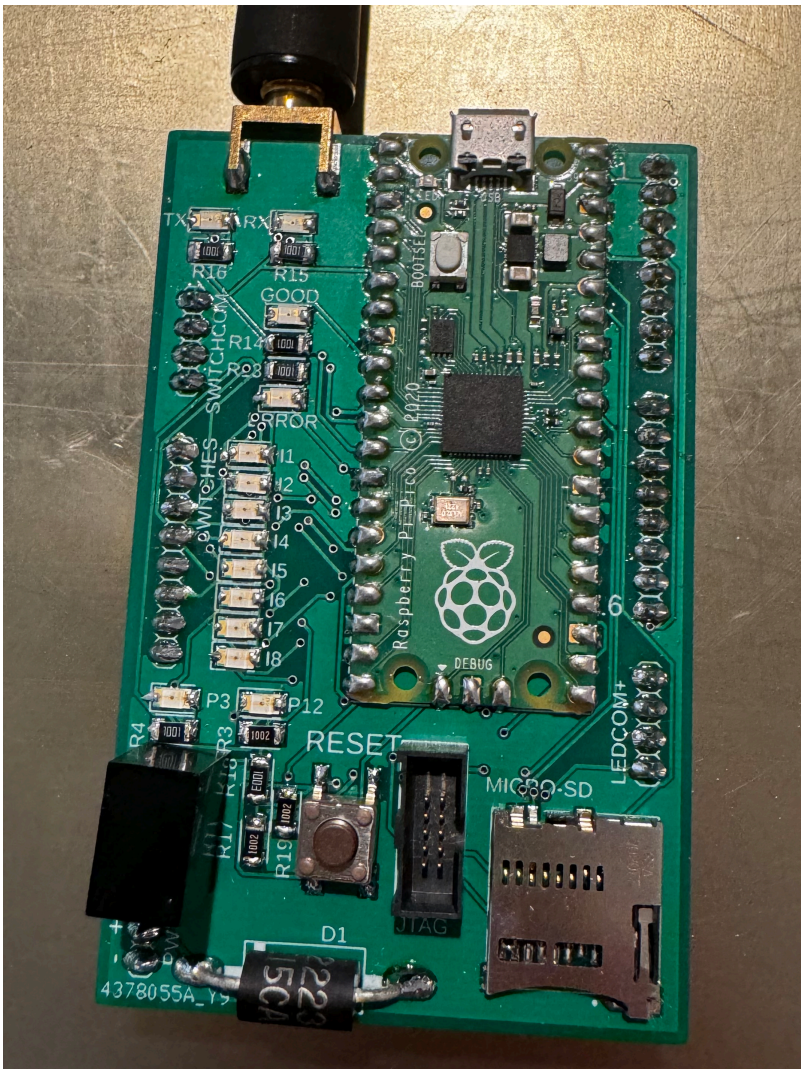
PicoSignal Signal Controller

User Manual



Author: Tristan Leavitt

Version: 2

For Software Version V2R3

May 31, 2024

Support: tpleavitt@me.com

1. Engineer's Responsibilities

Engineers are always responsible for the safe operation of their equipment. This signal system is not fool proof and can fail, so all engineers must ensure they are operating in a safe manner, regardless of the aspect displayed by the signals.

This signal system is intended to add a level of prototypical operation and simplifying traffic handling. It is not, nor ever will be, intended to be used as a safety device and it should not be treated as such.

2. System Overview

This system was designed with the intent of creating a flexible signal system for miniature railroads that was more feature rich and removed or reduced the issues and limitations from relay based systems. Part of the approach chosen was to use a wireless communication system to eliminate damaging surges in communication wires from lightning. This had the added benefit that this system is simpler to install or move compared to a wired system since there is no wiring to connect all parts of a block.

