Programmable Load

A DIY programmable load, intended for power supply and battery testing

CPU Board

The thing that runs the show

Peripheral Allocation

- CAN0: Expansion
  - PA22 (TX), PA23 (RX)
- SERCOM0: I2C, front panel/rear IO (through mux)
  - IOSET1
  - PA8 (SDA, PAD0), PA9 (SCL, PAD1)
- SERCOM2: I²C, analog board
  - PA12 (SDA, PAD0), PA13 (SCL, PAD1)
- SERCOM3: SPI, analog board
  - IOSET1
  - PA16 (SCK, PAD1), PA17 (MOSI, PAD0), PA18 (MISO, PAD2)
  - Chip select: PA19 (/EN)
  - Chip index: PB16, PB17
- SERCOM4: SPI, front panel display
  - IOSET1
  - DIPO = 0x0
  - DOPO = 0x2
  - PB12 (MISO, PAD0), PB13 (SCK, PAD1), PB14 (/CS, PAD2), PB15 (MOSI, PAD3)
- SERCOM5: SPI, NOR flash (bonus data)
  - IOSET6
  - DIPO = 0x3
  - DOPO = 0x0
  - PB2 (MOSI, PAD0), PB3 (SCK, PAD1), PB0 (/CS, PAD2), PB1 (MISO, PAD3)
- TC3: Fan PWM
  - PA14: WO[0]
- TC5: Beeper
  - PB10: WO[0]
- EIC: External interrupt controller
  - PA15: /TRIGGER
    - EXTINT15
  - PA20: /ANALOG_IRQ
    - EXTINT4
  - PB08: ENCODER_B
    - EXTINT8
  - PB07: ENCODER_A
    - EXTINT7
  - PA10: /IO_I2C_IRQ
    - EXTINT10
- XOSC1: 12MHz oscillator
  - XIN (PB22), XOUT (PB23)

- Debug
  - SWCLK (PA30), SWDIO (PA31), SWO (PB30)

> If desired, the driver communication interface can use CAN0 instead. It uses the same IO pins as the I²C bus, and requires a CAN transceiver on the board.

DMA Channel Allocations

DMAC should operate with dynamic, round-robin priority arbitration within a DMA priority level. Priority levels listed are from 0 (highest) to 3 (lowest.)

- Ch0: NOR flash SPI Tx empty (SERCOM5)
  - Priority: 2
  - Operate in SPI 32 bit data mode
  - Burst transfers
- Ch1: NOR flash SPI Rx complete (SERCOM5)
  - Priority: 2
  - Operate in SPI 32 bit data mode
  - Burst transfers
- Ch2: Display SPI Tx empty (SERCOM4)
  - Priority: 1
  - Operate in SPI 32 bit data mode
  - Burst transfers

Clocking

Clock Inputs

All clocks on the system are derived from one of the following clock inputs (oscillators and internal generators:)

Crystals

- XOSC1: External 12MHz oscillator
  - Provides primary system clock reference
- XOSC32K: External 32.768kHz oscillator
  - Runs in standby for RTC

FLLs

- DFLL48M: 48MHz
  - Used for USB reference clk
  - Uses external 32kHz osc for reference

PLLs

- DPLL0: 120MHz
  - CPU core clock

Clock Sources

The above clock inputs are then synthesized into multiple clock sources, each used by a different set of peripherals:

- GCLK0: 120MHz
  - Sourced from DPLL0 / 1
  - General high speed clock
- GCLK1: 48MHz
  - Sourced from DFLL48M
  - Intended for USB use
- GCLK3: 32.768kHz
  - Sourced from XOSC32K

    - SERCOM slow clock
- GCLK4: 12MHz
    - Sourced from XOSC1
    - General low speed clock
- GCLK5: 32.768kHz
    - Sourced from ultra low power 32kHz osc

Clock Consumers

- CPU core: GCLK0
    - Clock division factor: /1
    - Low power clock: /4
    - Backup domain: /8
    - High-speed: /1

