Alpine AiNet

Physical layer.

The Alpine AiNet physical layer is very similar to the SAE J1850 PWM bus standard.

It is a differential bus with lines named Ainet1 and Ainet2.

Ainet1 is biased to +2.0V with reference to ground , it has positive signal pulses of between 1.5V and 2.5V.

Ainet2 is biased to +3.0V with reference to ground, it has negative signal pulses of between 1.5V and 2.5V.

PWM data is transferred over the bus as a RZ (return to zero) signal at a speed of 41.7Khz.

A logic “0” is transmitted as a 16us pulse and a logic “1” is transmitted as a 8us pulse. The data bit width is 24us.

A data packet consists of 10 bytes of data plus 1 byte as a CRC check. A packet starts with a start of frame pulse (SOF) of 32us then a 16us gap to the first character in the packet. After most packets there is a gap of approx 40us before a single acknowledgement byte is sent by the receiving device. This byte is the same as the first byte in the packet just received. Packets are sent about once every second.

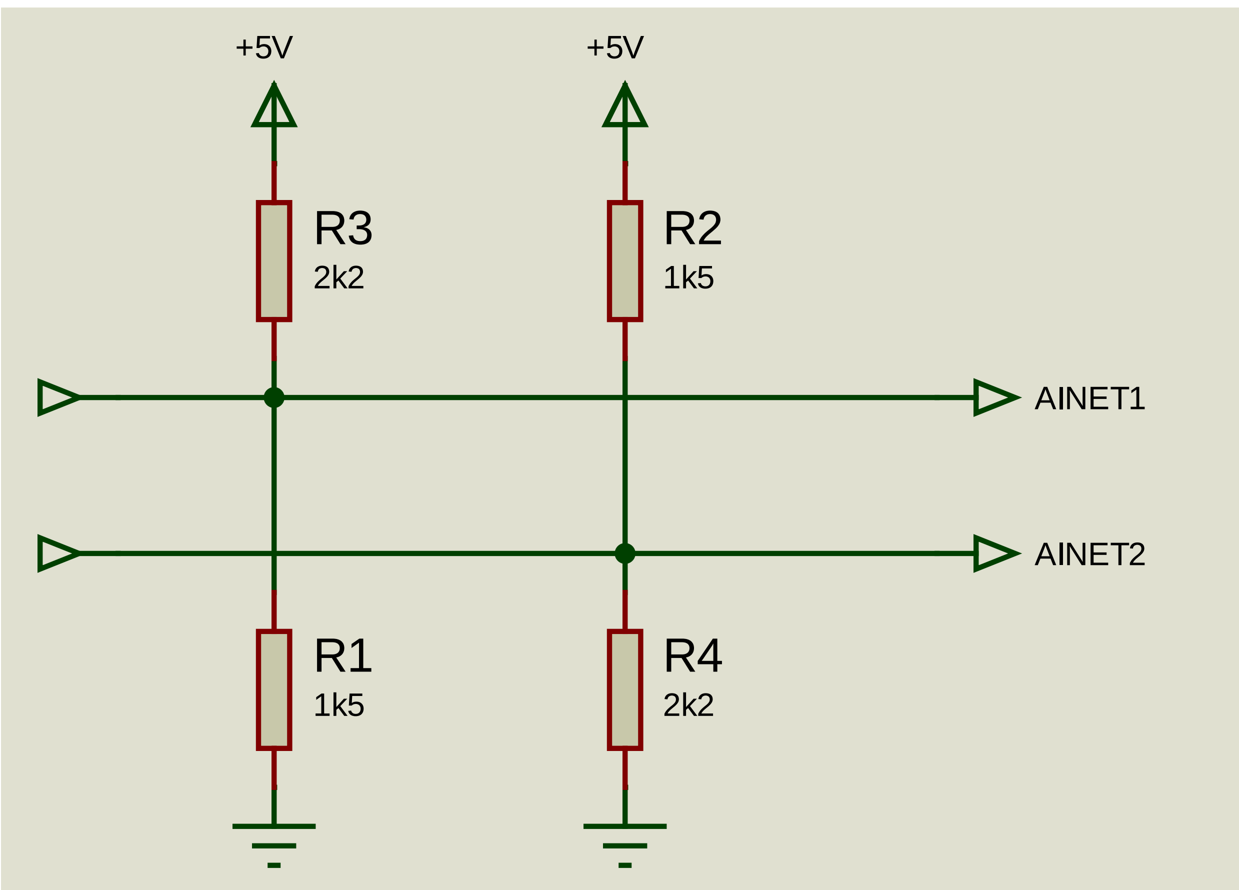

Fig 1. AiNet bus termination.