The wireless signal concept has evolved a lot since I first set out to create it, the original test was a simple Arduino Uno with a RFM69 radio module attached. From there it has expanded to it current form, using a Raspberry Pi Pico mounted on a custom made carrier PCB to support all of the circuitry needed for the system. The base card has eight protected digital inputs and 16 PWM capable current limited LED outputs. It has a 1.5kW surge suppressor on the voltage input as well as a reverse voltage protection diode to help prevent damage to the system. These components are currently rated to work with a 12V battery system but could be modified to work with up to 24VDC very easily.

The system is intended to be used with a button or whisker switch system for capture and release so no bond wires or rail connections are needed. It includes a timeout function to automatically release the block after a

adjustable amount of time to prevent back ups and cornfield meets that can occur when someone forgets to release a block. It also includes adjustable timers to dim and turn off the heads from inactivity, with the heads returning to full brightness either from a local button activation or radio activity detected from another controller. The configuration for each controller is stored in a human readable JSON file on a micro-SD card on the controller. No special software is needed to create or edit this config file, a simple text editor is all that is needed. Just put the micro-SD card in a standard PC card reader and open the file like you would any plain text file.

3. The Future...

The wireless signal concept has evolved and expanded a lot since its inception and development is always ongoing with new features being investigated all the time. Ideas for other features are always appreciated and I would gladly help work through the possibilities if you have any ideas, just send them to me at the email address listed on the title page.

4. Hardware Features Overview

1. Power

- Minimum input of 4.75VDC for the logic, LED voltage will vary by manufacturer and model. The input voltage needs to be higher than the minimum voltage for the LEDs to account for diode voltage drop.
- Maximum input voltage of 14.3VDC, suitable for lead acid batteries with solar chargers. This could be expanded if needed, feel free to reach out if you need a higher voltage for your application or are looking for recommendations on batteries and solar chargers.
- Diode reverse voltage protection up to 1000V.

- TVS diode surge protection up to 1.5kW, the breakdown voltage of this diode sets the maximum input voltage but must clamp below the 36V max for the voltage regulator.
- Logic circuits pull just 7mA in operation, total current will be the sum of all active LED currents plus the logic current.
- Low power sleep mode pulls just 5mA.

2. Communication

- Micro USB for programming and debug. Debug is a 115200 baud serial port.
- I2C to input and output circuitry. Allows for the possibility of expansion boards in the future.
- RFM95 LoRa radio communication to other controllers using the 915MHz ISM band so no FCC license required. The radios communicate at effectively 7000 baud so average round trip communication time is 70ms and provide reliable communication even in a heavily wooded mountainous environment with ranges up to 600ft.

3. Inputs

- 8 diode protected, active low inputs.
- Inputs are clamped to the input voltage.
- LED indication of input status.
- Input is active if resistance to ground is below 2500 ohm
- Input is inactive if resistance to ground is above 3200 ohm

4. Outputs

- 16 PWM capable, constant current outputs.
- Pins are tolerant up to 40VDC, but reverse voltage diode protection is recommended at the LEDs.
- Constant current drivers are programmable up to 58mA.
- 8 bit dimming control for fine control of LED brightness, including control of RGB LEDs for search light signals.

5. Control

- Raspberry Pi Pico micro controller running at 48MHz.
- Easily updatable with a computer.
- Micro-SD stored config file, easily edited on any computer.

5. Theory of Operation

On power up, the Pico starts executing the software, configuring its internal oscillators, initializing all communication buses and GPIO pins, and the holds for 5 seconds to allow a debugging serial monitor to connect. After the wait period, the system checks the input voltage and will halt boot up and turn on the Error LED if it is below 8VDC. Then the controller will attempt to mount the file system on the micro-SD card and load the config file. Boot will stop and the Error LED will be lit if this fails. Once the config file is loaded it will be parsed and the configuration will be loaded into memory for operation. If the address is not present or read as 0, or if any of the defined heads do not have any destination addresses set, boot will stop and the Error LED will be lit.
If the boot and config sequence is successful, the controller will cycle through the four colors available on all heads, Red, Amber, Green, and Lunar. If any colors are not configured, they will show as off. This allows to visually confirm the settings and ensure all LEDs are functional. The controller will then configure the RFM95 radio module with the parameters loaded from the configuration. It will set all heads to red and will attempt to establish communication with the destinations for each head with a release message. The head will go green if the communication is successful, if the head stays red after the transmission sequence double check that the other controller(s) are functional and the config file is set up correctly. With that, start up is complete and normal operation will begin.
Normal operation is the continuous loop of code responsible for running the signal controller. The first step in the loop is to check if the radio has received any messages and loaded them into RAM. If it has, the message will be