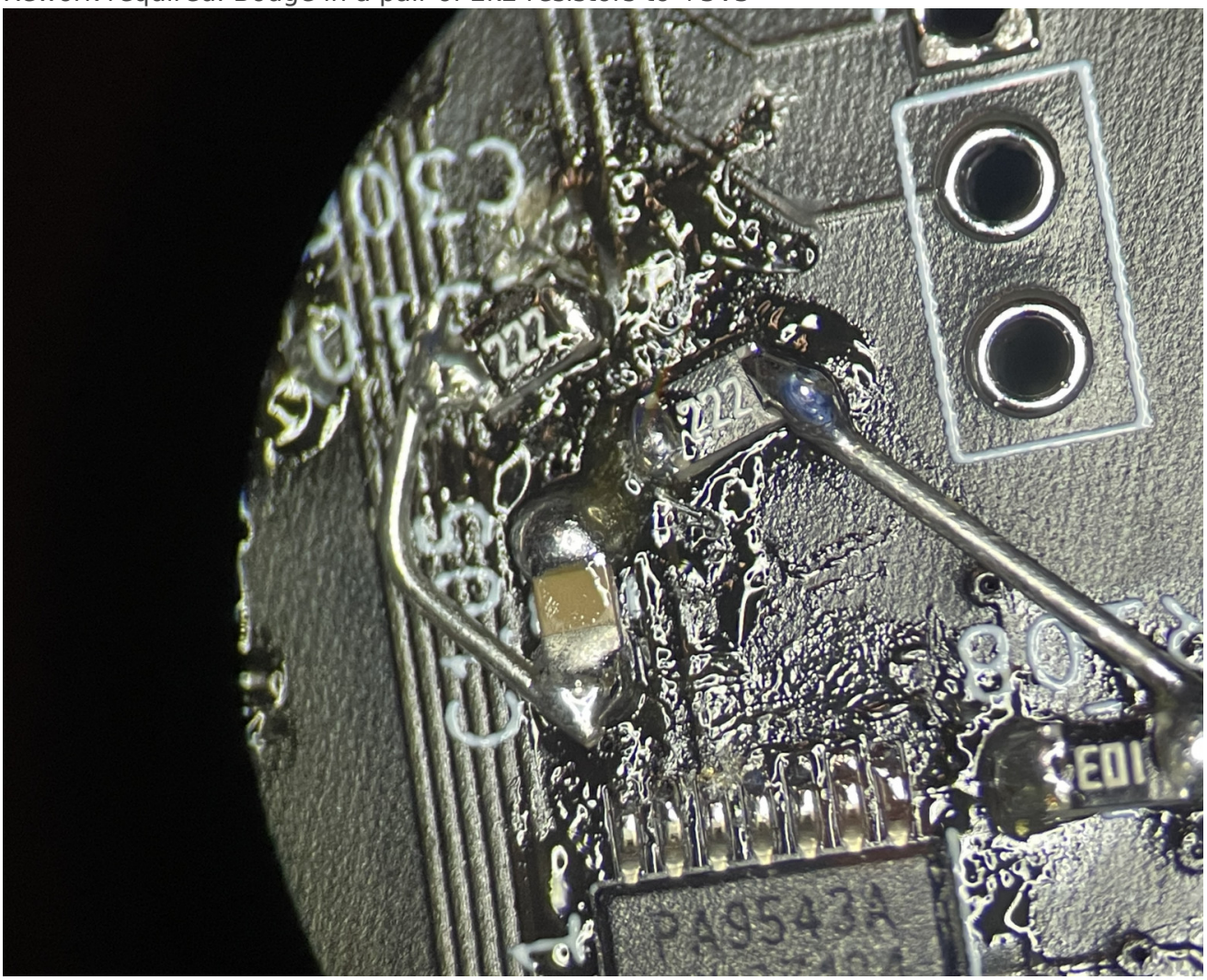
Hardware Errata

This page lists some issues with CPU board hardware, as they are discovered, and some workarounds.

