CAI UM216

LCD Keypad Module

Data Written: 23 June 2009

Issue: Issue A

Firmware Version: V1.0.0

Hardware Version: V1.0.0

Contents

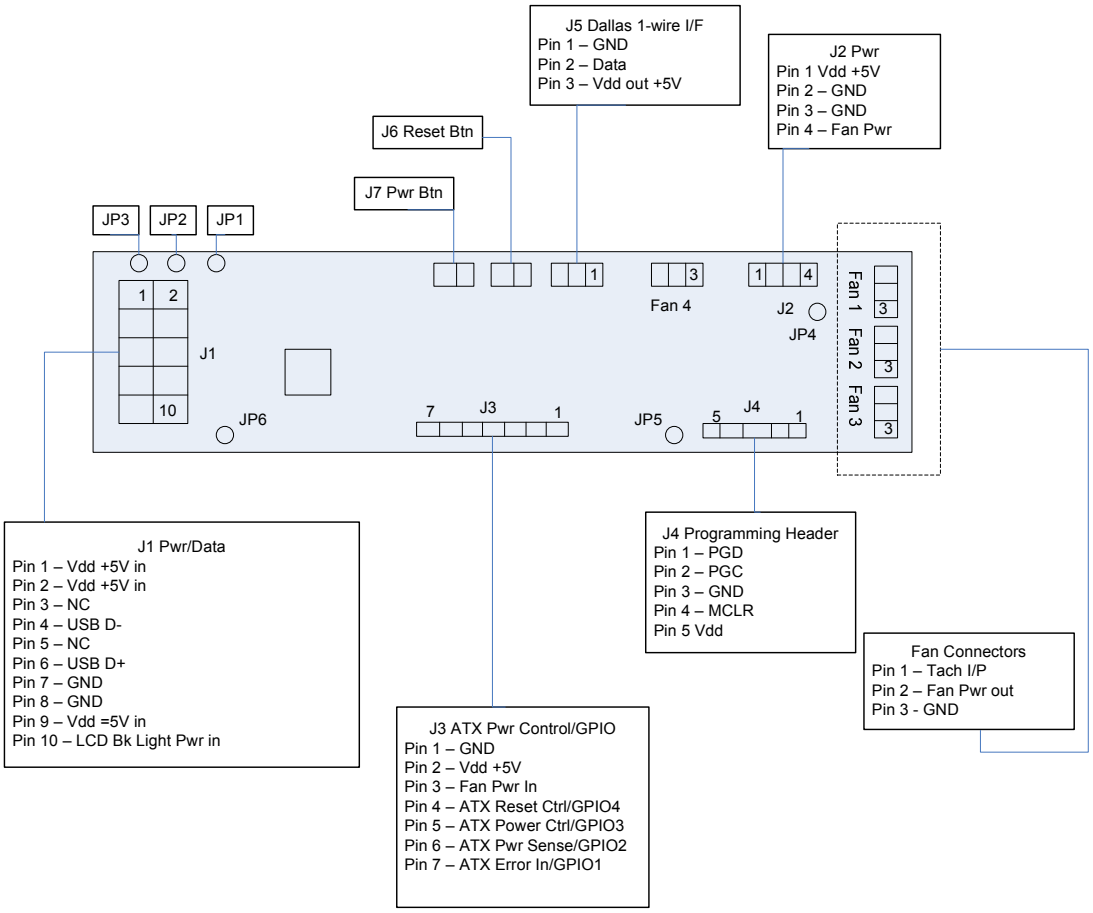
1 Introduction

This document provides a detailed over view of all functionality offered by the UM216 LCD keypad module. It also details the communications protocol and command set used to control the UM216 for a host PC. The document starts by outlining the core functionality offered by the UM216 but the reader is directed to section 4 the host communications section for a detailed description on how to use each function.

1.1 Abbreviations

- NVM – Non volatile memory
- DOW – Dallas one wire
- SCD – special character data
- JP – jumper
- LCD – liquid crystal display
- GPIO – general purpose input output
- RPM – revolutions per minute
- PPR – pulses per revolution

2 Connections/Pin-out



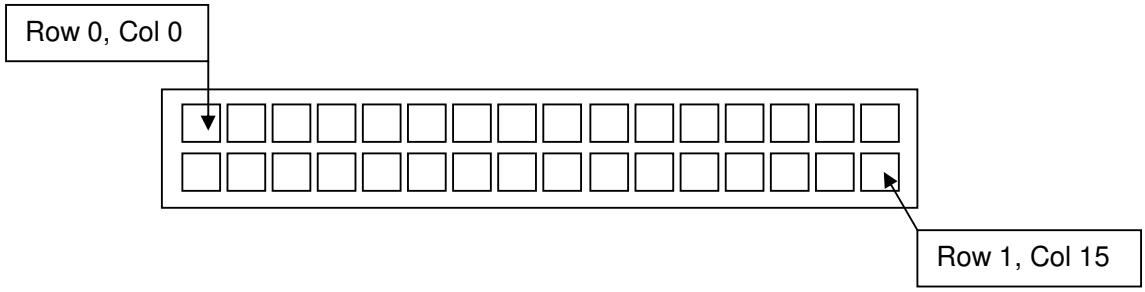
Jumper Settings	
JP1	***Close this to power UM216 from USB***
JP2	***Close this to power LCD backlight from USB***
JP3	Close this to power LCD backlight from pin 10 on J1
JP4	Close this to power UM216 from external 5V supply on pin 5 of J4
JP5	Close this to power UM216 from external 5V supply on pin 2 of J3
JP6	Close this to power UM216 from external 5V supply on pin 9 of J1

Note those marked in bold and italic are the default jumper settings

3 Core Functionality

3.1 LCD Display

The UM216 has a 2 line 16 character LCD display. Row 1 is indexed as 0 and row 2 indexed as 1. Each row has its start character indexed at position 0. This is shown in the diagram below.



The display has two control modes a start-up mode and a host mode. The display is also capable of displaying fan and temperature readings directly. The backlight and scrolling speed of the LCD can be set via a host command.

3.1.1 Start-up Mode

When power is first applied to the UM216 it enters its start-up display mode. It stays in this mode until it receives a command from the host. In start-up display mode the UM216 cycles through two display screens and displays a start-up message to the user. These two start-up message screen strings are configurable by host command. The maximum length of text on each row of the start-up display screens is limited to 16 characters. By default the UM216 will display its firmware version in the start-up display. If the start-up display message is changed and it is wished that the firmware version still be shown then use the “\F” escape sequence when specifying a string. If you require a normal back slash in the string the use it twice e.g. “\\” will print as “\” on the display.