checked that its the correct size. If it is, all heads on this controller are set to Green if off, and brightness is returned to normal if it was dimmed. Then the message is checked to see if it is meant for this controller and a valid head on it, and is not from the same ID as this controller. If it passes the checks, the controller sets the appropriate head to the commanded color and responds if necessary. If the head was amber and the controller received a release command for it, the head will stay amber but be allowed to go red until an adjustable timer completes and the head goes to green. This process repeats until all messages have been processed. If the controller is still waiting for a response to a message, it will loop through this code checking for messages for up to the retry timeout parameter.

In the event multiple heads on the block are trying to capture at the same time, the controller with the highest ID wins.

After the radio is serviced, the stat light on the Pico itself will be toggled if it has been more than one second since the last time it was toggled. This provides a clear visual indication that the controller is running. A status light that is not blinking indicates there is a problem and the controller has locked up, even if the Error LED is not lit.

Next, the retry status of all of the heads is checked. If the controller did not receive the expected message during the last retry sequence, the controller will set up to try again. If the number of attempts has exceeded the allowed number from the config file, then no more attempts are made.

Next, the state of the input pins is pulled from the input chip for processing. For the input chip to register as active, the input must be less than 2500 ohm to ground. And to register as inactive, the input must be more than 3200 ohm to ground. Weather tight switches with gold plated contacts are recommended for outdoor service.

Each input is debounced depending on what type of input it is and what the previous debounced state was. Capture and release inputs must be stable for 5ms to register as active if they were previously inactive and they must be stable for 2.5 seconds to register as inactive if they were previously active.

This allows the voltage to stabilize in particularly noisy environments and prevents excessive intermittent shorts from impacting operation. Inputs from a turnout must be stable for 50ms to register in any state. Currently turnouts only support one input so point monitoring is not feasible. This can be expanded if needed.

If a capture input registers, all heads will be set to green if off and to full brightness if dimmed. If a turnout is associated with the capture input, the turnout input state will be checked to determine which of two heads the capture is for. The controller will then send a capture message to the first destination in the config for the head the capture pin is associated with. If the first destination captures successfully, the controller will display amber and continue capturing the remaining destinations for that head. If the first destination does not capture successfully, the controller will not continue with the remaining destinations.

If a release input registers, the controller will send a release message to all destinations for the associated head in sequence. If the associated head on this controller was amber, the amber timer will not activate and the head will immediately go green.

After all input signals have processed, the controller will check if any of the heads have been not green nor off for more than the amount of time allowed in the config file. If any head has, the controller will start the release sequence for that head.

Next the controller will check a timer to see if the incoming voltage should be checked. The voltage is checked once per second. If the voltage is below the low voltage threshold in the config file, the controller will blink all heads, toggling every 4 seconds, until the voltage exceeds the reset threshold in the config file. If the battery voltage is below the reset threshold, the LEDs will run at the dim brightness but not blink.

Next the controller will check the time since the last activity either local or on the radio network. If it exceeds the dimming time set in the config, the controller will dim all of the heads. If the time exceeds the sleep time in the

config, the controller will turn off all heads and enter low power mode. Both modes will be reset by any activity on the switches or the radio.