Rev 1

- /I2C_IRQ's external IRQ line conflicts with ENCODER_A
  - Move /I2C_IRQ from PA7 to PA10
  - Rework required: solder line from pin 11 (/IRQ) of U101 to middle of J302 (TXD)
- Status LED (D302) footprint is wrong
  - The common (+) and red pins are swapped
- Footprint for MMBT3904 transistors (Q101, Q301) are slightly too small
  - The two pads side is slightly too much spacing between pads
  - They are still solderable, it just looks ugly
- NOR Flash (U303) is actually 4Mbit
  - This is the part I actually had lol
- Power LED driver doesn't work
  - The weird trick with the two resistors doesn't seem to work... like, at all
  - We'll have to revisit how this is controlled (external logic? sacrificing another pin?)
  - Footprint notes
    - Copper size on the pads could be increased
    - For mechanical retention of the switch, nudge the outer (switch) pads in by a small amount
- Switched front/rear I²C bus (from mux to MCU) is missing pull ups
  - Need to insert these between the mux (U101) and microcontroller

- Rework required: Bodge in a pair of 2k2 resistors to +3V3



IO

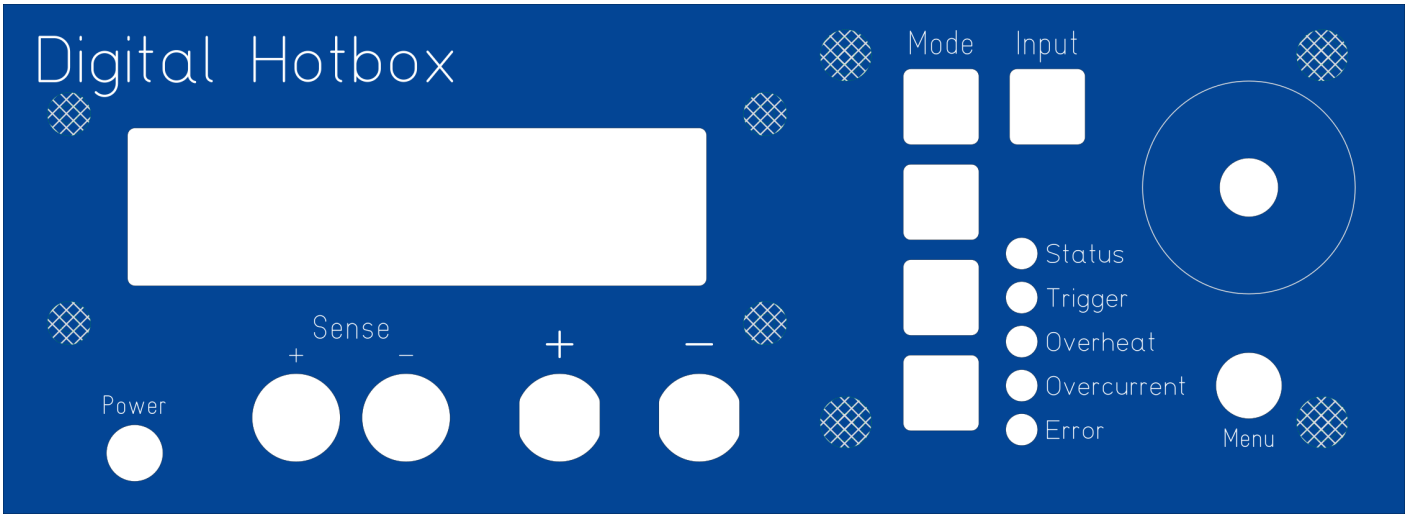
Front and rear panels

IO

Front Panel

The front panel features a few buttons, indicators, and a rotary encoder.

Overview



Note that the actual front panel board only covers the right third of the actual front panel: the area with the push buttons, indicators, and rotary encoder. It's mounted to the case via four 8mm standoffs with M2.5 thread, and should be screwed in with washers to distribute the force of button pushes more evenly across the board, rather than stressing the screw holes.

Connectors

There are two primary connections to the rest of the world on the front panel: the external voltage sense input, and the supply input.

> External voltage sense may or may not be used for measurement, depending on software control.