3.1.2 Host Mode

When the UM216 first receives a command from the host, it enters the host display mode. In this mode it is the host’s responsibility for what is displayed on the LCD. The host can write up to 32 characters to each row of the LCD. When there are more than 16 characters written to a row then the UM216 will scroll that row. The scroll speed is adjustable via a host command.

3.1.3 Live Display

The UM216 can be configured to automatically update a potion of the display with a ‘live’ fan RPM or temperature reading. The live display is configured using a host command. The live display is based on the idea of display slots. There are 8 possible slots that may be enabled or disabled independently. A slot may be configured to display any possible data that is available for example slot 0 may be configured to display a 5 digit RPM of fan 1 and slot 7 configured to display the temperature is degrees Celsius from temperature sensor 5.

3.2 Fan Monitoring

The UM216 is capable of monitoring the RPM of up to 4 fans with tachometers. In order for the UM216 to calculate the RPM of a fan accurately the pulses per revolution of the fans tachometer must be configured using a host command. The default PPR the UM216 will use for each fan is 1.

3.2.1 Host Reporting

The UM216 can be configured to report the fan speed of all fans every 500ms. The reporting function can be individually enabled or disabled per fan. The UM216 also as the ability to send a warning to the host when the speed of a fan falls below 70% of the fans memorized required operating speed. The UM216 is instructed to remember the required operating speed of the fan by running the fan at the desired speed and sending the UM216 a stored current speed command.

3.3 Temperature Monitoring

The UM216 is able to monitor up to 32 Dallas One-Wire temperature sensors that are connected to the DOW interface connector J5. The current temperature sensor types supported are the DS18S20 (0x10), DS1822 (0x22) and DS18B20 (0x28).

3.3.1 Host Reporting

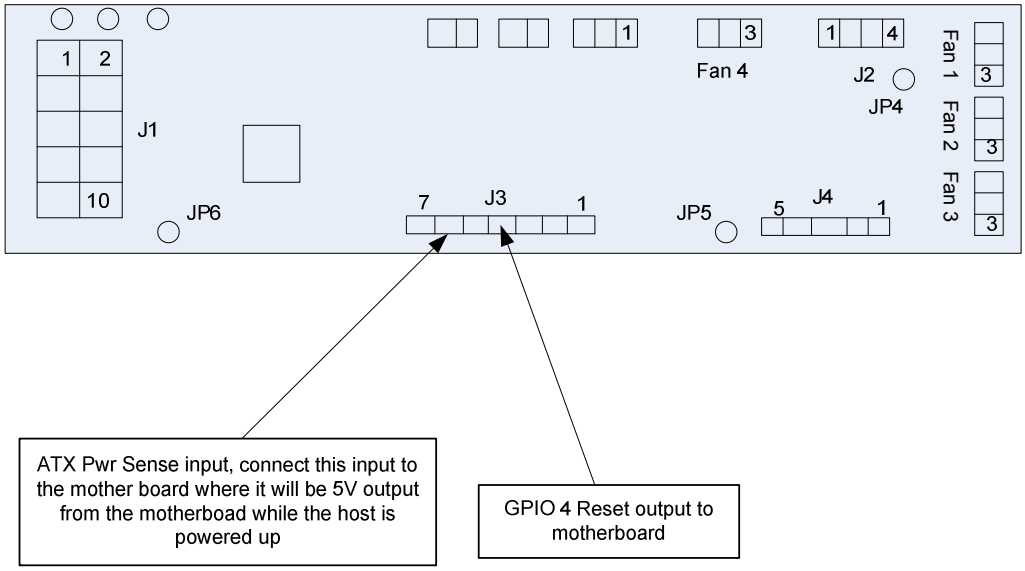
The UM216 can report the temperature of selected temperature sensors tot the host. Host reports can be enabled or disabled individually for each temperature sensor. When enabled the UM216 will report the current temperature reading of a sensor every 1 second.

3.4 ATX Power Switch

The UM216 has the ability to power on/off and reset the host. It does this by connecting the GPIO pins of J3 to the reset and power on inputs on the host's mother board. The UM216 also requires a power sense +5V from the host's motherboard in order for it to detect when the host is powered up. All ATX power switch functionality must be enabled by host commands. The following can be enabled or disabled individually or they can all be enabled or disabled. Some of these functions generate host reports.

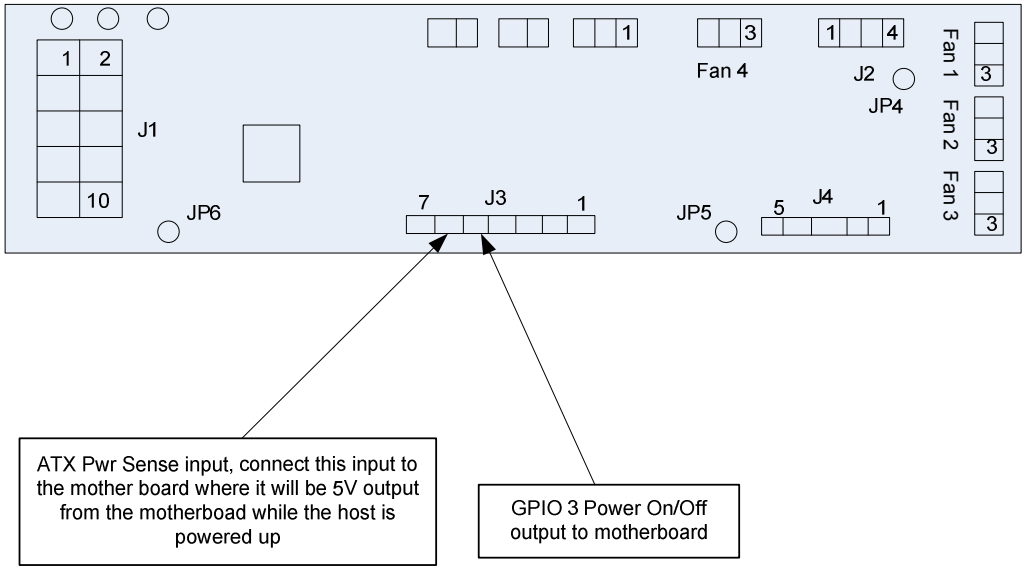
3.4.1 Keypad Reset

If keypad reset function is enabled and the power sense input is high, then holding the green check key on the keypad down for 4 seconds will cause the UM216 to pulse the reset GPIO line to reset the host. The UM216 will then reset it's self. The following diagram shows the connections needed for this:



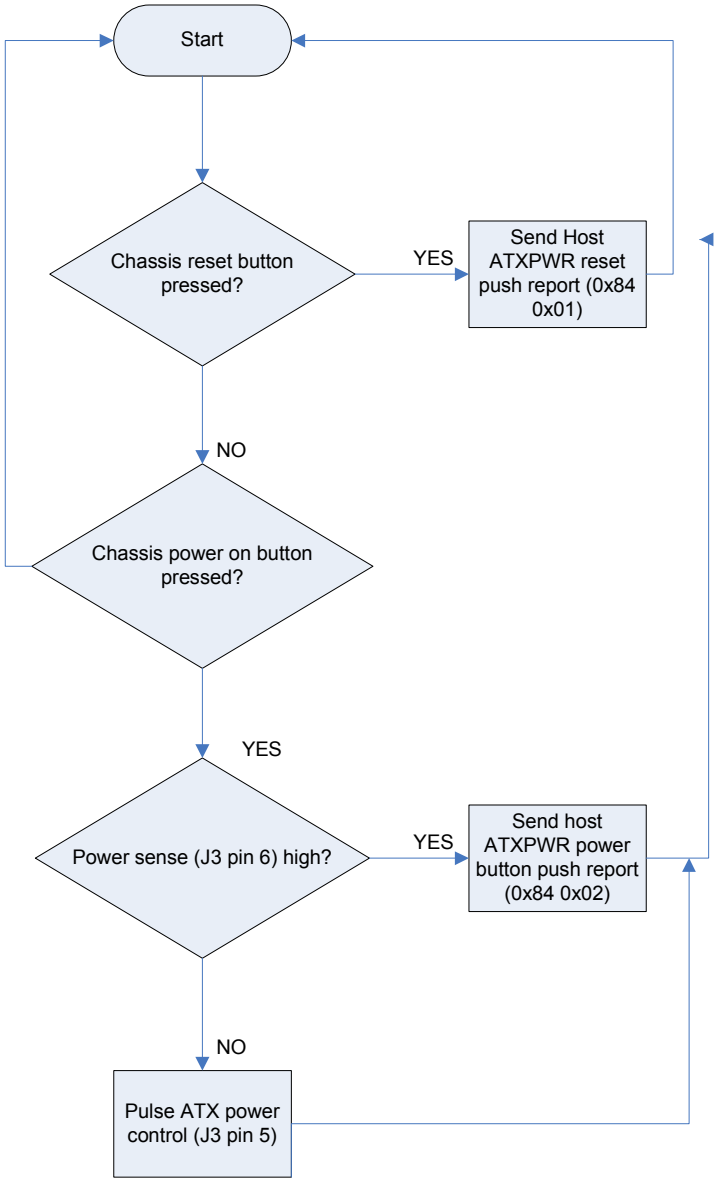
3.4.2 Keypad Power On/Off

If keypad power on/off is enabled then the following logic applies. When power sense input is low and the green check key is held down for 0.25 seconds the UM216 will pulse the power on output to switch the host on. After this the UM216 will reset it's self. The connections required for this are:

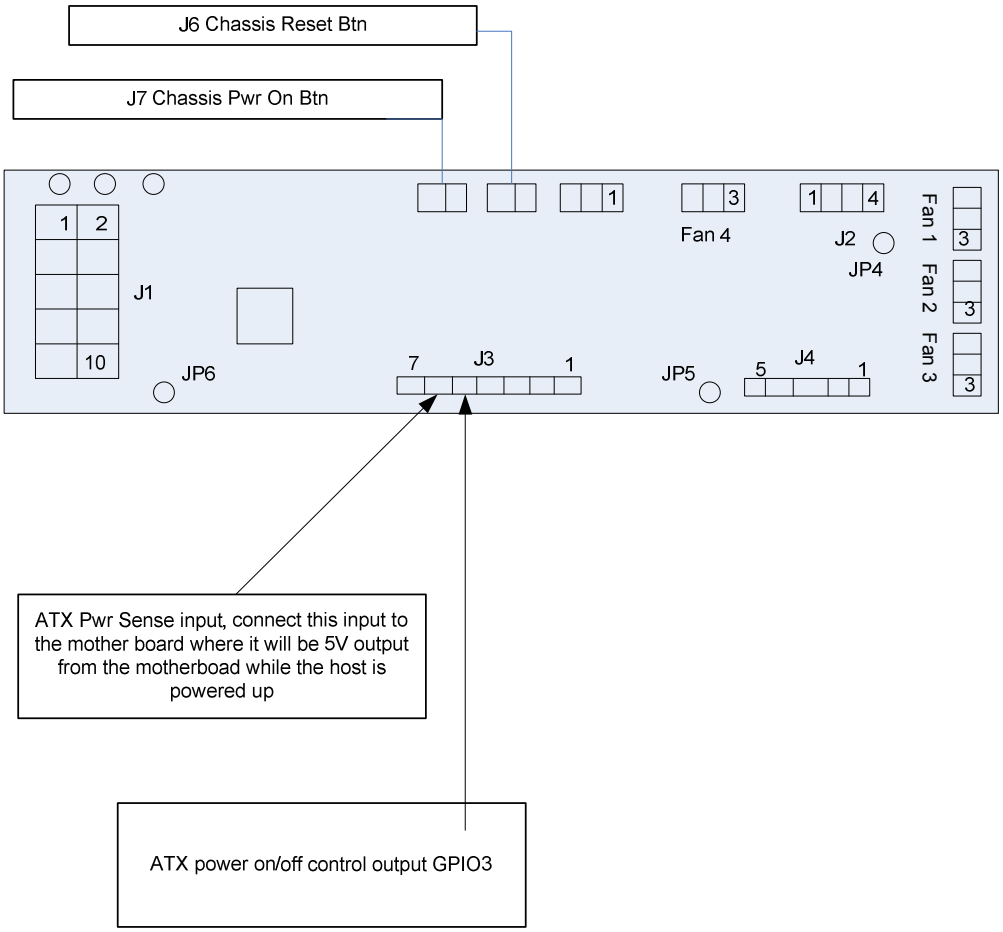


3.4.3 Chassis Switch Monitoring

The UM216 is able to capture the chassis reset and power on button presses. Upon doing this it will issue host commands or set a GPIO pin high to the motherboard to power on the host. The following flow diagram shows how this feature operates:

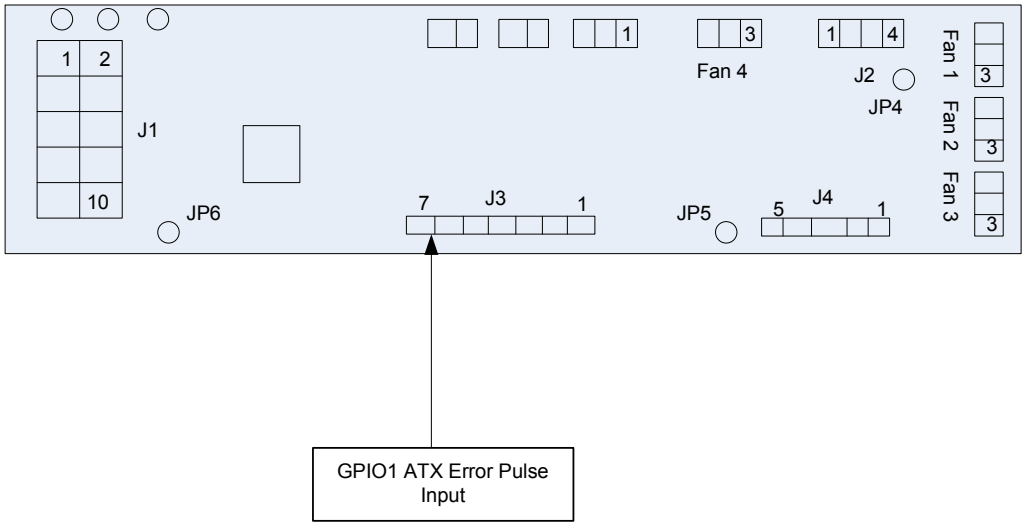


The following diagram shows the connections to make between the mother board and chassis buttons to enable this feature:



3.4.4 External Error Signal Input

The error signal input can be used to signal a PSU failure to the host. The input expects to be pulsed 10 times before it will enter an error state and sends an ATX failure report to the host. Once the input stops being pulses for 5 seconds the UM216 will exit the error state and issue and ATX OK report to the host. The error input is a TTL level input so the pulses must be 0 - +5V dc only. The required connections are shown in the diagram below:



3.5 GPIO

The UM216 has 4 general input output pins on J3 that can be used for GPIO as long as they are not being used by the ATX power switch functions. The GPIO pins are set and read via host commands. GPIO1 and GPIO2 have pull up resistors on them and GPIO3 and GPIO4 both have pull down resistors fitted to them.

3.6 Keypad

The UM216 has a 6 button keypad with backlight. The keypad can be polled by the host or it can send the host reports when the user presses or releases a key. The UM216 sends these host reports by default but they can be individually enabled or disabled per key. The keypad backlight brightness is set by the same command as used to set the LCD backlight; the two backlights should be at the same level.

3.7 Non-Volatile Memory (NVM)

The UM216 has a 256 byte EEPROM that is used to stored configuration items. When changes to the configuration are made they are not stored to the NVM until a store configuration command is sent by the host. The UM216 can be forced to reset its configuration and wipe its EEPROM if the host sends a restore defaults command.

3.7.1 User Flash

The UM216 reserves 16 bytes of it's NVM for use by the user to store arbitrary data. This data is read and written using host commands.

4 Host Communications

UM612 communicates with its host using a serial over USB interface. When the UM216 is connected to a host PC it will appear as a standard serial port; this is achieved by using the communications class diver (CDC).

4.1 Packet Structure

All communication between the UM612 and the host takes place in the form of a simple and robust CRC checked packet. All packets have the following structure:

```
<cmd><data_length><data><CRC>
```

`Cmd –` this is a one byte field, bits 5 - 0 identify the function of the packet. Bits 7 and 6 are used to identify the type of the packet:

```
<TTcc cccc>

   || ||||--Command, response, error or report code

||---------Type:
```

00 = normal command from host to UM216

01 = normal response from UM216 to host

10 = normal report from UM216 to host

11 = error response from UM216 to host

`data_length –` this is a one byte field that specifies the number of bytes that will follow in the data field. The valid range of data_length is 0 to 32.

`data` – this is the payload of the packet. The size of this field is a maximum of 32 bytes and the data structure is command specific.

`CRC` – this is a standard 16-bit CRC of all the bytes in the packet except the CRC itself. The CRC is sent LSB first.

4.1.1 Calculating the CRC16

This example C code demonstrates how to calculate the CRC16. The CRC is calculated on all bytes of the packet excluding the CRC it's self.

```
UINT16 TRANSPORT_get_calculated_crc(CHAR* packet, UINT8 len)
{
      UINT32 newCRC;
      CHAR data;
      UINT8 bit_count;

      /* This seed makes the output of this shift based algorithm match
       * the table based algorithm. The center 16 bits of the 32-bit
```

```
        * "newCRC" are used for the CRC. The MSb of the lower byte is used
        * to see what bit was shifted out of the center 16 bit CRC
        * accumulator ("carry flag analog") */
       newCRC=0x00F32100;

       while(len-- > 0)
       {
             /*Get the next byte in the stream */
             data= *packet++;
             /* Push this byte's bits through a software
              * implementation of a hardware shift & xor */
             for(bit_count=0;bit_count<=7;bit_count++)
             {
                   /* Shift the CRC accumulator */
                   newCRC>>=1;
                   /* The new MSB of the CRC accumulator comes
                    * from the LSB of the current data byte */
                   if( (data & 0x01) > 0)
                   {
                         newCRC|=0x00800000;
                   }
                   /* If the low bit of the current CRC accumulator was set
                    * before the shift, then we need to XOR the accumulator
                    * with the polynomial (center 16 bits of 0x00840800) */
                   if( (newCRC&0x00000080) > 0)
                   {
                         newCRC^=0x00840800;
                   }
                   /* Shift the data byte to put the next bit of the stream
                    * into position 0 */
                   data>>=1;
             }
       }
       /* All the data has been done. Do 16 more bits of 0 data */
       for(bit_count=0;bit_count<=15;bit_count++)
       {
       /*Shift the CRC accumulator */
             newCRC>>=1;

             /* If the low bit of the current CRC accumulator was set
              * before the shift we need to XOR the accumulator with
              * 0x00840800 */
             if( (newCRC&0x00000080) > 0)
             newCRC^=0x00840800;
       }

       /* Return the center 16 bits, making this CRC match the one's
        * complement that is sent in the packet */
       return((~newCRC)>>8);
}
```

4.2 Host Reports

The UM216 can be configured to send up to 4 report items to the host. These reports are send asynchronously at a period specific to each report. The 2 bit type mask in the command of the packet will be set to ‘10’ to identify the packet as a report. Reports are not sent in response to any particular command.

