Boston University ENG EC464: Second Deliverable Test Report

Team 14: Bikeputer (Re.Cycle)

Brian Kane

Katherine Murphy

Ben Paolillo

Jiwon Song

Todd Sukolsky

**1.0 Battery Switching System**

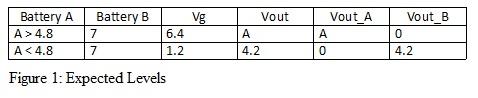
**Equipment and Setup:**

The switch circuit was set up on a breadboard, powered by an Agilent E3631A Triple Output DC Power Supply, which simulated battery levels. The batteries themselves were not used due to the fact that they cannot be drained/charged safely in the time given for demonstration, and that the purpose of this demonstration is to show switch functionality and control. An Agilent 34401A 6½ Digit Multimeter was be used to monitor the output voltages and and LED was used to indicate the status of the only switch to have no input voltage, so it was clear which “battery” was supplying and which is being charged at each given stage. Battery A is designated as the master battery, and a comparator is used to monitor this battery and control the gates. Battery A was simulated by adjusting the 0-6V supply, and Battery B was simulated by adjusting the -25- +25V supply.

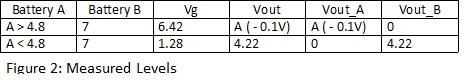
**Measurements Taken:**

The output voltage Vout was measured at each stage of the process and compared with expected/required values. An oscilloscope showed a smooth output voltage which varied only when expected, i.e. when the batteries were switched.

The expected outputs were as follows, calculated and measured before the test:



The measured outputs were as follows:



Vout indicates the switching circuit output to the main module; Vout\_A indicates the contribution of Battery A; Vout\_B indicates the contribution of Battery B; Vg indicates the voltage at the gates of the switches. All measurements are in volts.

**Assessment:**

The measured results match the expected outputs closely enough to be considered accurate since the differences are so small so as not to affect any change in which battery is in which position. All differences are less than 1% and may therefore be considered negligible. The results show that when Battery A reaches the chosen threshold of 4.8V, switches to charging position while Battery B is simultaneously switched to supplying position. By adjusting the power supply, it was shown that when Battery A is sufficiently recharged, the two batteries switch back to their original positions.

**Conclusion**

We have shown that the output voltage (the supply voltage to the main module) is acceptable and behaves as expected, and that the switches operate in a way that keeps the batteries from over and undercharging. The Switch Circuit Test was considered a success since the measured results matched the predicted/desired results. Based on this test we are therefore able to finalize the Switch Circuit design with the only necessary changes being the use of a lower power consuming comparator, and surface mount MOSFETs. The Switch Circuit is included in the Schematics section with the updated components.

**2.0 Beaglebone and AVR Communication**

**Equipment and Setup:**

The final implementation will see a different processor than the one used to test the graphics AVR (GAVR) communication. Instead of an ATmega644PA for the GAVR an ATmega2560 will be used. There are no peripheral disadvantages for testing with the ATmega644PA. The Watchdog AVR (WAVR) is an ATmega324PA and will be used in final implementation.

The Beaglebone is connected to the Graphics AVR (GAVR) over UART through the bone’s UART4 ports and the GAVR’s UART0 ports. The two also share a common ground to eliminate off-balance voltage levels. There was a also a GPIO signal connected between the bone’s P8, pin 4 port and the GAVR’s PCINT0 on PA0. This pin change interrupt alerted the GAVR that the beaglebone was about to send a command and should therefore wait for a character to be sent over.

A second serial line is bridged between the WAVR and the GAVR on both UART1 ports, respectively. The WAVR has a GPIO line connected to the GAVR’s INT2 port allowing the same warning that the bone sends to be acknowledged on the GAVR, however instead of listening on UART0 the GAVR listens on UART1.

The final serial connection is made between a host linux machine and an external GPS chip that streams NMEA strings. This final connection will be changed to move UART data directly from the Linx GPS system’s TX/RX ports to the bone’s UART2 ports. Becasue the BeagleBone’s kernel does not include drivers/support for USB serial connections the demonstration of the custom GPS streaming could not be done on that machine.

**Measurements and Commands Performed:**

The first test consisted of running a custom python script, “myGpsPipe”, to pipe the NMEA strings from the USB to a temporary file entitled “raw\_strings\_dev.txt”. By executing the script with the options “-d” a debug state was enabled showing the NMEA strings being streamed across the console. After running the script for 10 seconds the text file was populated with NMEA strings, invalid since no connection could be made to GPS satellites, but live strings nonetheless. The second test, which was integrated in the first test, was placing the time of GPGGA (location data) strings into a temporary file. In final implementation this occurrence will fork another process that sends the time string in the format “hh:mm:ss”. In this test, however, the string “NONE” is placed into a text file “initTime.txt” to indicate the finding of a location string with time stamp with an invalid time/lock. In the case of a no lock the script will send the string “NONE” which the WAVR does not respond to. The third test completed was a parsing of the NMEA strings for longitude, latitude and time. This was done by executing the python script “nmeaLocation” which wrote the respective data into another text file “locations.txt”. Since we had no connection to satellites test strings were used to show the parser worked correctly and could pull out longitude, latitude and time strings.