Supply input is made through a pair of binding posts (Cal Test CT4231) supporting up to 36A per post. The positive and negative terminals follow the standard red/black convention, and are internally connected to the driver board via thick wiring. The binding posts should use some sort of screw mount for the wire, which itself should have crimped ferrules or other terminals on the end

to attach conveniently to the wire.

External sense voltage is delivered through a pair of banana jacks (Cal Test CT2240.) These do not need to carry much current (if any) and are on a yellow and black connector (for positive and negative, respectively) there. These jacks are solder mount, which should be soldered to a (possibly shielded) twisted pair wire, which in turn connects to the driver board's external voltage sense input.

Buttons

The bulk of the buttons on the front panel consists of illuminated tactile switches (Omron B3W-9 type), which are further subdivided into the mode selectors (column of four switches, each with a single yellow LED) and the input on/off button (one switch, with dual red/green LEDs.)

Additionally, the button inside the rotary encoder (triggered when the knob is depressed) is available as a switch for selecting items in menu. It works in conjunction with the circular menu button (C&K D6R type) to browse menus.

On the front panel, all buttons are connected to a XRA1203 I²C IO expander. It features an interrupt output, which is connected to the controller board so buttons needn't be polled in software.

Lastly, there is a power button in the lower left corner; this switch actually exists on the controller board, rather than the front panel. It's an illuminated, right angle tactile switch (CTS 228A type) with the appropriate power icon (the little circle with the dash on it) printed on its cap. The board is set up to support bi-color illuminated switches so that there can be a standby/active type lighting situation.

Indicators

Indicators are LEDs, which are brought out to the front panel via light pipes. These indicators consist of one RGB LED (status,) one amber LED (trigger,) and three red LEDs (overheat, overcurrent, error.) Each LED can be individually controlled, and its brightness (current) adjusted.

Each of the buttons, with the exception of the rotary encoder itself, contains at least one LED. These LEDs are available to drive the same way indicators are, via the LED controller.

All LEDs are connected to a PCA9955B constant current LED driver.

Display

Any SPI display is compatible with the front panel, though it is specifically designed for the ER-OLEDM032-1Y module (or any other compatible modules with other colors; do note that the display may need to be modified for 4 wire SPI operation, by soldering some resistors and jumpers) and its pinout.

> Regardless of the type of display, it should not consume more than roughly 350mA of current. The main board has a 500mA polyfuse for the display.

The display mounts on 6mm long M2.5 studs, with nylon stand-offs to get the correct distance between the front panel and the display surface. Then, a washer and nut are attached to the end, to fasten the display securely.

Miscellaneous

On the front panel board, there is also an I2C EEPROM (AT24CS32 type) that contains the front panel's exact configuration, including the switches available, mapping of switch inputs and LED outputs, and LED drive characteristics such as maximum current.