4.2.1 Key Activity Report

If configured the key activity report is send when ever a key on the keypad is pressed or released. Each key can have event reporting enabled or disabled. The format of the report is:

```
cmd = 0x00 | 0x80

data_length = 1

data[0] – is the type of keyboard activity:

 KEY_UP_PRESS              1

 KEY_DOWN_PRESS            2

 KEY_LEFT_PRESS            3

 KEY_RIGHT_PRESS           4

 KEY_ENTER_PRESS           5

 KEY_EXIT_PRESS            6

 KEY_UP_RELEASE            7

 KEY_DOWN_RELEASE          8

 KEY_LEFT_RELEASE          9

 KEY_RIGHT_RELEASE        10

 KEY_ENTER_RELEASE        11

 KEY_EXIT_RELEASE         12
```

4.2.2 Fan Speed Report

If configured the UM216 will send fan speed reports for each enabled fan every 500ms. The structure of the report is:

```
cmd = 0x01 | 0x80

data_length = 4

data[0] – is the index of the fan being reported:
```

```
 0 = FAN 1

 1 = FAN 2

 2 = FAN 3

 3 = FAN 4

data[1] Fan RPM (MSB)

data[2] Fan RPM (LSB)

data[3] Fan speed status, 1 = OK, 0 = fan to slow
```

Note that in order for the fan speed to be reported accurately it is necessary to set the fan pulse per revolution for each fan. This is done using the set PPR command. Also in order for the fan speed status to be set correctly the UM216 must memorise its max fan RPM to monitor for each fan, this is done by sending a store fan speed command. Data[3] will then be set to 0 if the fan speed falls below 70% of this memorised fan RPM.

4.2.3 Temperature Sensor Report

The UM216 can be configured to report the temperature on degrees Celsius for each of its temperature sensors (max 32) every 1 second. The format of the report is:

```
cmd = 0x02 | 0x80

data_length = 4

data[0] - is the index of the temperature sensor being

          reported:

 0 = temperature sensor 1

 1 = temperature sensor 2

 . . .

 31 = temperature sensor 32

data[1] is the MSB of the temperature value

data[2] is the LSB of the temperature value
```

4.2.4 ATX Power Report

Each ATX power control feature can be configured to send a report on a control event. The format of the report is:

```
cmd = 0x03 | 0x80
```

```
data_length = 1

data[0] - the atx power event code can be one of the
          following:
                0x01 - the mother board reset button has been
                       Pushed
                0x02 - the mother board power button has been
                       Pushed
                0x03 - the error input has been pulsed 10
                      times indicating an error on the
                      external PSU
                0x04 - the error input has stopped being
                       pulsed for 10 seconds indication
                      external PSU is now OK
```

4.3 Command Codes

Each command packet is answered by either a response packet or an error packet. The lower 6 bits of the `cmd` field of the response or error packet is the same as the lower 6 bits of the `cmd` field of the command packet being acknowledged.

4.3.1 Ping Command

The UM216 will return the Ping Command to the host. The command structure is:

```
cmd = 0

valid data_length is 0 to 36

data[] - can be filled with any arbitrary data
```

The return packet is identical to the packet sent, except the type will be 0x40 (normal response, Ping Command):

```
cmd = 0x40 | 0

data_length = (identical to received packet)

data[identical to received packet]
```

4.3.2 Get Version

The UM216 will return the hardware and firmware version information to the host. The command structure is:

```
cmd = 1

valid data_length is 0
```

The return packet will be:

```
cmd = 0x40 | 1

data_length = 17

data[] = "UM216:hX.X,fY.Y.Y"


hX.X is the hardware revision, "1.2" for example

fY.Y.Y is the firmware version, "1.2.9" for example
```

4.3.3 Write User Flash

The UM216 reserves 16 bytes of non-volatile memory for arbitary use by the host. All 16 bytes must be written in this command. The command structure is:

```
cmd = 2

data_length = 16

data[] = 16 bytes of arbitrary user data to be stored in non-

          volatile memory
```

The return packet will be:

```
cmd = 0x40 | 2

data_length = 0
```

4.3.4 Read User Flash

This command will read the UM216's user non-volatile storage area and return the data to the host. The command structure is:

```
cmd = 3

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 3
```

```
data_length = 16

data[] = 16 bytes user data recalled from the UM216's
         Non-volatile memory
```

4.3.5 Save Configuration

This command will save the following configuration items into Non-volatile memory and will reload then at boot up when power is applied:

- LCD power up display strings
- Display scroll speed
- Display brightness
- Current cursor position
- Current cursor style
- Live display configuration
- LCD special character data
- Key reporting configuration
- Fan reporting configuration
- Temperature reporting configuration
- ATX power control and reporting configuration

The format of the command is:

```
cmd = 4

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 4

data_length = 0
```

4.3.6 Reboot

This command instructs the UM216 to either, reboot the host, power off the host, or reboot it's self.

