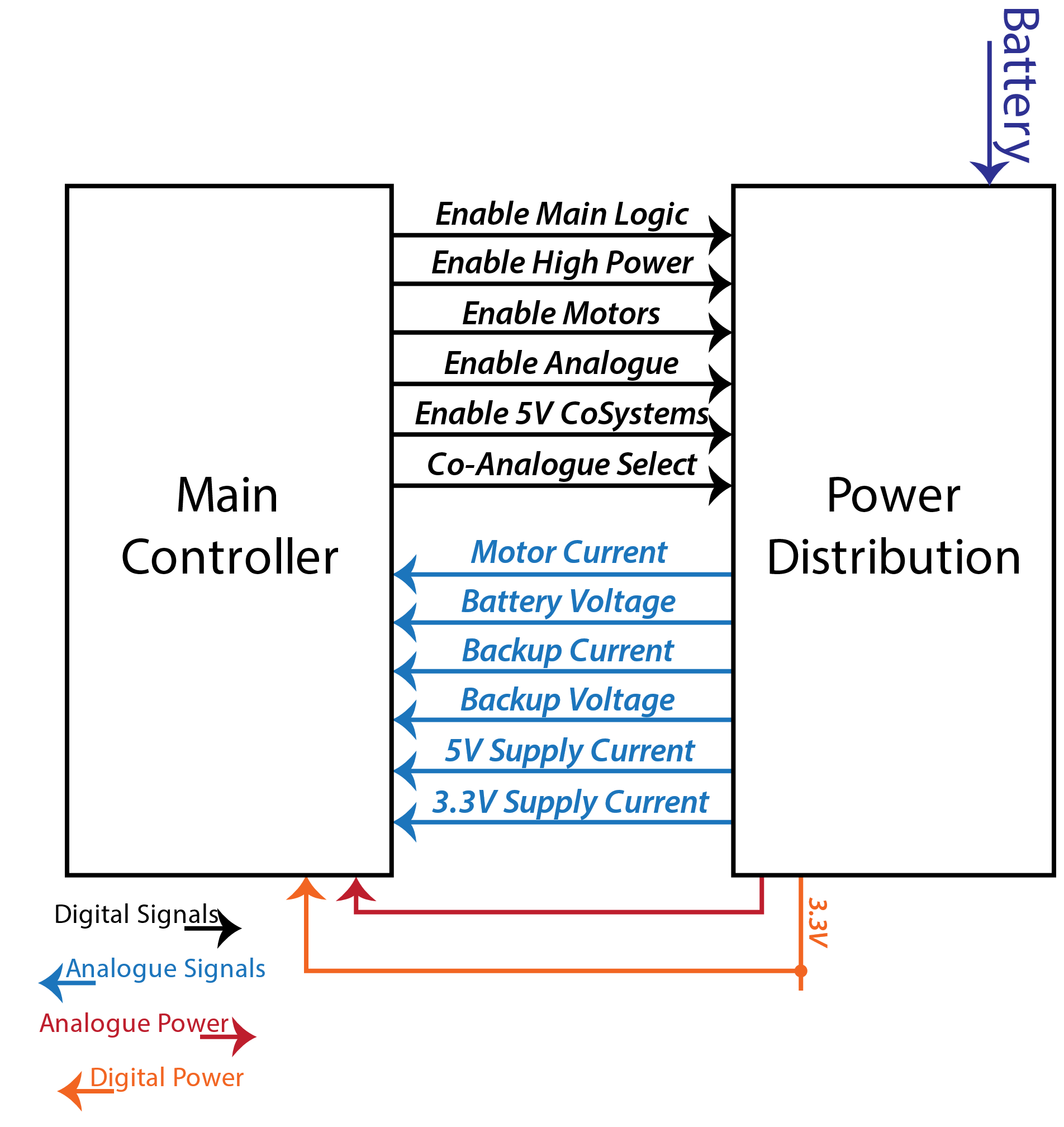
Off the Clock: Designing a Robot part 2

In my previous post, part one of this series, I ended on specifying the connections between the main controller and the power block. In this post I will finish that work, as it is the next logical step to verify if everything fits the processors I initially chose, or if things need to be moved around or even a processor needs to be replaced.

If you haven’t yet, I strongly suggest you read part one first, which lives here:

<https://medium.com/@asmyldof/off-the-clock-designing-a-robot-part-1-a3a37719c2c2>

To bridge the cliff-hanger neatly, I’ll recap the wiring description set at the end of the last post:

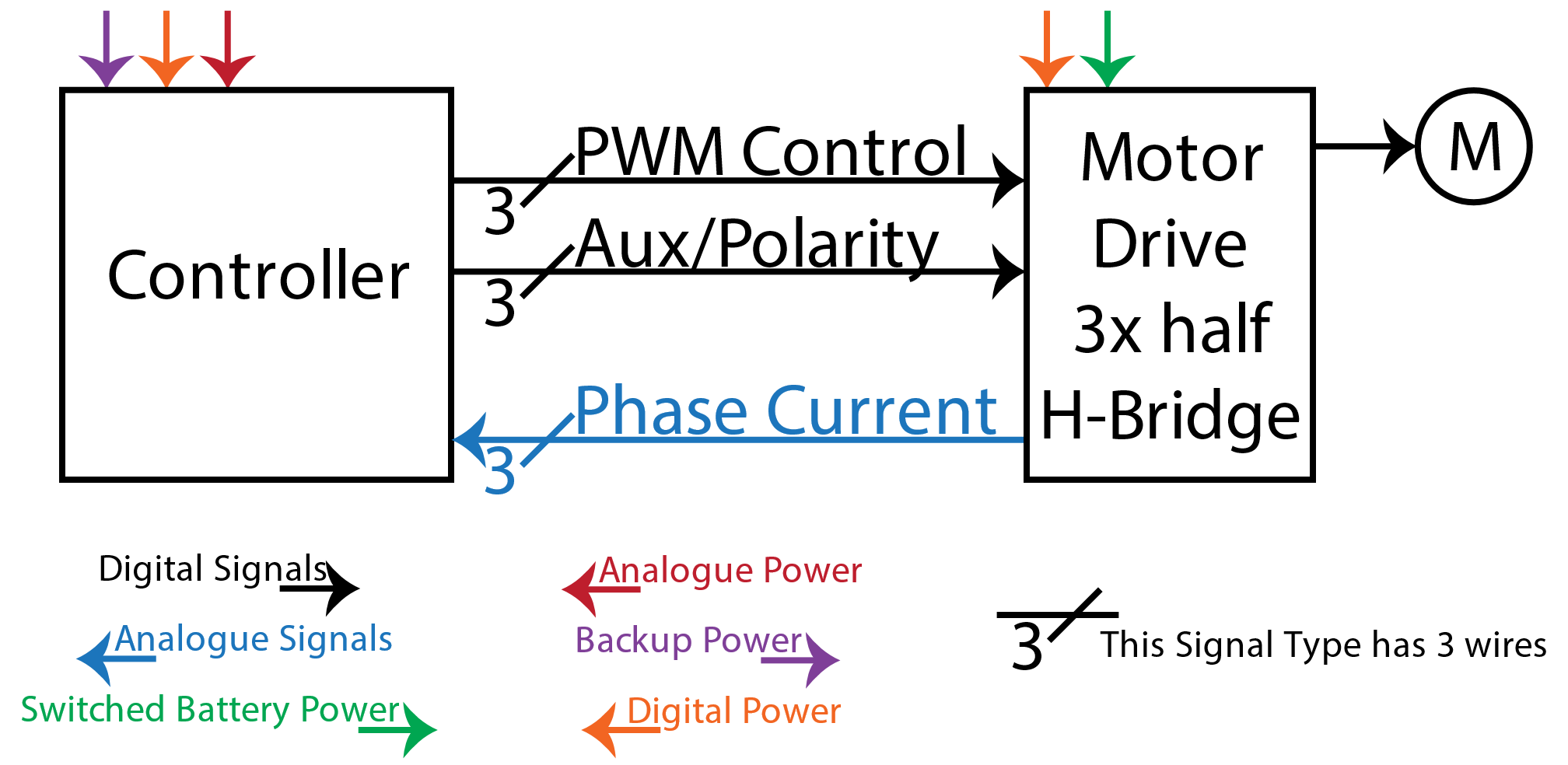


This image simply shows what kinds of control signals the Main Controller sends to the Power Distribution and what is measured back. Some of the analogue signals may be processed by hardware comparators or dedicated charge control chips at a later stage in the design, but for now we assume all that is done in Firmware, through the internal ADC. Since the backup voltage is also supplied to the expansion connection it is assumed for now we keep an eye on power consumption in that as well in some manner. Maybe the thing will trip a shut-off of the expansion at more than a few mA, or possibly there will be several alarm levels. This will also depend on the later choice of whether or not to add an actual backup battery or not. Right now, the backup power could also be a very-low-power linear regulator from the main battery, allowing the main processor to only “leach” a few dozen uA from it while sleeping. But I want expansions to also have the option to “stay ready” in a low power state when all main power is switched off, so the expansion will get a backup power wire, switched or not. The Co-Analogue Select is a provision I put in, in case I want to be able to select between two Analogue power and/or reference voltages for the Co-Processor.

## Controlling a Motor

To control a motor you need to decide what kind of motor you will be controlling. I choose to support three-phase and/or three-wire motors, which are normally controlled with on/off pulses to generate currents in the phases. Each successive set of currents puts the rotor at a known location, so by switching through given sets of currents at a specific speed, you can control the motor’s speed.

Instead of going for the normal approach of being able to actively control the high-side switch (the switch in the positive supply) and the low-side switch (the switch in the ground rail / negative supply), I’m doing it a bit differently. At least at first thought:



The idea is that the Motor Drive will be turned on and off at high frequencies – at least 10kHz, preferably more – while the polarity is switched by software at much lower speeds. This allows me to control the current more directly, hopefully allowing me to make sine-wave currents. There’s a half-H-bridge driver that seems to allow this behaviour, PWM’ing its “Output Disable” pin, while toggling the Polarity pin. But, since this is not normal use of the chip, I’d need to test this before committing to it. Luckily, I put a bunch of them on the Wurth Demo Board I mentioned in my first Off the Clock post:

<https://medium.com/@asmyldof/off-the-clock-3069af1cdd63>

Normally I’d test this by quickly manually etching a demo-set-up over two days or less and test it into destruction, but since I have a Demo Board with this set-up capable of driving up to 10A per channel (in theory), I can jump right to the testing right after this series of macro-design blogs. Don’t worry, I’ll write about that too.

## Positional Sensors

## Internal Bus

I was tempted to call this the Front-Side Bus (FSB) for a while, borrowing from main-stream computing, but I hate such blatant misnomers, since the FSB would be inside the main controller, along with the “North-Bridge” for the most part. The co-processor would be the South-Bridge for the most part. So this would be the un-named internal bus.

## Expansion Header