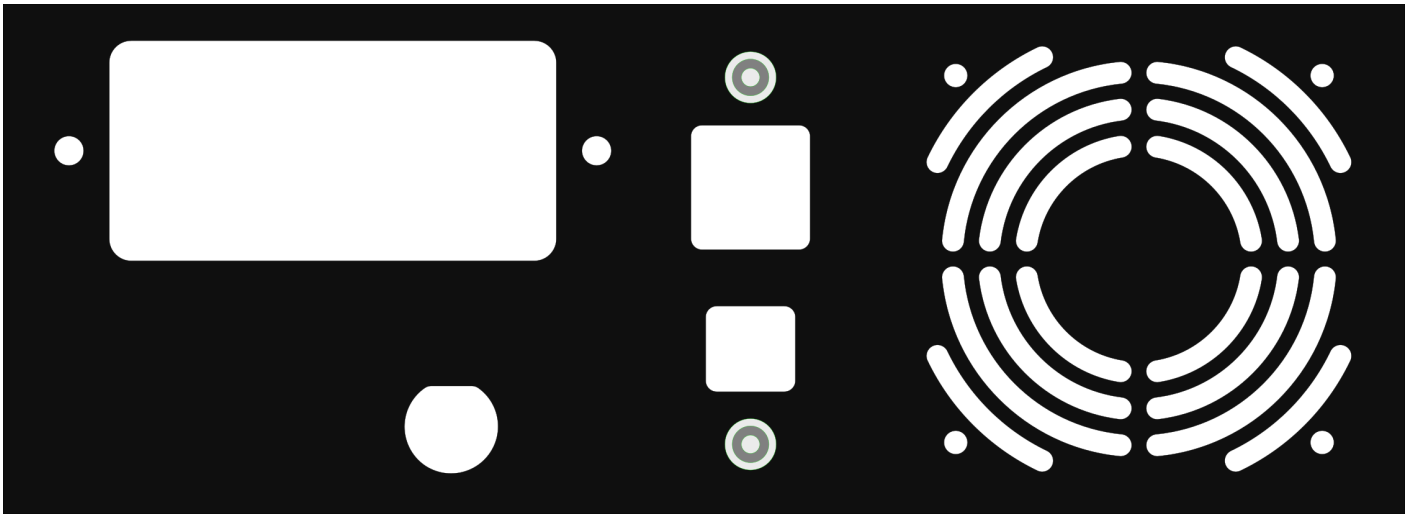
Revisions

Currently, there's just one revision of front panel, pictured above. It works, but it needs some minor rework:

- Menu button too small: needs +.4mm to diameter
- Edge machining on display cutout: A bevel would be more attractive

Rear Panel

On the rear of the device are a few auxiliary connections, including AC power input, communications (Ethernet, USB) and an external trigger input.



Rear panel, as viewed from front (inside)

Power

A cutout is provided for an IEC mains filter/input module with switch and fuse – in this case, the cutout is sized to fit the Astrodyne TDI 084 series, specifically the 084.00301.00 with a 3A fuse.

> It's very important that the IEC input module has a fuse built in; the only protection, if using the driver board's AC/DC module is a varistor to arrest surges.

Output from the filter (usually provided on spade terminals, or directly on wires) should then go via (sufficiently insulated) wires to a connector (CUI Devices TBP02P1W-381-03 or similar 3.81mm pluggable terminal block) to the driver board, which has a power module (CUI PSK-20D-12, Meanwell IRM-20-12, or similar) to produce the 12V from mains input.

Alternatively, this power module can be skipped in favor of an externally mounted power supply, in which case the mains input can be directly wired there; though take care to ensure that the case is still grounded. (The driver board has a 3.96mm JST-VH B2P-VH connector as a +12V input to accommodate external power supplies in place of a soldered power module, if that is desired instead.)

If an external supply is used, it should be capable of providing at least 1.6A at 12V with reasonably low noise.

IO Board

Ethernet and USB connectors are mounted on a small auxiliary circuit board, which mounts by means of screws on two 12mm M2.5 standoffs, and assumes a panel thickness of ~1.5mm. It serves as not much more than a simple breakout, converting from the 20 pin, 1.27mm ribbon cable coming from the controller board to these connectors.

On the USB connector, there exists basic transient filtering (via a TVS) and an I²C ADC to sense the VBus voltage. (There's no reason for a full blown ADC, but it was cheaper than an IO expander for just a single line.)

The Ethernet connector (Abracon ARJC07-111071A) is a vertical MagJack type, with the required magnetics integrated into the connector; therefore, the Ethernet side is not much more than a straight through connection, with some current limiting resistors for the connector's activity indicator LEDs.

Lastly, the board features an EEPROM (AT24CS32 type) for identification by software.

External Trigger

The last connector, near the bottom of the panel, is a cutout for a board mount BNC connector (TE 1-1634624-0)
on the driver board. This connector is used as an external trigger for various custom modes; it is relatively low impedance and drives directly an optoisolator.

No special treatment is required for the connector beyond the cutout, though it may feature a nut or other retention mechanism on the other side of the panel.

Miscellaneous

For cooling purposes, the remaining area on the right side of the panel has a cutout for a 60mm fan. This fan is automatically controlled by the processor board. It should be set up to suck air out of the chassis, to encourage more intake of fresh air at the front. This works in conjunction with the smaller fan on the driver board's heatsink.