To reboot the UM216 send the following packet:

```
cmd = 5
```

```
data_length = 3

data[0] = 8

data[1] = 18

data[2] = 99
```

To reboot the host, the hosts reset line must be connected to GPIO[3], pin 4 of J3. Send the following packet to initiate the host reset:

```
cmd = 5

data_length = 3

data[0] = 12

data[1] = 28

data[2] = 97
```

To power off the host, the hosts power control line must be connected to GPIO[2], pin 5 of J3. Send the following packet to initiate the host power off:

```
cmd = 5

data_length = 3

data[0] = 3

data[1] = 11

data[2] = 95
```

The response packet to all commands above will be:

```
cmd = 0x40 | 5

data_length = 0
```

4.3.7 Clear LCD

This will clear the contents of the LCD. The command structure is:

```
cmd = 6

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 6

data_length = 0
```

4.3.8 Set LCD Contents Line 1

Writes to row 1 of the LCD, each line of the LCD can accept a string of up to 32 characters. If the string length is greater than 16 characters then the LCD will automatically scroll the row. The command structure is:

```
cmd = 7

data_length = 0 - 32

data[] = row 1 display content
```

The return packet will be:

```
cmd = 0x40 | 7

data_length = 0
```

4.3.9 Set LCD Contents Line 2

Writes to row 2 of the LCD, each line of the LCD can accept a string of up to 32 characters. If the string length is greater than 16 characters then the LCD will automatically scroll the row. The command structure is:

```
cmd = 8

data_length = 0 - 32

data[] = row 1 display content
```

The return packet will be:

```
cmd = 0x40 | 8

data_length = 0
```

4.3.10 Set LCD Special Character Data

Sets the special character definition for one of the special characters of the LCD's CGRAM. Format of command is:

```
cmd = 9

data_length = 9

data[0] = index of special character that you would like

          to modify, 0-7 are valid

data[1-8] = bitmap of the new font for this character
```

data[1-8] is the bitmap information for this character. The msb is at the left of the character cell of the row, and the lsb is at the right of the character cell. data[1] is at the top of the cell, data[8] is at the bottom of the cell.

The return packet will be:

```
cmd = 0x40 | 9

data_length = 0
```

4.3.11 Read 8 Bytes of LCD Memory

This command will return the contents of the LCD's DDRAM or CGRAM, and is intended for debugging. The format of the command is:

```
cmd = 10

data_length = 1

data[0] = address code of desired data
```

The following address codes are valid:

- 0x40 (\064) to 0x7F (\127) for CGRAM
- 0x80 (\128) to 0x93 (\147) for DDRAM, line 1
- 0xC0 (\192) to 0xD3 (\211) for DDRAM, line 2

The return packet will be:

```
cmd = 0x40 | 10

data_length = 9

data[0] - the address code.

data[1-8] - the requested data
```

4.3.12 Set LCD Cursor Position

This command sets the position of the cursor on the display. It is necessary to select the cursor style in order for the cursor to be made visible (see select cursor style command). The format of the command is:

```
cmd = 11

data_length = 2

data[0] = column (0-16 valid)

data[1] = row (0-1 valid)
```

The return packet will be:

```
type = 0x40 | 11

data_length = 0
```

4.3.13 Set LCD Cursor Style

This command sets the LCD hardware generated cursor style. The format of the command is:

```
cmd = 12

data_length = 1

data[0] = cursor style (0-2 valid)

            0 = no cursor

            1 = blinking block cursor

            2 = underscore
```

The return packet will be:

```
cmd = 0x40 | 12

data_length = 0
```

4.3.14 Set LCD and Keypad Backlight

This command sets the brightness of the LCD and keypad backlights. The format of the command is:

```
cmd = 14

data_length is 1

data[0] = backlight power setting (0-100 valid)

            0 = fully off

            1-99 = variable brightness

            100 = fully on
```

The return packet will be:

```
cmd = 0x40 | 14

data_length = 0
```

4.3.15 Setup Fan Reporting

This command configures the UM216 to report fan speed information every 500ms for each of its four fans. The command structure is:

```
cmd = 16

data_length = 1

data[0] = bitmask indicating which fans are enabled to

          report (0-15 valid)

---- 8421 Enable Reporting of this Fan's Tach Input

|||| ||||-- Fan 1: 1 = enable, 0 = disable

|||| |||--- Fan 2: 1 = enable, 0 = disable

|||| ||---- Fan 3: 1 = enable, 0 = disable

|||| |----- Fan 4: 1 = enable, 0 = disable
```

The return packet will be:

```
cmd = 0x40 | 16

data_length = 0
```

If the bitmask in data[0] is not 0 then the UM216 will start sending fan speed reports every 500ms.

4.3.16 Read DOW Device Information

This command reads the one of the Dallas one wire device's 8 byte ROM code. If there are no devices connected to the one wire interface or the requested index is empty a zero ROM code will be returned. The format of the command is:

```
cmd = 18

data_length = 1

data[0] = device index (0-31 valid)
```

The return packet will be:

```
type = 0x40 | 18

data_length = 9
```

```
data[0] = device index (0-31 valid)

data[1-8] = ROM ID of the device
```

4.3.17 Setup Temperature Reporting

This command will configure the UM216 to report the temperature of selected temperature sensors every 1 second. The format of the command is:

```
cmd = 19

data_length = 4

data[0-3] = 32-bit bitmask indicating which temperature
            sensors are enabled to report (0-255
            valid in each location)
```

data[0]

08 07 06 05 04 03 02 01 Enable Reporting of sensor with

| | | | | | | | device index of:

| | | | | | | |-- 0: 1 = enable, 0 = disable

| | | | | | |----- 1: 1 = enable, 0 = disable

| | | | | |--------- 2: 1 = enable, 0 = disable

| | | | |---------- 3: 1 = enable, 0 = disable

| | | |---------- 4: 1 = enable, 0 = disable

| | |---------- 5: 1 = enable, 0 = disable

| |---------- 6: 1 = enable, 0 = disable

|---------- 7: 1 = enable, 0 = disable

data[1]

16 15 14 13 12 11 10 09 Enable Reporting of sensor with

| | | | | | | | device index of:

| | | | | | | |-- 8: 1 = enable, 0 = disable

| | | | | | |---- 9: 1 = enable, 0 = disable

| | | | | |------- 10: 1 = enable, 0 = disable

| | | | |--------- 11: 1 = enable, 0 = disable

| | | |------------- 12: 1 = enable, 0 = disable

| | |--------------- 13: 1 = enable, 0 = disable

| |------------------ 14: 1 = enable, 0 = disable

|-------------------- 15: 1 = enable, 0 = disable

data[2]

