µBase Instructions

Chris Wendt, WIPAC (chris.wendt@icecube.wisc.edu)

Contents

Revision Log

Date	Version	Author	Comment
2019-09-04	v0.1	Ch. Wendt / T. Karg	First revision v0.1
2019-12-02	v0.2		Update for V2.3
2019-12-03	v0.3		Correct voltage ratio, K/Dy10=12.0
2021-01-11	v0.4		Describe quickscan
2021-01-12	v0.5		Update Ureportavg description

Connection to PC

The 10-conductor ribbon cable is used for communication and low voltage power. It is intended to be wired to an MDOM main board or equivalent that will supply it with 3.3V (for Cockcroft Walton power) and 1.8V (for microcontroller power), and will communicate with UART protocol at speeds up to 57600 baud. It is possible to use 3.3V for both the C-W power and microcontroller power.

The pinout of the ribbon cable is shown in the schematic (V2.3); here are some notes:

- BOOT_CONTROL (pin 10) should be held at ground, unless you are reprogramming the firmware. It can be set high (or left open) during power-up if the board should run the STM32 built-in bootloader code that allows reprogramming.
- VDDU (pin 2) is nominally 1.8V but could also be 3.3V. If not using the default 1.8V, you should tell the board what the actual value is after powering it up, because it is used as a reference value for the ADC that reads output voltage (use the command “Uset_vddu 3.3” for this).
- VDDSW (pin 4) is the Cockcroft-Walton power supply input. It is nominally 3.3V but is allowed to be up to 5.5V. Using 5V would allow very high output voltages to be reached, that could be useful for reliability testing but normally is not appropriate.
- UART TX (pin 6) is data flowing out of the base, so wire to RX on controller UART.
- UART RX (pin 8) is data flowing into the base, so wire to TX on controller UART. In order to reduce EMI from communication, the base has a multi-stage RC filter for outgoing data, and that could be used as a model for TX signal filtering on the controller board. See the schematic for details; it allows communication at 57600 baud while attenuating 1MHz very strongly.
- There is no connection for RTS or CTS signals from the controller.

It is possible to use a USB-serial converter like FTDI model TTL-232RG-VREG3V3-WE, in which case a ribbon cable could be made to connect with the wire ends of the converter: Black=Ground (pin 1 on MicroBase), Red=power output 3.3V (connect to both pin 2 and pin 4), Orange=TXD (connect to RX on MicroBase, pin 8), Yellow=RXD (connect to TX on MicroBase, pin 6). See comment above regarding use of 3.3V for both VDDU and VDDSW.

The anode signal output is coax MMCX type.

Software Installation on Microcontroller

The MicroBase board includes the STM32L432KC microcontroller, which includes a permanent bootloader program. The bootloader communicates with a host PC using the same UART connection that is later used for normal operational communications. The bootloader commands, including transfer of a firmware image for programming into the flash memory, follow a special protocol that is handled by the `STM32CubeProg` software running on the host PC.

The `STM32CubeProg` software can be obtained by searching for “STM32CubeProg” or “STM32CubeProgrammer” on the ST website, http://www.st.com, and following a download link. It can be installed in Windows, Linux or Mac environments using the instructions provided by ST. Although the software includes a GUI option, a single command line in a terminal emulation window is sufficient to program the firmware.

- Obtain the desired firmware image that is to be loaded into the flash memory. It should have the file extension “.bin”.
- Power up the MicroBase while the BOOT_CONTROL signal is held high or floating (there is a weak pull-up resistor on the board). The control signal is pin 10 on the ribbon cable connector. This will cause the built-in bootloader to run on the microcontroller, which will wait for commands from the PC. If the MicroBase is being powered from a USB adapter cable, this step is simply to plug the USB cable into the PC.
- Determine which serial port on the PC corresponds to the USB cable adapter (or equivalent). Its name will be needed on the command line, and on a Windows PC has a name starting with “COM”, referred to below as “COMn”.
- On a Windows PC, the following command line will transfer the new .bin file to the bootloader, which will program the flash memory and verify it:

```
STM32_Programmer_CLI.exe -c port=COMn br=57600 -w (bin file) 0x08000000 -v -g
```

- After programming, control will have been transferred to the MicroBase firmware, because of the “-g” option on the command line. It is acceptable to unpower the MicroBase at this point (e.g., by unplugging the USB adapter cable).
- To start a MicroBase without programming new firmware, either hold the BOOT_CONTROL signal at ground level during power up, or issue a command like this to the bootloader:

```
STM32_Programmer_CLI.exe -c port=COMn br=57600 -g
```

