# G2 Mk2 Front Panel Implementation Notes

This documentation is for the G2 mk1 front panel adapter. This interfaces to the existing design front panel via its 40 pin connector and presents an interface to the Raspberry pi via USB serial. A Waveshare RP2040 -piZero Development Board is proposed which, if it can be mounted, may be used unmodified.

# Control Layout

A close-up of a white rectangle

Description automatically generated

The existing board is used, with existing controls. Controls should have identical functions to the current board both in Thetis and piHPSDR. There are no indicators on the panel, and no shift functionality.

Note that Thetis has code for “Reset G2 Panel Settings” with no distinction between V1 & V2; the settings are optimised for V2. P2app with a G2V1 panel will report numbers that mimic appropriate controls for a G2V2.

# Scan Codes

This section lists the scan codes from initial hardware reading of devices. A lookup table converts them to the reported codes in the next table.

## Encoder Numbers

The VFO encoder is treated differently from the dual encoders because its inputs are processed differently. From a software perspective, each dual encoder can have A (upper) and B (lower) encoders plus a “click” function. The “click” is treated as a pushbutton by the Arduino.

Normal dual shaft encoders have encoder numbers as follows:

|  |  |  |
| --- | --- | --- |
| **Encoder** | **Software Scan Code** | |
| **Upper knob** | **Lower knob** |
| 1 (top left) | 0 | 1 |
| 2 (bottom left) | 2 | 3 |
| 3 (top right) | 4 | 5 |
| 4 (bottom right) | 6 | 7 |

(CAT messages identify the encoder numbers as scan code + 1, ie values of 1-8)

## Internal Pushbutton Numbers

Pushbuttons have two numbers: the software scan code from the matrix algorithm; and the number that is reported to the Raspberry pi. A lookup table from scan code gives the reporting number. The scan codes are only used by the Arduino software and have no user meaning.

|  |  |  |  |
| --- | --- | --- | --- |
| **Button Function** | **Code** | **Button Function** | **Code** |
| MOX | 1 | A>B | 11 |
| 2 Tone/Tune | 2 | B>A | 12 |
| CTUNE | 3 | Split | 13 |
| LOCK | 4 | FCN | 14 |
| Mode + | 5 | RIT | 15 |
| Fil + | 6 | XIT | 16 |
| Band + | 7 | Encoder 1 press | 17 |
| Mode - | 8 | Encoder 2 press | 18 |
| Fil - | 9 | Encoder 3 press | 19 |
| Band - | 10 | Encoder 4 press | 20 |

# Thetis/HPSDR Reported Scan Codes

The scan codes sent by ZZZP / ZZZE messages, and reported to the client SDR app without modification, are identical to those reported by p2app when used with a G2V1 panel. These are “nearest equivalent” control assignments using the Andromeda mapping. This means the user experience will be the same as for a G2V1 panel read by p2app. Pushbutton and encoder ZZZE/ZZZP message codes are shown in the following diagrams.

The user can use the Andromeda assignments in Thetis and if necessary reset to Andromeda defaults; a little modification from there will be needed.

A white rectangular frame with black lines

Description automatically generated

Figure : Pushbutton codes reported to client app

The encoder functions, when Thetis is set to defaults for Andromeda, are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| **Encoder** | **Top knob** | **Bottom knob** | **Pushbutton Function** |
| Top left | Multifunction | Drive | Multifunction encoder |
| Bottom left | AF gain | AGC | MUTE |
| Top right | Filter high cut | Filter low cut | Filter reset |
| Bottom right | RIT | XIT | RIT/XIT reset |

A white rectangular object with a black border

Description automatically generated

Figure : Encoder codes reported to client app

# Software Structure

A Waveshare RP2040-piZero Development Board will be used, with software substantially based on the G2V2 panel code. Most control interfacing is polled, with the exception of the VFO encoder which will be interrupt driven.

## Differences from G2V2 Arduino

Key differences are:

* use of USB serial rather than wired serial;
* different I/O;
* no indicators;
* no encoder and pushbutton shift functionality;
* fewer encoders;
* pushbuttons are direct connected to MCP23017, rather than being in a matrix;
* VFO encoder is a “metal can” with different I/O treatment (see #define)

## Concept for Operation

The Arduino will interface the I/O devices, consisting of optical VFO encoder, mechanical dual shaft encoders and pushbuttons. It will handle all debouncing and present completed event data to the Raspberry pi. Events will be transferred at a timestep of every few milliseconds (potentially every 10ms) to avoid signalling and data transfer becoming a drain on processor resources.