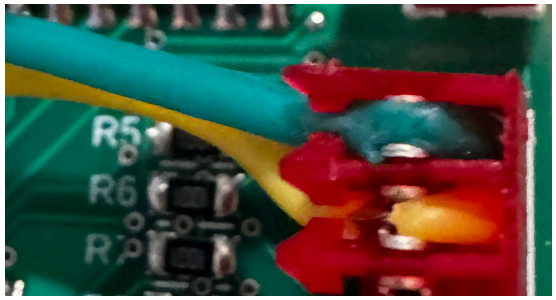
The radio module runs separately from the main controller code. It is interrupt driven in its action so it only runs when the main code loads in a message to be transmitted or the radio driver has received a message from another radio. For transmission, the message to be sent is loaded into the controllers RAM and the TX LED is lit. Then the radio finishes any reception before transitioning into channel activity detection mode. If activity is detected, the radio goes back to receive mode to receive the message and then waits a random amount of time between 1 and 100 microseconds before running channel activity detection again. Once there is no channel activity detected, or 500ms has passed, the message to transmit is loaded into the radios buffer and the radio is set to transmit the message. Then the radio returns to receive mode.
In receive mode, the radio is always listening for messages on the channel. When one is detected, the message is loaded into the radio's buffer as it is received. Once done, the radio checks that the message is valid. If the message is valid, then the destination is checked that it is either this controllers address or it is the radio broadcast address. At this time all messages from the controllers are transmitted on the broadcast address. If the message passes both checks, it is loaded into a circular buffer in RAM that can hold up to four messages where it will be held until the main controller code reaches the message processing section.

6. Installation

1. Connecting the controller

The power, inputs, input common, LEDs, and LED common all connect with MTA-100 insulation displacement type connectors. The connectors

supplied are for 22AWG wire only, a large wire can damage the connector and a smaller wire may not make contact. These connectors do not require the wire be stripped for installation and are more vibration and thermal cycle resistant than screw terminals. To install wires in the connectors, it is recommended to use a MTA-100 install tool. I can supply this or they can be ordered from many sources online. In a pinch, a small flat blade screw driver will work but does require care to prevent injury or damage. To install a wire, place the wire over the location in the connect you wish to install it with the connector on a steady surface. Then use the tool to firmly seat the wire in the connector. Checking for continuity with a multimeter is a good idea. See attached image for a properly wired connector. I can also supply connectors with wire preinstalled on request. Battery polarity is marked on the PCB.



It is recommended all wire leaving the controller's enclosure be direct burial rated to prevent current leakage. This could be landscape lighting wire, direct burial CAT-5/6 cable, or something similar with the required number of conductors for your application. The last 6in on either end is recommended to be a flexible wire such as stranded copper with silicone insulation for ease of movement. Its recommended that these wires be soldered and heat shrunk to the direct burial wire but crimp connectors are acceptable.

The antenna uses a standard SMA connector. The controller is provided with a dipole "rubber ducky" style antenna. Extension cables can be ordered online and added as needed, but it is recommended to keep the

cable as short as possible and not exceed 18in. It is best to mount the controller in such a way it does not need an extension for the antenna. Antenna direction does not matter as long as it is near vertical. The test railroad in Georgia has several controllers mounted in PVC boxes with the antennas protruding down through the bottom of the box with no issues.

2. Connecting Switches

Capture and release switches should be normally open momentary switches. Any normally open switch or button can be used, but it is recommended to use those that are weather tight and have gold plated contacts for the best service life.

Turnouts can be connected using a limit switch on the throw mechanism or can be attached to a polarity reversing selector switch for electric turnout machines. The reverse polarity will not damage the inputs.

3. Connecting LEDs

The LED common connector supplies reverse polarity protected voltage to the LEDs and then they are grounded through the LED driver. LEDs need to be wired with this polarity in mind as reversing this will not work.

4. Configuring the controller

The config file is loaded on a Micro-SD card on the controller. The config file itself is a plain text JSON file and can be edited with any standard text editor such as Windows Notepad. Care should be taken when editing the file as typos will cause the controller to not recognize the parameter. The syntax and acceptable values for all parameters in the config file are listed in Section 8.