Front Panel Errata

Rev 1

- Mode/load switches footprints need soldermask pulled back from pads
  - The pads are covered by soldermask. This is bad

Rear IO Errata

Rev 1

- Copper rings around USB connector pads should be larger
- Retention holes for Ethernet jack should be slightly smaller (to accommodate push-in expansion action for mechanical stability)

Load Driver

Separate board to actually do the load operations, with MOSFETs or whatever else

Hardware Errata

This page lists some issues with the hardware.

Rev 2

- Holes for current sense resistors should be slightly larger
  - Datasheet specifies 1.5mm ±0.12mm
- Increase spacing between heatsink and MOSFET/resistor slightly
  - Right now, the legs need to be bent at a bit of an angle, which makes fitting everything a huge pain in the ass
- Zero offset resistor (R307, ???) is too large
  - 4M7 is too large and doesn't let us trim out the entire DC offset (~4.5mV)
  - 1M was also too large (~3.5mV)
  - 200k works (able to trim to ~1µV remaining offset)
    - This is probably too low, maybe something like 330k or 500k is better
    - The trimming range is quite small

Rev 1

- MOSFET gate drive voltage too low
  - V_{Gs}, with the current configuration can only drive to max +3V3. This is not sufficient to turn on the MOSFETs selected (IXTH80N075L2) with a V_{Gs}(th) of 4.5V max
  - May be salvageable by rework (op-amp powered from 5V instead) and selecting a different MOSFET
  - Future work
    - Select a MOSFET driver opamp that can be powered from ±12V
    - Update power section to generate isolated 12V (replace PS201 with PDSE1-S12-S12-S)
    - Generate 5V locally (switching supply off 12V)
- I²C isolator (U203, ADuM1250) has the output SCL/SDA swapped
  - The I²C bus is swapped for all devices downstream of the EEPROM

Adjustments

The driver boards need to have done to work.

Rev 1

Current driver zero offset

Trimmers: RV301, RV302

This adjustment controls the zero offset of the current sense amps. Connect the load to a (current limited) power supply, with a current meter in line. Ensure the current DACs are outputting an all zeros code, then adjust each of the trimmers so that the load is not drawing any current.

External trigger pulse shaper

Trimmers: RV501

Changes the length of external trigger pulses. Connect a signal generator to the external trigger input, and an oscilloscope to TP501. Adjust until the pulse is approximately 50ms in length.

Voltage sense offset

Trimmers: RV601

Controls directly the input voltage offset differential amp, and is used to trim any residual offset in the voltage. Select external voltage sense input with the relay, then short the input sense terminals to each other. Adjust until the voltage ADC reads an all zeros code.

Remote Control

Information about the remote control interface of the load, including the underlying protocol, and differences between supported transports.

Protocol

This page describes the native communications protocol with the device. It is binary based, with a small header, and no other requirements on the data.

By convention, most (if not all) endpoints accept and provide data that is CBOR encoded.

Header

All packets have a simple four byte header that precedes the payload, regardless of the underlying transport:

- 1 byte: Message type
  - Indicates the "endpoint" inside the protocol handler that should receive this message
- 1 byte: Tag
  - Can be used to differentiate multiple outstanding requests and their replies
- 2 bytes: Length
  - Indicates the number of bytes of payload data that follow

Note that all multi-byte values in the header are sent in big endian (network) byte order.

Message Types

Below are all currently implemented message types/endpoints:

Number	Name	Description
`0x01`	Property Request	Get/set various properties on the device. See Properties for more info.

Transports

This same protocol can be carried over a variety of physical transports. The format of messages, including the packet header, are the same; however, the transport may add additional headers and padding, if needed. At this time, the following transports are supported:

USB

The first interface exposed by the device is a vendor specific interface, which consists of two bulk endpoints (one in and one out) used to transmit and receive the packets and their responses; as well as an interrupt endpoint, used to notify the host that the state of the device changed.

In this case, the device does not send unsolicited data. No additional headers are added to payloads sent over the endpoints, beyond the above packet headers.