Events will consist of:

* VFO encoder steps (potentially up to 20 steps per 10ms timestep);
* Mechanical encoder steps;
* Pushbutton events.

The optical VFO encoder may generate 500 cycles per revolution and can be turned at 4 revolutions per second. Several encoder steps per 10ms timestep are possible. VFO encoders generate clean edges, and interrupt driven operation is viable. The “metal can” encoder should interrupt on one input only, and use the level of the other to determine direction. There is #define code for this.

Mechanical encoders generate noisy edges and will be polled, with a debounce period of around 5 - 10ms after detection of a transition. A high turn rate would be one revolution per second and typically 24 steps so typically a rate of 40ms per step. One step per event is sufficient. All encoders will be scanned in parallel, but it is unlikely that more than one will be turned at a time so a high event rate is unlikely.

Pushbutton encoders also generate noisy edges and will be polled. The pushbuttons will be individually debounced. A debounce period of ~10ms is required and the event rate will be a max of say two presses per second. Both “normal” and “long” presses will be decoded.

## Use of Timeslots

Like all of my Arduino projects, the code will work to a fixed timer driven timestep. It will not follow the normal Arduino practice of executing a loop continuously; it will only run a loop once per timestep. That allows real time operation with known event timings.

Follow the principle from the Andromeda front panel controller: 2ms timestep is the starting point.

In one timeslot we need to:

* Update pushbuttons every 5th tick (I2C read for 16 buttons, digitalRead() for the encoder buttons)
* Update the encoders;
* Send any messages to CAT serial;
* Process any incoming CAT commands.

## I2C ports

I2C connects to one MCP23017 device. There is no interaction or dependency with the Raspberry Pi interface.

I2C needs to support the following operations:

1. 8 bit register write, for configuration
2. 16 bit read from adjacent registers

# Software Implementation

## Processor Issues

RP2040-piZero has a dual core ARM cortex processor. I will need to change some libraries

* Timer: use the RPI\_PICO\_TimerInterrupt library
* I2C: use the wire library, but I have observed some differences in behaviour between “standard” Arduino and the RP2040. Before the normal Wire1.beginTransmission() I need to have 2 additional lines:

Wire1.beginTransmission(0x20);

Wire1.endTransmission();

Then the normal

Wire1.beginTransmission(0x20);

There is no on-board LED that can be “blinked” for debug. Consider wiring one to GP0 (40 pin connector, pin 27).

## Data Exchange with Raspberry Pi

Serial message exchanges needed for:

1. Read product ID, hardware ID, software version (ZZZS)
2. Read VFO encoder steps (ZZZU, ZZZD)
3. Read dual encoder events (ZZZE)
4. Read pushbutton events (ZZZP)

## CAT Messages

This unit uses the same CAT commands as used for Andromeda. That means the existing THETIS / piHPSDR CAT command set can be used.

|  |  |  |
| --- | --- | --- |
| **Control type** | **Command** | **Meaning** |
| VFO encoder | Up: ZZZUnn;  Down: ZZZDnn; | nn = number of clicks (0-99) |
| Other encoder | ZZZEnnm; | nn = encoder number:  0: unused  1-20: encoder 1-20, clockwise  51-70: encoder 1-20, anticlockwise  m=number of clicks (0-9) |
| Pushbutton (including encoder) | ZZZPnnm; | nn=button number (1-99, 0=n/a);  m=0: released; m=1: pressed; m=2: long pressed |
| Send s/w Version | ZZZS;  Response ZZZSppnnmmm; | pp=product id  1: Andromeda 2: Aries 3: Ganymede  4: G2 panel; 5: G2V2 panel; 6: G2V1 panel adapter  nn= hardware version  mmm= s/w version |

# Hardware Issues

## Waveshare Module

The waveshare module has a 40 pin connector that connects by ribbon cable to U60 on the G2V1 display. Pin 1 on the ribbon cable (marked with an arrow on the connector) should be by pin 1 on the display PCB and at the left end of the Waveshare module as shown below

A close-up of a blue circuit board

Description automatically generated

Use this USB!