7. Updating Software

The Raspberry Pi Pico bootloader makes software updates very easy. Updates will be provided in the form of a picoSignalsVxRv.uf2 file indicating the version and revision of software about to be loaded into the controller.
To update the software, connect the controller to a computer via the micro USB port. Then press and hold the small white boot select button on the Pico board while momentarily pressing the brown reset button on the main PCB. The Pico will then enter update mode and should appear as an external storage device, like a thumb drive, on your computer. Installing the update is as simple as dragging and dropping or copying and pasting the supplied .uf2 file to the Pico drive on your computer. The update will take a few seconds and then the controller will boot as normal.
If an update fails or one particular controller will not take an update, please reach out to me.

8. Configuration Parameters

The configuration is a JSON file stored in the root directory of the Micro-SD card. The file name needs to be “config*.json” where * can be any number. I recommend keeping a copy of the config files for all controllers somewhere in case a SD breaks or gets lost.
The parameter names are also case sensitive and should be entered exactly as shown. Parameters with “N/A” default values are required for operation. Parameters with defaults are optional and don’t need to be in the config file if the default is acceptable.

1. General Parameters

These parameters are general parameters that apply to the controller as a whole.

Name	Description	Unit	Min Value	Max Value	Default Value
address	The unique address of this controller	Units	1	255	N/A
retries	Number of retries to send a message	Units	1	255	10
retryTime	How long to wait between retries	Milliseconds	1	255	100
dimTime	How long of no activity before dimming the heads	Minutes	1	255	15
sleepTime	How long of no activity before putting the controller in low power mode	Minutes	1	255	30
lowBattery	Threshold to trigger low battery mode	Volts	0.1	36.0	11.0
batteryReset	Threshold to recover from low battery mode	Volts	0.1	36.0	12.0
ctcPresent	Indicates if a CTC map is present to send updates to	Boolean	FALSE	TRUE	FALSE
monitorLEDs	Monitors LEDs for faults and reports them. 0 is off, 1 monitors only for open circuits, 2 monitors for both open and short circuits. Only use 2 with AC to DC power supplies, the voltage range of batteries can cause false triggers.	Units	0	2	0

2. Head Parameters

These parameters are specific to each signal head. There are two different types currently supported, discrete and RGB. The colors for discrete heads are all optional but RGB heads require red, green, and blue be present. Heads are addressed in the file as “head1”, “head2”, etc.

General Head Parameters

Name	Description	Unit	Min Value	Max Value	Default Value
destination {x,…}	The destinations for the head, there can be up to 6 comma separated as shown	Unsigned Integer	1	255	N/A
dim	The 8 bit value for the dimmed brightness	Unsigned Integer	8	255	255
release	The time before the controller should release the block automatically	Minutes	1	255	6
Colors	See Below				

Discrete Colors - “red”, “amber”, “green”, and “lunar”

Name	Description	Unit	Min Value	Max Value	Default Value
pin	Pin number for the LED header, 0 indexed	Unsigned Integer	1	16	N/A
current	LED maximum current, values over 58 will default to 58	Milliamp	0	58	N/A
brightness	8 bit brightness value for the color at full brightness	Unsigned Integer	0	255	255

RGB LEDs - “red”, “green”, and “blue”

Name	Description	Unit	Min Value	Max Value	Default Value
pin	Pin number for the LED header, 0 indexed	Unsigned Integer	1	16	N/A
current	LED maximum current, values over 58 will default to 58	Milliamp	0	58	N/A

RGB Colors - “red”, “amber”, “green”, and “lunar”

“Color”: { “rgb” : {r, g, b}}

Name	Description	Unit	Min Value	Max Value	Default Value
R	8 bit brightness value for the red LED at full brightness for the specified color aspect	Unsigned Integer	0	255	N/A
G	8 bit brightness value for the green LED at full brightness for the specified color aspect	Unsigned Integer	0	255	N/A
B	8 bit brightness value for the blue LED at full brightness for the specified color aspect	Unsigned Integer	0	255	N/A

3. Input Parameters

These parameters define how each input pin operates, the pins in the config are 0 indexed. “pin1”, “pin2”, etc. All pins used require a “mode” config.