Fig.1 Shows how the Ainet bus is terminated at each device.

Fig.2 Shows a circuit capable of monitoring the bus.

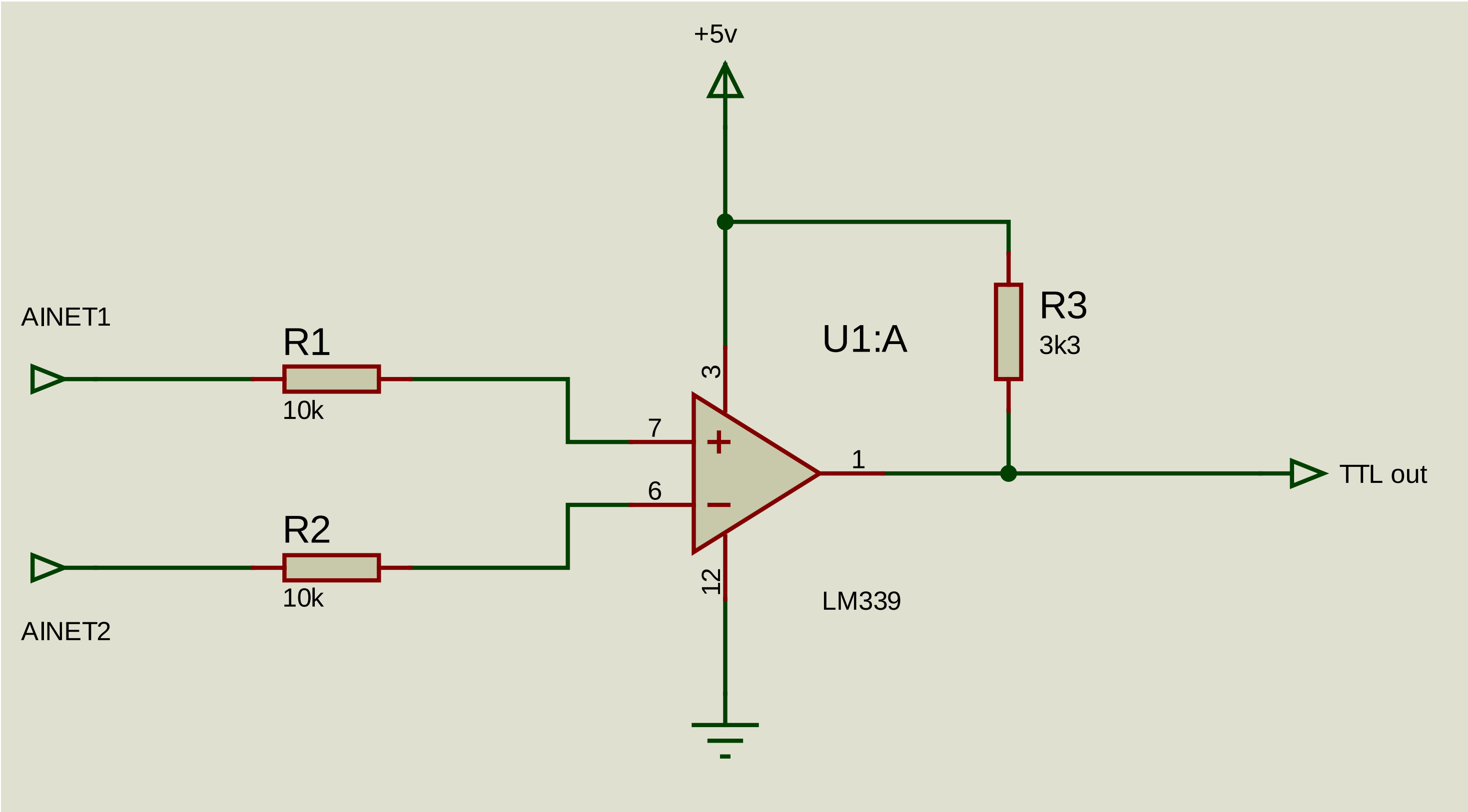

Fig.2 Bus monitor interface.

Fig 2 shows a simple interface circuit that can be used to monitor the bus. The TTL output can be taken to the circuit in Fig 3 to obtain a NRZ data and CLOCK output or could be taken directly to a microcontroller input and use firmware to decode the data.