Pin 1

## PCB 40 pin I/O Connector

The front panel PCB connects to the RP2040 using a standard Raspberry pi 40 pin connector. The pins are:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Conn Pin** | **RPi GPIO** | **RPi2040 GPIO** | **Function** | **Conn Pin** | **RPi GPIO** | **RPi2040 GPIO** | **Function** |
| 1 | 3.3V | 3.3V | 3.3V | 2 | 5V | 5V |  |
| 3 | GPIO2 | GP2 | I2C SDA | 4 | 5V | 5V |  |
| 5 | GPIO3 | GP3 | I2C SCK | 6 | GND | GND |  |
| 7 | GPIO4 | GP14 | Encoder 3 1A | 8 | GPIO14 | GP4 | Not used |
| 9 | GND | GND |  | 10 | GPIO15 | GP5 | MCP23017 interrupt |
| 11 | GPIO17 | GP17 | VFO encoder | 12 | GPIO18 | GP18 | VFO encoder |
| 13 | GPIO27 | GP27 | Encoder 3 pushbutton | 14 | GND | GND |  |
| 15 | GPIO22 | GP22 | Encoder 1 pushbutton | 16 | GPIO23 | GP23 | Encoder 5 pushbutton |
| 17 | 3.3V | 3.3V |  | 18 | GPIO24 | GP24 | Encoder 7 pushbutton |
| 19 | GPIO10 | GP11 | Encoder 5 2A | 20 | GND | GND |  |
| 21 | GPIO9 | GP12 | Encoder 3 2B | 22 | GPIO25 | GP25 | Encoder 7 1A |
| 23 | GPIO11 | GP10 | Encoder 5 2B | 24 | GPIO8 | GP8 | Encoder 7 1B |
| 25 | GND | GND |  | 25 | GPIO7 | GP7 | Encoder 3 2A |
| 27 | - | GP0 |  | 28 | - | GP1 |  |
| 29 | GPIO5 | GP15 | Encoder 1 2B | 30 | GND | GND |  |
| 31 | GPIO6 | GP6 | Encoder 1 2A | 32 | GPIO12 | GP9 | Encoder 7 2A |
| 33 | GPIO13 | GP13 | Encoder 7 2B | 34 | GND | GND |  |
| 35 | GPIO19 | GP19 | Encoder 5 1B | 36 | GPIO16 | GP16 | Encoder 5 1A |
| 37 | GPIO26 | GP26 | Encoder 1 1B | 38 | GPIO20 | GP20 | Encoder 1 1A |
| 39 | GND | GND |  | 40 | GPIO21 | GP21 | Encoder 3 1B |

The Front panel logic is 3.3V. The whole panel can be powered through USB.

## Device, Pin Allocations

The front panel I/O entities to be interfaced are:

|  |  |  |
| --- | --- | --- |
| **Interface** | **Devices** | **Inputs** |
| VFO encoder | 1 | 2 |
| Dual encoder | 4 | 16 |
| Switch/pushbutton | 16 (i2C) + 4 | 4 |

### MCP23017 Pins

The MCP23017 on the front panel board connects the 16 pushbuttons. Pullup resistors to 3V3 provided; the MCP23017 is 3.3V powered. No interrupt needed; p2app will poll the interface every 10ms or so.

|  |  |  |  |
| --- | --- | --- | --- |
| **MCP23017** | **Address** | **0x20** | **A2=0 A1=0 A0=0** |
| **Port A** | **Inputs** | **Port B** | **Inputs** |
| GPA7 | Mode- | GPB7 | XIT |
| GPA6 | Fil+ | GPB6 | RIT |
| GPA5 | Band+ | GPB5 | FCN |
| GPA4. | Mode+ | GPB4 | Split |
| GPA3 | Lock | GPB3 | B->A |
| GPA2 | CTUNE | GPB2 | A->B |
| GPA1 | 2Tone/Tune | GPB1 | Band- |
| GPA0 | MOX | GPB0 | Fil- |

## VFO Encoder

The VFO encoder is marginal of 5V power and won’t work on 3.3V. It needs +12v power which could be picked up from the switched 12v on J19.

## Power

The radio power on/off button on the front panel does not connect in any way to the Raspberry pi for “soft” power shutdown. If that function is needed it will need to be wired separately to the rear of the G2V1 front panel PCB.

The Waveshare RP2040-piZero has an LDO from 5V (if connected) but no switched mode power supply. Hence noise issues should not be relevant.