24 23 22 21 20 19 18 17 Enable Reporting of sensor with

| | | | | | | | device index of:

| | | | | | | |-- 16: 1 = enable, 0 = disable

| | | | | | |---- 17: 1 = enable, 0 = disable

| | | | | |------- 18: 1 = enable, 0 = disable

| | | | |--------- 19: 1 = enable, 0 = disable

| | | |-------------- 20: 1 = enable, 0 = disable

| | |---------------- 21: 1 = enable, 0 = disable

| |------------------- 22: 1 = enable, 0 = disable

|-------------------- 23: 1 = enable, 0 = disable

data[3]

32 31 30 29 28 27 26 25 Enable Reporting of sensor with

| | | | | | | | device index of:

| | | | | | | |-- 24: 1 = enable, 0 = disable

| | | | | | |---- 25: 1 = enable, 0 = disable

| | | | | |------- 26: 1 = enable, 0 = disable

| | | | |--------- 27: 1 = enable, 0 = disable

| | | |-------------- 28: 1 = enable, 0 = disable

| | |---------------- 29: 1 = enable, 0 = disables

| |------------------- 30: 1 = enable, 0 = disable

|---------- 31: 1 = enable, 0 = disable

The return packet will be:

```
type = 0x40 | 19

data_length = 0
```

Note that the one wire temperature sensors used must be detected as type DS18S20 (0x10), DS1822 (0x22) or DS18B20 (0x28).

4.3.18 Setup Live Display

This command configures the live display functionality of the UM216 to display fan or temperature sensor readings automatically on part of the LCD. The live display is based on a concept of display slots. There are 8 slots, and each of the 8 slots may be enabled or disabled independently. Any slot may be requested to display any data that is available. For instance, slot 0 could display temperature sensor 3 in ℃, while slot 1 could simultaneously display temperature sensor 3 in ℉. Any slot may be positioned at any location on the LCD, as long as all the digits of that slot fall fully within the display area. It is legal to have the display area of one slot overlap the display area of another slot, but senseless. This situation should be avoided in order to have meaningful information displayed. The format of the command is:

```
cmd = 21

data_length = 7

data[0]: display slot (0-7)

data[1]: type of item to display in this slot

           0 = nothing

           1 = fan tachometer RPM

           2 = temperature

data[2]: index of the sensor to display in this slot:

           0-3  are valid for fans

           0-31 are valid for temperatures

data[3]: number of digits

    for a fan: 4 digits (0 to  9999) valid fan speed range

    for a fan: 5 digits (0 to 50000) valid fan speed range

    for a temperature: 3 digits (  -XX or   XXX)

    for a temperature: 5 digits (-XX.X or XXX.X)
```

```
data[4]: display column

           0-13 valid for a 3-digit temperature

           0-12 valid for a 4-digit fan

           0-11 valid for a 5-digit fan or temperature

data[5]: display row (0-1 valid)

data[6]: temperature sensor units

         (0 = deg C, 1 = deg F)
```

The return packet will be:

```
cmd = 0x40 | 21

data_length = 0
```

4.3.19 Send Command Directly to LCD

This command is indented for debugging and writes an 8 bit command to the LCD controller. The format of the command is:

```
cmd = 22

data_length = 2

data[0]: data

data[1]: type (instruction or data)
```

The return packet will be:

```
cmd = 0x40 | 22

data_length = 0
```

4.3.20 Configure Key Reporting

This command enables or disables individual key event reports for each key. By default all events are enabled. The format of the command is:

```
cmd = 23

data_length = 2

data[0]: press mask

data[1]: release mask
```

mask values for keys are:

- KP_UP = 0x01
- KP_ENTER = 0x02
- KP_CANCEL = 0x04
- KP_LEFT = 0x08
- KP_RIGHT = 0x10
- KP_DOWN = 0x20

The return packet will be:

```
cmd = 0x40 | 23

data_length = 0
```

4.3.21 Poll Keyboard

The host can poll the keypad instead of having key event reports enabled. The poll command format is:

```
cmd = 24

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 24

data_length = 3

data[0] = bit mask showing the keys currently pressed

data[1] = bit mask showing the keys that have been pressed
          since the last poll

data[2] = bit mask showing the keys that have been released
          since the last poll
```

mask values for keys are:

- KP_UP = 0x01
- KP_ENTER = 0x02

- KP_CANCEL = 0x04
- KP_LEFT = 0x08
- KP_RIGHT = 0x10
- KP_DOWN = 0x20

4.3.22 Set ATX Power Switch Functionality

This command sets the functions that are enabled in the ATX power switch controller. The command format is:

```
cmd = 28

data_length = 2

data[0]: bit mask of enabled functions

data[1]: length of power on & off pulses in 1/32
         second
            1 = 1/32 sec
            2 = 1/16 sec
           16 = 1/2 sec
            .
            .
           255 = 8 sec
```

Functions mask values are:

- DISABLED = 0x00
- RESET_INVERT = 0x02
- POWER_INVERT = 0x04
- LCD_OFF_IF_HOST_IS_OFF = 0x10
- KEYPAD_RESET = 0x20
- KEYPAD_POWER_ON = 0x40
- KEYPAD_POWER_OFF = 0x80
- ERROR_CHECK = 0x01
- MB_SWITCHES = 0x08

The return packet will be:

```
cmd = 0x40 | 28

data_length = 0
```

4.3.23 Read Configuration

This command reads the configuration of the UM216. The format of the command is:

```
cmd = 30

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 30

data_length = 15

data[0] = fan 1-4 reporting mask

data[1] = temperatures 1-8 reporting mask

data[2] = temperatures 9-15 reporting mask

data[3] = temperatures 16-23 reporting mask

data[4] = temperatures 24-32 reporting mask

data[5] = key presses reporting mask

data[6] = key releases reporting mask

data[7] = ATX Power Switch Functionality mask

data[8] = Not use

data[9] = Not use

data[10] = Not use

data[11] = Not use

data[12] = Not use

data[13] = Not use

data[14] = lcd and keypad backlight setting
```

4.3.24 Write to LCD

This command writes a string to be displayed on the LCD. The maximum string length the LCD can show per row is 32 characters and if the string is greater than 16 characters the UM216 will scroll the row. The format of the command is:

```
cmd = 31

data_length = 1 to 34

data[0]: col to place text 0 - 31 valid

data[1]: row to place text at 0 - 1 valid

data[2-33]: text to place on the LCD, variable from 1 to 32
            characters
```