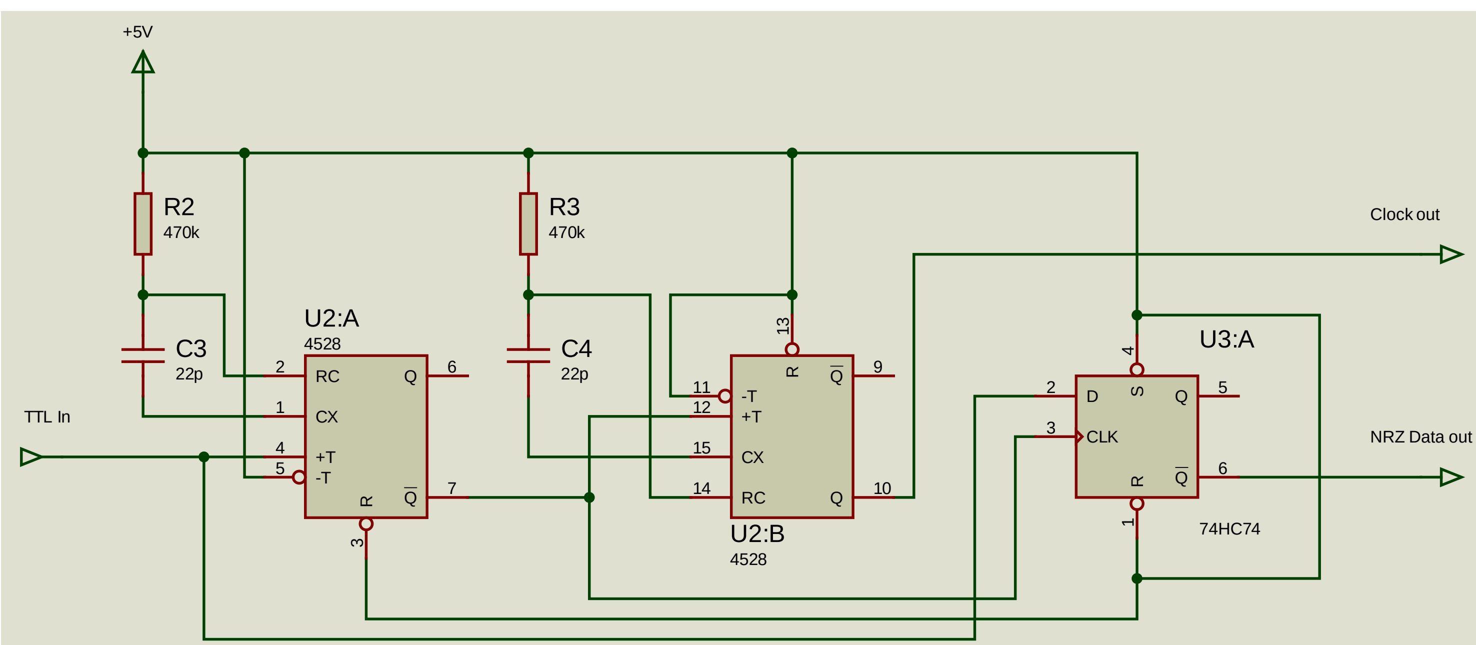

Fig .3 Bit decoder.