Moving away from the host linux machine the next test was a simple communication protocol between the BeagleBone and the GAVR. The BeagleBone communicates using python, specifically the pySerial library which opens serial ports using commands similar to python’s socket protocol. The test GAVR script is titled “testGAVR” (github.com/tsukolsky/beagle-bone.git) and can be used to send a single string with the “-s <string>” command or in debug mode with the “-d” command. We executed the script in debug mode sending the commands “time”, “date”, “both” and received the expected answer for each. When sending “NONE” the GAVR responded with NONE signalling the close of the connection.

The second serial connection, between the GAVR and WAVR was attempted, but not made correctly. The protocol was put in place one day before the test and was meant to solely run alongside the other protocols. At no time did the two chips attempt to send each other messages.

**Assessment:**

All tests in this section were successful. The python script “myGpsPipe” safely piped NMEA strings from an incoming serial connection to a temporary file as well as placed valid time strings, or if invalid “NONE”, into a separate file. Also, the python script “nmeaLoaction” successfully parsed live NMEA strings (even though blank) as well as valid test strings writing the locations to a file. These tests can be deemed successful.

The second partition of testings was also successful. The python script “testGAVR” successfully established a connection between the bone and GAVR, asked for certain internal values in the GAVR and received these strings. This communication protocol is groundwork for sending user trip data, usb data and other information between the two processors.

**3.0 Watchdog AVR Unit Functionality**

**Equipment and Setup:**

The WAVR is connected to the BeagleBone via the WAVR’s UART0 ports and the bone’s UART2 ports. The serial connection is warned using a GPIO port located on P8, pin 3 on the bone. There are also numerous LED’s attached to the WAVR used to debug and see the internal state of the WAVR. Two main led’s demonstrate whether the unit is asleep or on while another five represent power enable signals and another four represent power-down procedures. The WAVR is an ATmega324PA as previously stated. An addition to the circuit is an on-board temperature sensor, TI’s LM95071. This chip is connected to the WAVR via SPI, SS pulled high. The WAVR has two external oscillators, a 20MHz main system clock oscillator and a 32.768kHz real-time clock oscillator.

Power is provided to the module via two power sources: first, a 9V battery is being regulated down to 3.3V which is fed into a power mux, TI’s TPS2110A. The second input into this power mux is a 3.3V source from an external development board. The output of the power mux is distributed to the power rails for both the GAVR and WAVR.

**Measurements/Commands Performed:**

The first test of this section is the ability for the WAVR to keep a real time clock. To test this a time and date is programmed into the EEPROM at program time. On startup, the WAVR loads the data into a “clock”. To test the clock a program/script “testWAVR” is excuted in debug mode (“./testWAVR -d”) which sets up a communication line between the bone and WAVR. From there the WAVR is pinged for time, date, both, temperature and battery (ADC) readings. Next, to test the ability to set the current time given a valid GPS time string a sample time string was sent in the format “hh:mm:ss”. The communication line then closed and manually restarted (this is the correct protocol). From here the time on the chip was asked for again and, as expected, it matched the time that was sent to it. This protocol also, in turn, tested the temperature monitoring system. By asking for the “temp” the WAVR responded with the value calculated by the chip. This value can be converted into celcius by dividing by 4, then multiplying by .03125. The temperature found was around 22 celsius.

The second test of the functionality of the WAVR consisted of the power mux. When the power from the 9V is disabled the power mux should stream continuous power to the circuit with no interruption. This can be witnessed by disabling the battery when the WAVR is sleeping or by viewing the GREEN enable leds that are fed to other regulators. Upon pulling the battery from the circuit the WAVR did not wake up from sleep, as would happen given a shutdown, and the green led’s did not turn off then back on. On restart these led’s would turn off for three seconds before being re-enabled from good battery voltage and good temperature readings.

**Assessment:**

The testing of the WAVR was successful in every aspect. We successfully established a serial connection between the WAVR and the BeagleBone, communicated asking for certain internal values such as time, date and temperature, and set the internal real time clock to a correct value.

The test of power source was also successful. When pulling power from the TI power mux input 1, the second input was immediately sourced through and no circuit shut down or froze. This proves we can safely monitor the battery voltages needed to power the entire module and if need be, cut off main power and switch to a backup without causing a fatal restart to the WAVR.

**4.0 Schematics.**

\*Note: The attached schematics are the most updated schematics for the entire PCB. The parts and relays used in the test are all still prevalent in the schematics, however there are many more data lines and peripherals than the schematics attached to the test plan. The reason for the new schematics is to illustrate the forward progress after the test.