Operation

Command and response

All commands are sent on the UART as a single line of text, always starting with the uppercase character “U”, and ending with LF (or CR LF) character. After the initial “U”, the command is not case-sensitive. It doesn’t matter exactly what baud rate is used; the “U” character is used by the microcontroller to set the baud rate for the rest of the command line. UART settings are 8 bits, no parity, 1 stop bit. There should not be significant pauses between the characters of a single line, so either send each command using a script or use a terminal program on your laptop with a “line mode” feature. There is no editing capability for characters that have already been sent on the wire, i.e. you cannot use the backspace to fix typing mistakes. That is another reason to use “line mode” or a script. On a Mac laptop, I recommend “CoolTerm” because it has the “line mode” available (but you have to switch it

over from the default “raw mode” that just sends each character as it is typed). (CoolTerm is also available for other platforms but I haven’t tried it there.)

Every command will get a response, either just “OK” or some requested measured values or status information, depending on the command. The response will be at the same baud rate that was seen in the corresponding command line.

Operation frequency and output voltage

Because of the circuit design, any particular output voltage is best attained by one of a few choices of driving frequency. The resonance is determined by non-precision components and therefore the frequency has to be determined separately for each base. There is an automated and fast procedure (`Uquickscan` command) to pick an appropriate frequency, or it can be chosen by the user (`Upwmfreq` command) who then needs to be aware of the resonance curve (see `Uscan` command for how to get that). Having chosen the frequency, there is a software feedback loop which will adjust the “square” wave duty cycle (“PWM”) to maintain close to a requested voltage set point as measured at the first Cockcroft-Walton stage (between the anode and Dy10). The feedback loop is checking the output 10 times per second and uses proportional and integral error terms. If the frequency is too far off resonance, the output will be below the requested voltage.

The resonance frequency depends on temperature, but upward/downward changes of 10°C are not important.

The cathode output voltage is about 12 times the Dy10 voltage.

Uquickscan

The circuit performs best with a resonator driving frequency just below the resonance peak, which can be set quickly and routinely with the `Uquickscan` command. This command sets a desired output voltage and then scans the driving frequency upward until the output voltage is reached with good margin. The user should allow 15 seconds for the scan to complete and stabilize. Subsequently the frequency is left fixed and the voltage is automatically regulated to match the command value by adjusting the duty cycle upward or downward as needed. (In this context, "good margin" means that the available range of duty cycle values allows reaching voltage up to 10% above nominal.) The frequency range for scanning is preprogrammed (can be adjusted with the `Uquickscanrange` command if necessary).

Example (set Dy10 to 90 volts):

```
Uquickscan 90
```

(sets frequency to minimum of range, then scan upwards until Dy10 reaches 90 volts with good margin)

Uvoltage

If the base is already operating at a voltage v1, it's possible to use the `Uvoltage` command to change that to v2. However, the frequency must already be set to a good value because that will not be adjusted. Depending on how far off the frequency is from the "ideal" value, operation could be suboptimal in terms of power efficiency and regulation stability; generally the effect is unimportant if voltage is set between +5% and -20% compared to the natural output at the current frequency.

Example (set Dy10 voltage to 90 volts, then adjust up by 5 volts and then down):

```
Uquickscan 90
Uvoltage 95
Uvoltage 75
```

Uquickscanup

If changing to a higher voltage v2 which is more than 5% higher than previously set voltage v1, then the new target may not be reached without adjusting the frequency, and it's preferred to start a new scan. For this purpose, the `Uquickscanup` command works the same as `Uquickscan`, except that it will not waste time trying lower frequencies -- it starts at the current operation frequency and adjusts it higher if needed, typically needing only 5 seconds instead of 15 seconds to settle.

Example (sequence of setting Dy10 voltage to 90, 100, 110 volts, and finally 80 volts):

```
Uquickscan 90
Uquickscanup 100
Uquickscanup 110
Uquickscan 80
```

(for the last line, one could use "`Uvoltage 80`" instead, with a minor penalty on efficiency & regulation)

Upwmfreq

This is for manual setting of the operation frequency. If the automated procedure `Uquickscan` / `Uquickscanup` is not desired to obtain the preferred frequency, one can set frequency explicitly with the `Upwmfreq` command, which must be accompanied by a separate command `Uvoltage` to set the output voltage. The regulation (by adjusting PWM) will occur exactly as if the frequency were determined automatically. One foreseeable situation suggesting `Upwmfreq` would be if the “correct” operation frequency were already known for the base (e.g. kept in external database for all PMTs in operation), and then it would be faster to just set that frequency instead of going through the scan (using less than 5 seconds instead of 15 seconds).