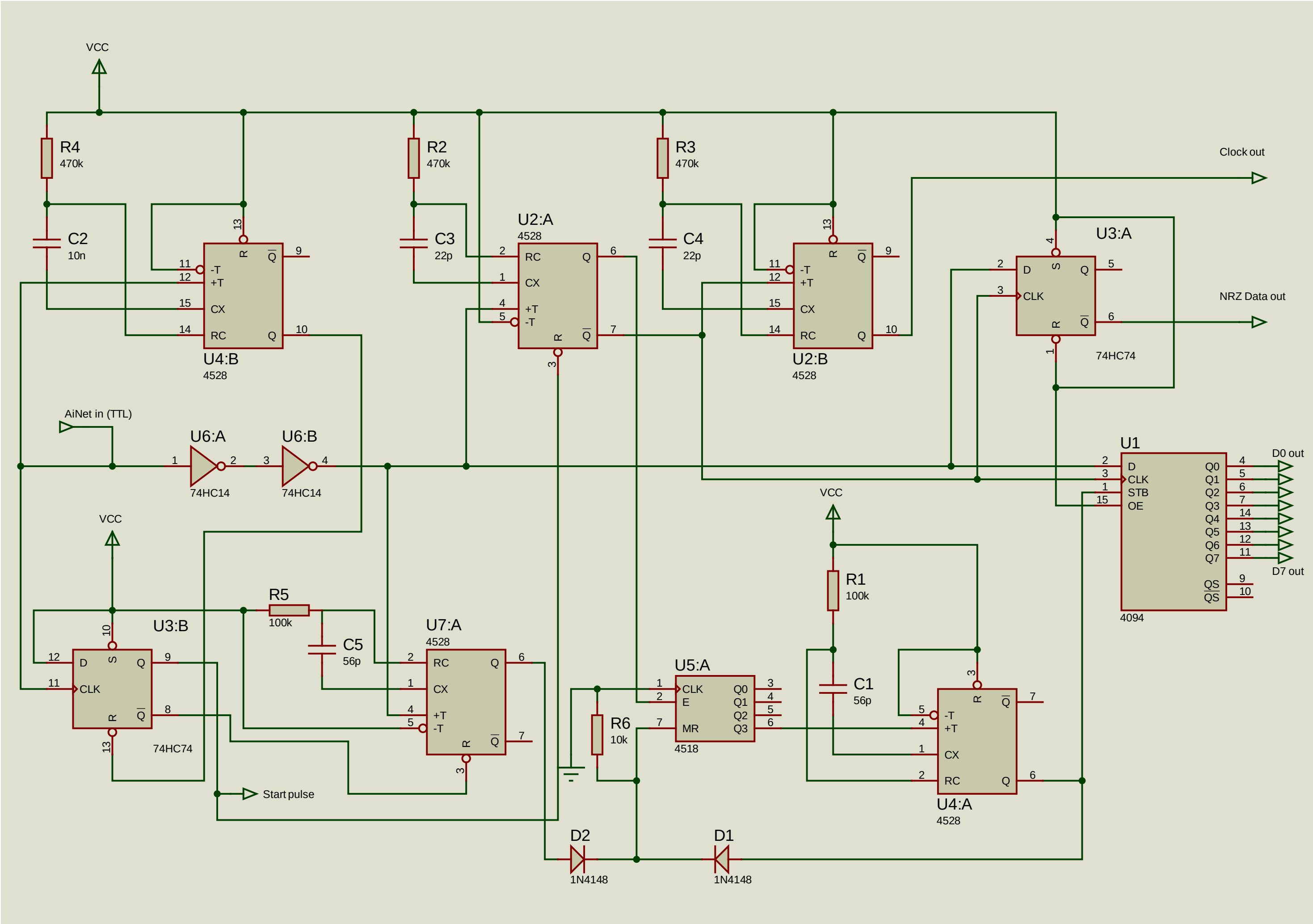

Fig. 4 Bit decoder and serial to parallel converter circuit.

I used the circuit of Fig.4 to convert the TTL RZ data from Fig.2 to 8 bit parallel data and applying this to an 8bit input port on a PIC16F877, the PIC then sent it out via RS232 to a pc where I could monitor it in a terminal program.

I did it this way initially because code for the PIC was a lot simpler and I could start monitoring very quickly, plus the fact that the PIC was not fast enough to decode the data even with a 20MHZ clock. Not with my code anyway, so I did it the easy way in hardware. I have since used a PIC 18F4550 at 48Mhz. More on that later.

The circuit I used to interface to the bus for reading and writing is shown in Fig. 5. It drives the bus very close to the way the Alpine CDC does, when used in place of the CD Changer. I have not tried driving the bus with this circuit with both the HU and CDC connected. I have no need to and I am not sure it is good idea at this stage. It is not necessary to use the LM339 (U1A) to drive Q2, it can be driven by a BC547, I did this on the first version of the circuit where I was using an MC3486 to read the bus, after finding the LM 339 worked ok I had a spare comparator so used it to drive Q2. This circuit is very similar to that used in some Alpine head units.