“mode” : “capture”

“head1” - The main head this capture is associated with, 1-4.

Optional parameters -

“head2” - The secondary head used if the turnout pin specified is active, 1-4.

“turnout” - The pin associated with the turnout, 1-9.

“mode” : “release”

“head” - The head associated with this release, 1-4.

“mode” : “turnout” - no other parameters

9. Sample Config File:


```
{
  "address": 3,
  "head1": {
    "destination": [
      4,
      0,
      0,
      0,
      0,
      0
    ],
    "dim": 10,
    "green": {
      "pin": 1,
      "current": 58,
      "brightness": 127
    },
    "amber": {
      "pin": 2,
      "current": 58,
      "brightness": 127
    },
    "red": {
      "pin": 3,
      "current": 50,
      "brightness": 127
    }
  },
  "head2": {
```

```
"destination": [
   5,
   6,
   0,
   0,
   0,
   0
],
"dim": 10,
"blue": {
   "pin": 3,
   "current" : 30
},
"red": {
   "pin": 4,
   "current": 30
},
"green": {
   "pin": 5,
   "current": 30
},

"red": {
   "rgb": [
      255,
      0,
      0
   ]
},
"amber": {
   "rgb": [
```

```
        255,
        255,
        0
      ]
    },
    "green": {
      "rgb": [
        0,
        255,
        0
      ]
    },
    "lunar": {
      "rgb": [
        50,
        50,
        255
      ]
    }
  },
  "pin1": {
    "mode": "release",
    "head": 1
  },
  "pin2": {
    "mode": "capture",
    "head1": 1,
    "head2": 2,
    "turnout": 3
  },
  "pin3": {
```

```
      "mode": "release",
      "head": 2
    },
    "pin4": {
      "mode": "turnout"
    }
}
```

10. Troubleshooting

Unfortunately sometimes things just go wrong. If you have an issue not listed here, feel free to reach out!

The PicoSignal controller performs a power on self test, or POST, every time it restarts. POST error codes blink both the Good and Error LEDS. If any POST codes are blinking, replace the PicoSignal

Number of blinks	Error
1	Bad Input Chip
2	Bad Output Chip
3	Bad Radio Transceiver

The error light blinks to indicate various caught errors during runtime of the signals, the meaning of each blink is listed below.

Number of Blinks	Error	Causes	Possible Resolutions
1	Battery voltage too low	Battery is 1V below the low battery threshold set in the config file	Replace or charge the battery
2	SD Card failed to mount	Micro SD card is not present or not functional	Replace the micro SD card
3	Failed to process the config file	The config file is not present on the SD card or is corrupted	Replace the config file on the SD card
4	Invalid configuration	The signal address is invalid or the signal head has no destination signal	Correct or replace the config file on the SD card
5	Transmission Failure	Noisy environment or damaged radio transceiver	Extend retry time in config file, if problem persists then replace the PicoSignal card

Problem	Resolution
Controller wont power up	Double check power supply polarity and operation.
Controller Error light comes on upon reset	Ensure power supply is above 8V. Check that Micro-SD card is fully seated in the socket. Double check config file.
Controller doesn't wake up	Double check power supply. Make sure that the capture switch resistance goes high enough long enough to ensure the debounce resets
Controller wakes up but stays green	Make sure other signals in the block are working
Controller locks up, ie stat light stops blinking	Reset controller. If the problem persists feel free to reach out to me with a description of what is happening when the controller locks up
Controller is slow to capture	Make sure the antennas on all controllers in the block are fully tightened.