Properties

This page describes the "property request" mechanism, as well as the message format used to interact with it.

Message Format

All messages on this endpoint are CBOR encoded.

Requests

Requests are maps, with one or more of the following keys:

- get: An array of property IDs to read
- set: A map containing properties to set. Keys in the map correspond to property IDs.

Replies

The device responds with a map, which contains the following keys. Which keys are included in the response depends on the request:

- get: A map (keyed by property IDs) containing the value of all requested properties. Unsupported/unknown properties are returned as `undefined`.
- set: An array containing the property IDs of all properties that were set. Any properties that were requested to be set, but are not supported (or read-only) will not be included.

Supported Properties

Below are all currently supported properties, including their IDs and value types:

ID	Name	R/W	Type	Description
`0x01`	HwSerial	R	string	Serial number of the hardware
`0x02`	HwVersion	R	string	Version information (such as revision) for the device

`0x03`	HwInventory	R	array	Information about all peripherals connected to the load. The array contains maps, which will have the following keys: • type: Peripheral type; may be one of ["load", "hmi" or "io"] • sn: Serial number (string; optional) • driver: Driver id (blob; optional)
`0x04`	SwVersion	R	string	Current software version (including build number)
`0x05`	MaxVoltage	R	int	Maximum allowable input voltage (mV)
`0x06`	MaxCurrent	R	int	Maximum allowable input current (mA)
`0x07`	DefaultVSense	RW	int	Voltage sense source on power-on: • -1 = state at last power off • 0 = internal • 1 = external This setting is persistent.

`0x08`	DefaultMode	RW	int	Operation mode on power-on: • -1 = mode at last power off • 0 = Constant current • 1 = Constant voltage • 2 = Constant wattage This setting is persistent.
`0x09`	DefaultCurrent	RW	int	Current limit value (for constant current mode) to apply on power on, in mA. -1 = last user specified value at power off
`0x0A`	DefaultVoltage	RW	int	Voltage limit value (for constant voltage mode) to apply at power on, in mV. -1 = last user specified value at power off
`0x0B`	DefaultWattage	RW	int	Wattage (for constant wattage mode) to apply at power on, in mW. -1 = last user specified value at power off

Controller (New)

A new and improved controller for the programmable load, based around an STM32MP1 dual Cortex A7/M4F.

Overview

This is a new controller board for the programmable load, designed around the STM32MP1 microprocessor, which contains a Cortex A7 core (suitable for running a full-blown operating system) as well as a Cortex M4 core (suited to running real time sensitive tasks with an RTOS) on the same package. To avoid needing to deal with BGA packages, high speed DDR routing, and a bunch of power rails, a MYIR MYC-YA15X SoM is used.

Connectivity

As with the previous designs of the load, it will feature an USB device mode connection, as well as standard 100Mbps Ethernet with a standard IPv4 and IPv6 stack. Additionally, a CAN expansion bus is provided to connect multiple units together, and possibly to other, larger loads down the road.

Internally, the controller provides both a high-speed SPI for the analog interface (to allow fast control of ADCs and DACs by software) as well as a low-speed I^2C interface, which is primarily intended for identification of the analog board, and some auxiliary control tasks like thermal management and input selection.

User Interface

Most of the user interface remains the same from the previous iteration, including the button and indicator layout on the front panel, barring some minor spacing changes.

However, the small greyscale OLED is replaced by a 800x480 4” capacitive touch LCD. This is driven directly by the LCD interface controller in the Cortex A7 side of the microcontroller, and provides an user interface that allows interaction by a hybrid means of the front panel buttons and touch controls.

Power

The switching AC/DC supply in earlier versions is replaced by a regular mains transformer, a standard rectifier and smoothing capacitors, followed by several DC/DC modules to generate the required voltages. Eliminating the AC/DC switching supply reduces the noise in the system, and provides greater isolation from mains.

Peripherals

Below are listed the general peripherals used by the controller. These are selected so that they consist of the peripherals that are available on pins broken out by the MYIR STM32MP1 SoM.