The return packet will be:

```
cmd = 0x40 | 31

data_length = 0
```

4.3.25 Set GPIO Pin

Sets the state and the configuration of one of the GPIO pins. The format of the command is:

```
cmd = 34

data_length = 3

data[0]: - GPIO pin index to set(0 - 3) valid
           0 = GPIO 1
           1 = GPIO 2
           2 = GPIO 3
           3 = GPIO 4
data[1]: - state to set (0 = off, 1 = on)
data[2]: - drive mode
           0 = standard output
           1 = standard input
```

The return packet will be:

```
cmd = 0x40 | 32

data_length = 0
```

4.3.26 Read GPIO Pin

Reads the state and configuration of a GPIO pin. The format of the command is:

```
cmd = 35

data_length = 1

data[0]: - index of GPIO pin to read (0 - 3) valid

           0 = GPIO 1

           1 = GPIO 2

           2 = GPIO 3

           3 = GPIO 4
```

The return packet will be:

```
cmd = 0x40 | 35

data_length = 3

data[0]: - index of GPIO pin

data[1]: - state (0 = off, 1 = on)

data[2]: - drive mode

           0 = standard output

           1 = standard input
```

4.3.27 Store Current Fan RPM

This command instructs the UM216 to store the current RPM for a fan. The UM216 then issues a warning to the host when the RPM of the fan falls below 70% of this speed for more than 5 seconds. The structure of the command is:

```
cmd = 36

data_length = 2

data[0]: index of the fan to save RPM for (0 - 3 valid)

data[1]: set to 1 to enable host reporting if the fan speed

         falls below 70% of the memorised RPM.
```

The return packet will be:

```
cmd = 0x40 | 36

data_length = 0
```

4.3.28 Set Fan PPR

This command will set the pulses per revolution for a fan's tachometer. The PPR needs to be set to enable the UM216 to calculate the RPM accurately. The command structure is:

```
cmd = 37

data_length = 2

data[0]: index of the fan to set PPR for (0 - 3 valid)

data[1]: PPR value (1 - 4 valid)
```

The return packet will be:

```
cmd = 0x40 | 37

data_length = 0
```

4.3.29 Show Display

Forces the UM216 to switch to the specific display type, command structure is:

```
cmd = 38

data_length = 1

data[0]: display type id:

            0 = start-up display

            1 = host display
```

The return packet will be:

```
cmd = 0x40 | 38

data_length = 0
```

4.3.30 Set Start-up Display

This command sets the text for a portion of the start-up display. The format of the command is:

```
cmd = 39

data_length = 3 to 18

data[0]: screen id (valid values 0 - 1)

data[1]: row (valid values 0 - 1)
```

```
data[2 – 17] – text string
```

The return packet will be:

```
cmd = 0x40 | 39

data_length = 0
```

4.3.31 Set Scroll Rate

This command sets the scroll speed of the display rows. The format of the command is:

```
cmd = 40

data_length = 1

data[0]: scroll rate (valid values 0 – 100)
```

The return packet will be:

```
cmd = 0x40 | 340

data_length = 0
```

4.3.32 Get Serial Number

This command returns the UM216's 16 byte serial number. The format of the command is:

```
cmd = 41

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 41

data_length = 16

data[0 – 15] 16 byte serial number (MSB first)
```

4.3.33 Read Start-up Display

This command reads a row of text from one of the start up display screens. The format of the command is:

```
cmd = 42

data_length = 2

data[0]: start-up display screen (valid values 0 – 1)
```

```
data[1]: start-up screen row (valid values 0 -1)
```

The return packet will be:

```
cmd = 0x40 | 42

data_length = 16

data[0 - 15]: string
```

4.3.34 Restore Defaults

This command invalidates all configuration data stored in the non-volatile memory and returns the configuration to the default state. This command requires that the UM216 be rebooted for the restore to take place. The format of the command is:

```
cmd = 43

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 43

data_length = 0
```

4.3.35 Read Extended Configuration

This command returns more configuration items. The format of the command is:

```
cmd = 44

data_length = 0
```

The return packet will be:

```
cmd = 0x40 | 44

data_length = 15

data[0 - 1]: - stored RPM for fan 1

data[2 - 3]: - stored RPM for fan 2

data[4 - 5]: - stored RPM for fan 3

data[6 - 7}: - stored RPM for fan 4

data[8]: - report low speed fan mask, FAN1 = 1, FAN 2 = 2, FAN
           3 = 4, FAN 4 = 8

data[9]: - fan 1 PPR
```

```
data[10: - fan 2 PPR

data[11]: - fan 3 PPR

data[12]: - fan 4 PPR

data[13]: - display scroll rate

data[14]: - display cursor style
```

4.3.36 Get Live Display Slot

This command returns the configuration for one of the live display slots. The command format is:

```
cmd = 45

data_length = 1

data[0]: the slot to return (0 - 7)
```

The return packet will be:

```
cmd = 0x40 | 45

data_length = 6

data[0]: display slot (0-7)

data[1]: type of item displayed in this slot

           0 = nothing

           1 = fan tachometer RPM

           2 = temperature

data[2]: index of the sensor to displayed in this slot:

           0-3  for fans

           0-31 for temperatures

data[3]: number of digits

            for a fan: 4 digits (0 to  9999)

            for a fan: 5 digits (0 to 50000)

            for a temperature: 3 digits (  -XX or   XXX)

            for a temperature: 5 digits (-XX.X or XXX.X)

data[4]: display column
```

```
                0-13 for a 3-digit temperature

                0-12 for a 4-digit fan

               0-11 for a 5-digit fan or temperature

data[5]: display row

data[6]: temperature units

           0 = deg C, 1 = deg F
```

4.3.37 Get LCD Special Character Data

This command returns the bitmap for one of the special character LCD definitions. The command format is:

```
cmd = 46

data_length = 1

data[0]: - index of special character to get (0 - 7 valid)
```

The return packet will be:

```
cmd = 0x40 | 46

data_length = 8

data[0 - 7]: - bitmap of character definition
```

5 Electrical Characteristics

- Max input voltage into any VDD pin 5V
- Max input voltage into fan power supply pin 24V
- Max input current into fan power supply pin 5A
- Voltage into GPIO pins 5V
- Voltage output by GPIO pins 5V
- Max source/sink current of GPIO pins 20mA
- Max voltage input on fan tachometer pins 5V
- All logic i/o on board is TTL level (0/5V dc)

6 Revision History

Data	Changes
08th June 2009	First Issue A for firmware V1.0.0