# Testing

## Testing in Arduino Alone

|  |  |
| --- | --- |
| Connect USB cable to PC. Open Arduino terminal window. “blink” LED blinks at 1Hz rate |  |
| VFO encoder |  |
| Dual encoders |  |
| Pushbuttons |  |

## Testing with Raspberry Pi

|  |  |
| --- | --- |
| Powers up with no bus conflicts |  |
| Serial Data Transfer |  |

## Testing in p2app

|  |  |
| --- | --- |
| P2app retrieves product ID, HW & SW version |  |
| LEDs set from CAT polled data |  |
| Dual shaft encoder events sent to Thetis |  |
| VFO encoder sent to Thetis |  |
| Pushbutton events sent to Thetis |  |

# p2app CAT Interface

If the panel is controlled by p2app, then it will need to exchange “CAT over TCPIP” messages with Thetis.

Thetis operates a timeout: if a connection does not exchange data for 30s, it times out and drops the TCP/IP connection. “Keep alive” message exchange works, and a 15s message request means connection is maintained indefinitely.

Encoders and pushbuttons can all push data to Thetis.

# Arduino Software Installation

This guide describes how to download, install and load the Arduino software . The guide assumes that you are using the Arduino Integrated Development Environment (IDE) running on a windows platform. For users with different operating systems, different folder locations will probably apply.

## Install the Arduino IDE

The Arduino IDE is downloaded from the Arduino web page. The download links are on this page:

<https://www.arduino.cc/en/software>

Download and install the IDE. When you run it for the first time, it will look something like:



This is showing you a new, blank program. Arduino programs are called “sketches”.

## Install Arduino-Pico Core on Arduino IDE

From the Waveshare wiki:

1. Open Arduino IDE, click the File on the left corner and choose "Preferences".  
   [A screenshot of a computer

   Description automatically generated](https://www.waveshare.com/wiki/File:RoArm-M1_Tutorial04.jpg)
2. Add the following link in "Additional boards manager URLs", then click OK.

https://github.com/earlephilhower/arduino-pico/releases/download/global/package\_rp2040\_index.json

Click on Tools -> Board -> Board Manager

Type **pico** in the search box

Search through the list for Raspberry Pi Pico/RP2040 by Earle F. Philhower, III then click **Install**

## Upload Demo At the First Time

1. Press and hold the BOOTSET button on the Pico board, connect the Pico to the USB port of the computer via the Micro USB cable, and release the button when the computer recognizes a removable hard drive (RPI-RP2).  
   [A green circuit board with many small chips

   Description automatically generated](https://www.waveshare.com/wiki/File:Pico_Get_Start.gif)
2. Download the demo from [#Resource](https://www.waveshare.com/wiki/RP2040-PiZero#Resource), open the D1-LED.ino under arduino\PWM\D1-LED path.
3. Click Tools -> Port, remember the existing COM, do not need to click this COM (different computers show different COM, remember the existing COM on your computer).  
   [A screenshot of a computer

   Description automatically generated](https://www.waveshare.com/wiki/File:UGV1_doenload02EN.png)
4. Connect the driver board to the computer with a USB cable, then click Tools -> Ports, select uf2 Board for the first connection, and after the upload is complete, connecting again will result in an additional COM port.  
   [A screenshot of a computer

   Description automatically generated](https://www.waveshare.com/wiki/File:UGV1_doenload03EN.png)
5. Click Tools -> Board -> Raspberry Pi Pico/RP2040 -> Raspberry Pi Pico.  
   [A screenshot of a computer

   Description automatically generated](https://www.waveshare.com/wiki/File:Pico_Get_Start02.png)

To download code: the pizero needs to be put into a “download mode” where it is visible as an additional USB disk drive. Press and hold BOOT; then press and release RUN; then release boot. Code can now be compiled and uploaded.

To restore to normal operation: press and release RUN.

## Download the Panel Software Repository

1. Visit the repository on github: https://github.com/laurencebarker/ SaturnG2V1\_Front\_Panel\_Adapter
2. Click “clone or download” then “download zip”
3. Store the zip file on your PC for example in the “downloads” folder
4. Open the zip file and extract to your PC; for example into a folder “SDR” in “documents”
5. There will be a folder called “SaturnG2V1\_Front\_Panel\_Adapter-master” in your “SDR” folder

There are several folders:

|  |  |
| --- | --- |
| documentation | The user guide and this installation guide |
| g2v1picoadapter | Folder for the Arduino sketch |
| hardware | h/w design schematics etc |

## Build the code

To open the appropriate software sketch (the filenames etc are listed in the tables above)

1. Run the Arduino IDE
2. Use the "File|Open..." menu command
3. Open the “g2v2panel” folder
4. Navigate to " g2v2panel.ino" and click "open"
5. you should now see the files listed in tabs above the editor window