appendix C: Configuring Tera Term

Tera Term is the simplest terminal program to get running for a serial connection.

Download it for Windows at <https://ttssh2.osdn.jp/index.html.en>. If that site is down, try https://download.cnet.com/tera-term/3001-2094_4-75766675.html. As far as I know, Tera Term is only available for Windows.

When you first open Tera Term you'll be presented with the **Tera Term New connection** window. Simply select the **Serial** radio button, select the COM port Window's assigned to your Micro when you plugged it in, and click the **OK** button.



Since you are connecting to a USB serial port there is no need to set the baud rate. It will run at USB speed regardless of the baud setting.

That's it! Tera Term will connect to the *MIDI*. Press any key to open the configuration display.

You can change the terminal size in the **Setup** menu. Select **Terminal...** Set the **Terminal Size** to 41 x 20. The *MIDI*'s terminal interface was designed for this size.

While in the **Terminal...** settings verify the **New-line** options are set to **CR** for both **Transmit** and **Receive** and the **Terminal ID** is set to **VT100**.

You'll probably want to change the font size and colors. These are also changed in Tera Term's **Setup** menu. Select **Display** to change the font and background colors to your liking. Select **Font** to change the font and font size. I like *Courier New, Regular, and 14 point size*. You're preferences may differ.

Once you have the program configured the way you like, select the **Setup->Save Setup...** menu and save your configuration. If you use the default file name, TERATERM.INI the program will automatically start a Telnet session using the COM port you selected above when it opens. This provides one-click access to your *MIDI*.

Appendix D: Configuring Putty

Putty is a terminal program that can be configured for a variety of needs. The *MIDI* only supports serial connections. This section describes how to configure the program to interface with the *MIDI*.

Download Putty for Windows from <https://www.putty.org/>. It's also available for Linux at <https://www.ssh.com/academy/ssh/putty/linux> and for Mac at <https://www.ssh.com/academy/ssh/putty/mac>.

You'll need to connect to the *MIDI* using its USB serial port in order to configure it.

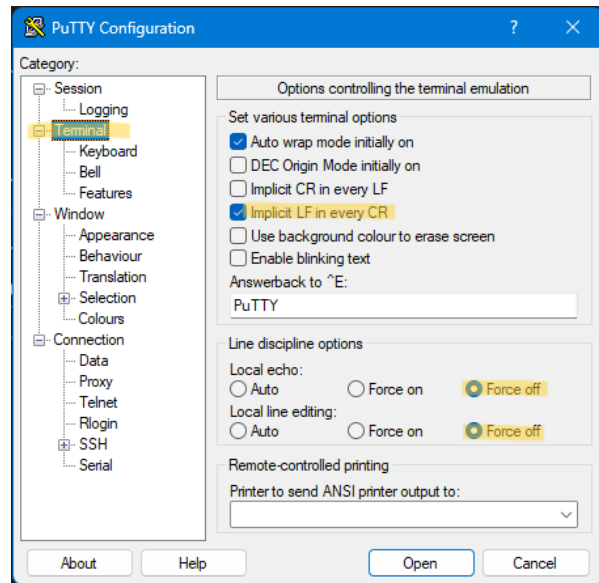
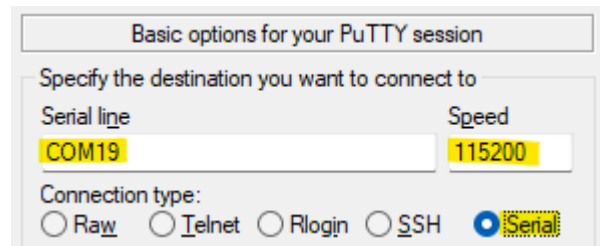
Serial Session

Select **Serial** then set the **Serial Line** to the COM port you found in the Device Manager and set **Speed** (Baud Rate) to 115200.

NOTE: Since this is a USB serial port the **Speed** (baud rate) doesn't matter. Data will be sent at USB speeds regardless of the **Speed** setting.

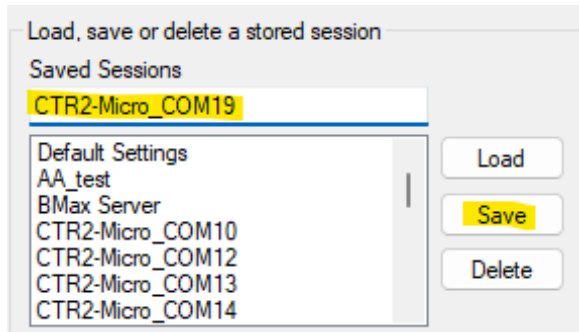
Next, select the **Terminal** item and set the **Implicit LF in Every CR** to on, and **Local Echo**, and **Local Line Editing** to **Force Off**.

You can change the window size under the **Window** item. Set the **Columns** to **41** and the **Rows** to **20**.



Once this has been done, return to the **Session** menu item, enter a name for this session and click the **Save** button. This allows you to easily re-open this session with just a couple of clicks.

If you right-click on the Putty icon in the Windows toolbar the last few sessions you had open will be displayed. Just select the one you want to open it.



You can adjust the display colors on the **Windows->Colours** menu item. The Micro uses the **Bold** attribute to highlight the *hotkeys* and other items. I like to set the **Background** color to blue and the **Bold** color to yellow but you can find the colors that work for you. After you get a color combination you like return to the **Session** menu and **Save** the session.

Appendix E: Change Log

v1.00.05 – April 7, 2024

- Added a reference to the [Conji](#) app for MacOS to help with connecting the *Midi* to a Mac computer.

v1.00.05 – March 26, 2024

- Revised button read code to reduce false triggers
- Added [Show/Hide MFB Counts](#) option in terminal set up
- Added *No Sidetone* solution to [Troubleshooting](#) table

v1.00.04 – March 23, 2024

- [Green LED now flashes](#) to indicate unit is powered up (CTR2-MIDI hardware only)

v1.00.03 – March 21, 2024

- Added additional information on importing the [.map file](#)
- Added additional information on [connecting CTR2-MIDI to the app](#)
- Added additional [troubleshooting](#) help

v1.00.03 – March 10, 2024

- Add beep types to tables
- Save encoder values to initialization file

v1.00.02 – March 8, 2024

- Added [Troubleshooting](#) section

v1.00.02 – March 7, 2024

- Shortened the MIDI device name from CTR2-MIDI_XXXX to CTR2_XXXX
- Removed **Paddle PTT** mode and replaced it with **Paddle Mode**
 - When the green LED is off the paddle inputs control MIDI Buttons 20 and 21
 - When the green LED is on the paddle inputs control MIDI Buttons 30 and 31
- Removed PTT latching logic that is associated with **Paddle PTT** mode
- The red LED now lights when either paddle is pressed
- Redesigned terminal screen

v1.00.01 – March 6, 2024

- Added an option in the [terminal display](#) to enable/disable **Paddle PTT** mode
- Added MFB ADC values to the terminal display

v1.00.00m – March 3, 2024

- Beta release for interface development

v1.00.00 – December 22, 2023

- Initial release of CTR2-MIDI firmware for CTR2-Micro