**UNSW Student Satellite Project: BLUEsat**

Year 2011

**Telemetry Systems**

**Overall Team Documentation**

**Andy Cheng, Samuel Ho, James Qin, Thomas Young**

**What Is the Telemetry System?**

The telemetry system consists of monitoring Temperature, Current and Voltage on the satellite. The system mainly revolves around Analogue to Digital convertors.

**Main Components of Telemetry**

Three different chips required for the functioning of the telemetry system are as follow:

* MAX127 - An 8 Channel, 12 bit-resolution chip which sends voltage information to CSC and is connected to the I2C bus.
* ADM4073F - A current to voltage convertor which connects to the MAX127 via one of the 8 channels
* AD590 - A sensor which converts temperature to current which connects to the ADM4073F which in turn is linked to the MAX127 etc.

**Further Information on Each Chip**

For our purposes the MAX127 chip we are using are MAX127BCNG which is slightly different to the MAX126ANCG. Version B allows higher resolution and more accurate data handling in comparison to the version A. Thus when MAX127 is referred to it is the MAX127BCNG chip.

The MAX127 chip has 7 bit addressing in which the first 4 bits are set |1010| and the last 3 bits are variable for our addressing purposes. There will be 8 individual sensors which will be connected to the MAX127 via the 8 channels on the chip. Each sensor can be called by their individual channels i.e. Channel 0, channel 1 are different sensors yet on the same telemetry board.

The AD590 requires 5V to operate and generates a current in micro amps. It initialised to approximately 300K which is approximately 300mA and fluctuates 1mA per K.

**Requirements on Telemetry**

The main requirement on the telemetry system is to:

* Effectively gather accurate telemetry data from sensors
* Effectively send telemetry data when requested
* Effectively store telemetry data
* Be able to obtain acknowledgement from CSC to maintain health status of each sensor
* Have 80 functional sensors

There is a requirement to be able to turn off any chip in the telemetry system. This is in order to:

* Save power
* Disable fried chips which can potentially short the circuit it is connected to.

**Further Information on Telemetry Components**

There will be two I2C buses for the telemetry system's functions which allow separation of sensor placement rather than putting all the sensors on one I2C bus. Each I2C bus is able to hold 64 sensors due to the 3 bit addressing of the MAX127, so there will only be 8 MAX127s on one I2C bus.

The Port Expander PCM with an initial 4 bit address of |0101| will handle disabling and enabling any chips. The Port Expander will be connected to the I2C and analogue switch for power; it is also in between any slave and I2C line. The PCM is similar to a switch in nature by using 1s and 0s to control each port for on and off functionality respectively. It is also important to note that even when the PCM has allowed a chip to be on, the chip can be set to sleep mode to save power and can be woken up to collect data without a warm-up period e.g. sensor in sleep mode -> send data -> sleep.

In order to prevent a slave (Telemetry board) from holding the line,

* There will be no direct connection from the slave to the I2C
* In between the I2C lines and the slaves there will be an additional isolation switch.

**Circuit Diagrams**

Circuit diagrams of all hardware will be recorded in this section to maintain an overall archive on the whole telemetry development process.

The following diagram is of the telemetry board including the MAX127BCNG and is the latest of 21/05/2010. The circuit diagram is for record purposes and the resolution is not for actual use and may be subjected to change. Please refer to the wiki for more updated information on circuit diagrams.

****

**Data Polling**

Every pattern is stored as 128 bits of data with each bit representing one potential sensor, a ‘1’ in the pattern variable is a signal that the sensor is to be polled in the sweep, while a ‘0’ is the opposite. In order to compute bitwise comparisons a bitmask is also introduced (of equal size) to check for each sensor.

An Example:

|  |
| --- |
| char bitmask = 0b10000000;  char bitmask = 0b11000001;  for (i = 0; i < 8; i++) {  if (bitmask & pattern != 0) {  poll();  }  bitmask = bitmask >> 1;  }  return; |

The pattern is '1' in the first two bits and the last one, thus only sensors 1, 2 and 8 will be measured, while the rest are '0' and thus ignored. The main advantage of this method is that the pattern can be scaled up and down (along with the accompanying comparison logic) to fit however many sensors are needed and patterns that have to be encoded can merely be added into the definitions of the patterns (such as making char pattern an array to sequentially store more patterns).

There is also the possibility of adding in a pointer to some dynamic point in memory to allow external users to poll the sensors they require in order to obtain a piece of information the pre-programmed patterns may not allow.

**Storage**

The telemetry system will be allocated 8 megabytes of data onboard the Critical System Computer. This limited space requires the compression algorithms and space-saving algorithms needed to be applied in the way Telemetry Data is to be stored onboard the satellite.

As of July 2011, a concrete algorithm has not been determined. The following is a space for ideas and evaluation of those ideas.

**James’ Method**

James’ method revolves around the Huffman coding for compression with tables of data compression and data sum checks.

**Reading Period**

An initial value is collected and the next value collected will always be the difference from the value before it. This occurs to prevent reliance on the initial value. An actual value can be inserted every set interval so all values are not reliant on each other but up to every 5 reading or set interval.

**Compression Period**

Since the satellite may require instantaneous data, it is not immediately compressed. After 24-48hrs (to be set later), each difference in temperature will be given a symbol and a frequency tally will be made.

It is then to follow a Huffman coding compression which creates a table with Huffman compression. This data then becomes a basis of what information is to be sent to ground station.

**Collect Period**

This mainly involves sending the data to ground station either by request or by set intervals. It is then possible to add a sum check to the Huffman table before sending off to increase validity of the data. After the data is sent off and a confirmation is received, the data will be cleared.

**Huffman Coding**

Huffman Coding is an entropy encoding algorithm used for lossless data compression. Huffman Coding mainly revolves around the use of a variable-lengthen code table for encoding a source symbol. For more information please visit Wikipedia.

**Last Edited**

Samuel HO – 21/05/2010

* Creation of Documentation and expanded certain areas.

**Revised and edited**

Ekram-Ul-Islam Nafis (24/09/2011)

* Data Polling