Cortex A7

These peripherals are reserved for the exclusive use of the Cortex A7 side. This runs a full OS, handles networking, expansion connectivity, the user interface, and so forth.

Communications

- I2C2: Front panel / local I²C bus
  - Through PCA9543A bus mux
- SPI3: Front panel display, CAN controller (through /CS-based mux)
  - Front panel display controller
  - On-board CAN controller
- UART4: Kernel TTY
- ETH: Ethernet MAC, in 100Mbps mode
  - PHY connected via RMII
  - STM32 generates 50MHz refclk for PHY
- USBH/USBOTG: Support for one USB host, one OTG device
  - Device exposes standard load interface
  - Support for boot over USB

GPIOs

- PB9: Front panel reset
- PC0: Ethernet PHY reset
- PC12: I2C2 mux reset
- PF6: Expansion connector power enable
- PF7: Expansion CAN termination
- PF9: Expansion CAN controller reset
- PZ6: Status LED (bicolor, 0/1/Z mode)

External IRQs

- PD2: I2C2 mux irq
- PA5: Ethernet PHY irq

- PC12: SPI3 irq

Cortex M4

These peripherals are reserved for the exclusive use of the Cortex M4 core, which runs the real-time control loop and a few other tasks better suited to an RTOS environment.

Communications

- I2C1: Analog interface (low speed)
  - Used for IDPROM, fan controller, temperature sensing, etc.
  - Runs at ~400kbps
- SPI5: Analog interface (high speed)
  - Runs at up to 20MHz
  - Up to 7 devices (3-bit select code)
- UART5: Debug console

Timers

- TIM2: Magnetic transducer (beeper)
  - PA3, Ch4 PWM output
- TIM4: Front LCD backlight
  - PB7, Ch2 PWM output

GPIOs

- PA12: SPI select bit 1
- PD8: Analog interface reset
- PD13: Status LED (green)
- PE11: SPI select bit 2
- PF8: Status LED (blue)
- PG5: Status LED (red)

External IRQs

- PG3: Analog interface
- PA4: Front panel encoder “A”
- PA5: Front panel encoder “B”
- PG8: External trigger
- PF10: Zero crossing detect

Hardware Errata

This page documents some issues with the hardware.

Rev 3

- Footprint for RTC backup battery (BT301) is backwards
  - The + terminal of the battery connects to the GND node, and vice versa.
  - Resolution: Populate battery in reverse
- RTC charger should be powered by system +3V3 rail
  - It's powered by the SoC's own 3V3 rail now, which toggles off during reboot
  - Use instead the system +3V3 rail
- Incorrect power-on reset behavior
  - ~~There is no reset pulse generated on boot-up (by an external reset generator... which we don't have. lol) which causes spurious boot failures~~
    - ~~So, we should add a supervisor for the reset line~~
    - ~~Generate a power on reset pulse of ~1 sec with the +5V rail~~
  - ~~We may need to shorten the reset delay on the +3V3 voltage supervisor line~~
    - ~~Alternatively – directly use the +3V3 input to switch it~~
  - This was caused by the RTC charger coming off the +3V3 SoM rail. Using the system +3V3 rail resolves this
- Pulse shaper stuff
  - Adjusting it sucks as TP501 is on the *analog* side of the filter, rather than the nice, pristine digital from U507
    - Fix: Add another test point at the output
    - Alternatively, we can probably work around this in software (with a special calibration UI?)
  - Possibly expand the adjustment headroom
- Front panel connection
  - The pinout of the connector is mirrored (left to right) with respect to the front panel when assembled
  - We can probably leave this (using a longer flex to compensate) or fix it on either the front panel or controller board revision (probably the controller board)
- Rear case fan (M501) connector sucks
  - The location is awful (it will be right underneath the analog board)
  - Relocate it elsewhere - but where?
    - Where VBUS LED is chilling
    - In the back by the expansion connector
      - We'd need to relocate some of the expansion IO stuff, like the termination and so forth

- Power supply section sucks to assemble
  - Is there anything we can do to make this less awful
  - Also, add an easier way to power the board off DC
    - Spring/screw terminal near power supply area