The following example sets the driving frequency to 102,000 Hz and the Dy10 output to 110 volts:

```
Upwmfreq 102000
Uvoltage 110
```

Uget_uid

There is a unique identifier built into each microcontroller chip, which you can print with the command `Uget_uid`.

Ureportavg

To monitor the output, usually use `Ureportavg`, which reports a running average with a time response of about one second (so wait several seconds for complete settling). There are five lines of text returned; the first one is the first stage output voltage. The second line tells the current flow into DY10; the accuracy is limited to around 1uA but a significantly larger value (>2uA) may indicate excessive illumination. The third line tells what is the current consumption of the C-W driver, and does not include the microcontroller power. (The microcontroller consumption is around 1.8mA at either 1.8V or 3.3V supply.) The fourth line reports the C-W driver supply voltage VDDSW; note the accuracy is limited to about 0.1 volt and the ADC scale is set from the VDDU value (MCU power supply voltage) so erroneous values could indicate a problem on either VDDU or VDDSW. The final line “frac” gives the ratio between the Dy10 voltage and the value it would have at maximum duty cycle at the same frequency; values between 0.7 and 0.95 are typical.

The values from Ureportavg can be obtained individually by the commands `Uget_avg_v10`, `Uget_avg_di10`, `Uget_avg_isup`, `Uget_avg_vsup` and `Uget_avg_frac`.

Umonstatus

There is another monitoring command `Umonstatus` which reports what happened in the last feedback event (which occur 10 times per second). The voltage measurement in this mode appears more noisy, it doesn’t have the averaging like `Ureportavg`. The level of fluctuations is non-trivial especially far from resonance, but not large enough to spread the SPE peak noticeably. Some of the fluctuations are clearly just measurement noise, while some of it is due to frequency drift that is being compensated by the feedback algorithm.

The operation frequency is one of the reported items, and doesn’t need any averaging because it is generally fixed (by `Uquickscan` or `Upwmfreq`).

While `Umonstatus` reports multiple items separated by line breaks, the command `Uget_freq` just reports the frequency in Hz.

Example of obtaining the preferred operation frequency for a setting of 90 volts on Dy10 (first checking the actual voltage with `Ureportavg`):

```
Uquickscan 90
Ureportavg
Umonstatus
```

Uscan

For proving good performance of a particular board, it can be helpful to measure and report the full resonance curve. In that case, use the command `Uscan 100 95000 115000 -1` or some variation on that. Within the specified frequency range (95,000 Hz to 115,000 Hz in this case), the firmware will try every possible frequency (generally in steps of 250Hz) with a target output of 100 volts, and will record the performance at all those frequencies. Each value is allowed to settle for 10 seconds before moving on to the next one. While the "OK" response should be returned immediately, the scan will take quite a while (5 to 15 minutes depending on what range is chosen for the frequencies). You can determine if the scan is done using the `Uscanstatus` command and looking for the value of 0 or 1 (0 = scan is completed). After the scan is done, the command `Uscanprint` will list the results. It will show the maximum voltage attainable at each frequency even if that is higher than the requested target voltage, and without actually delivering more than the requested target voltage; it does that by a formula based on the PWM duty cycle that the feedback loop settles into. Actually if a PMT is attached, you might want to use a lower voltage like 80 or 90 volts, that when multiplied with a factor ~12.0 will not be too high. Also, you can speed up the process by only scanning a narrower range of frequencies, assuming you have previous information about the board and just want to look for some small changes. Note that after the scan is done, the set frequency and target voltage will be set to whatever they were before the scan. Look at the results and pick a value below the resonance peak. It is okay to pick any frequency that lists an output maximum that's greater than your desired set voltage. Remember, the actual output will not go above your set point. If you operate closer to the resonance, the output fluctuations are a bit lower, but the current consumption might be 10% higher.

Ustatus

The `Ustatus` command lists several operational mode bits. This is mostly for diagnostic purposes.

Usleepenable

By default, the microcontroller core sleeps during idle periods. It will wake up 10 times per second for the feedback loop, also if any commands are received on the UART. This is not a deep sleep, but reduces the average power consumption by about a factor of two. The behavior can be changed with the command `Usleepenable 0` (to disable sleeping) or `Usleepenable 1` (to enable sleeping).