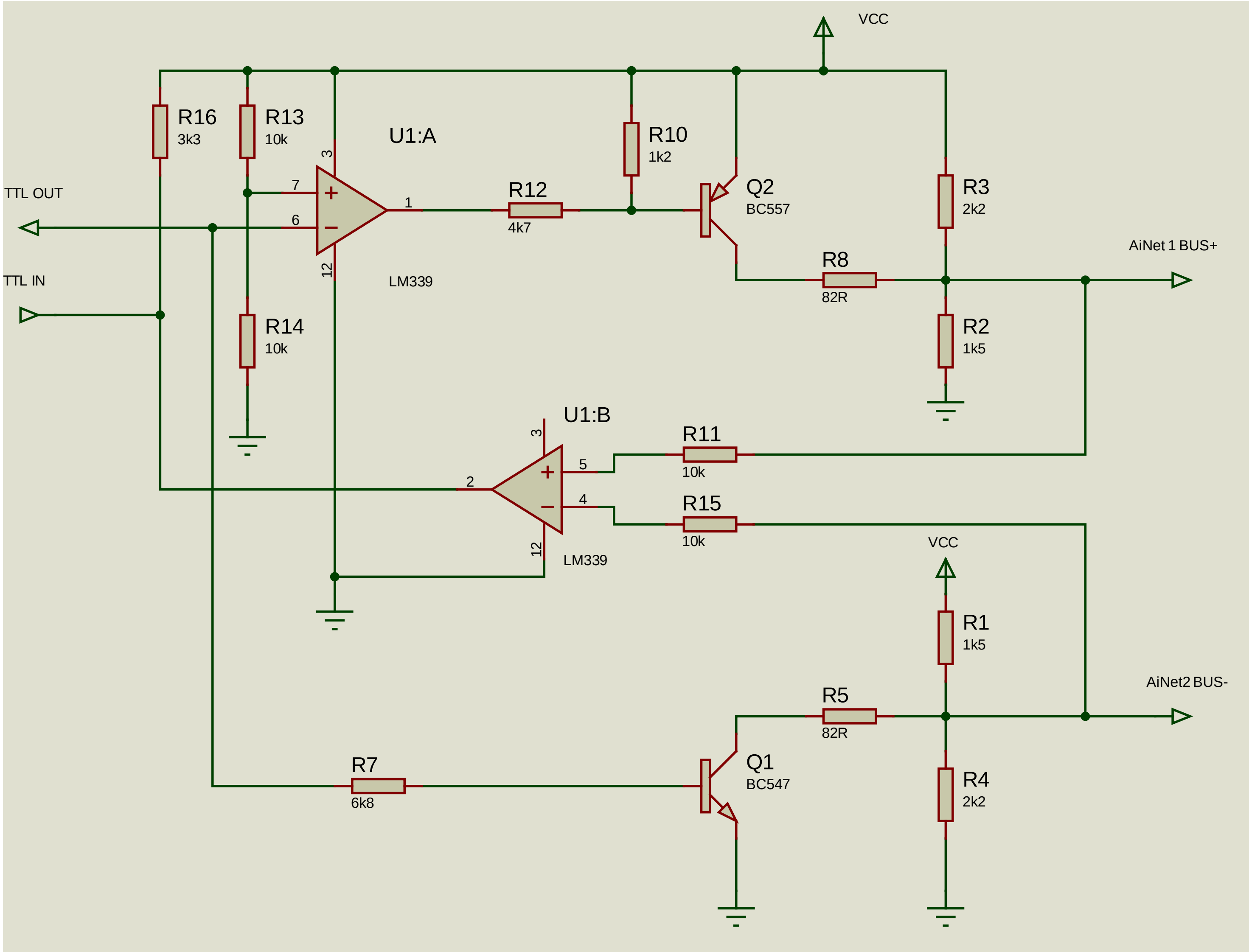

Fig. 5 Complete Ainet bus interface circuit.

AiNet Protocol.

I have not fully decoded the Ainet protocol or command structure by any means, however I will detail what I have found so far.

As stated earlier each packet on the bus consists of 11 bytes, 10 bytes of control and data information plus one a byte CRC check.

Some packets are acknowledged by the receiving unit, by retransmitting the fist byte of the packet just received. This byte is transmitted approx. 32us after the last byte of the packet, during this 32us period the bus is held low. I have not determined exactly which packets are acknowledged and which are not as yet. It appears as though this gap varies in width depending on weather the last bit in the packet is a “1” or a “0”. I have to investigate this more this variation in width may be due to my hardware setup.

Packet structure.

The CRC byte uses the same algorithm as the SAE J1850 standard. The polynomial used is $P(X) = X^8 + X^4 + X^3 + X^2 + 1$.

An internet search will provide much information on the subject, including sample code to perform the calculation.

I used a modified version of the code below, which can be found at www.obddiagnostics.com/obdinfo/crc.txt

```
// Define Global variables

// message buffer variables

unsigned char msg_buf[20], nbytes, bit_point;
unsigned char * byte_point;

// Switch to select VPW PWM or ISO  0, 1, 2,  respectively

unsigned char protocol;

// values to define protocol variable

#define vpw     0
#define pwm     1
#define   iso   2

/*****************************************************************
CRC CALCULATION SUBROUTINE

Calculates the crc as defined by SAE, or the checksum for ISO
*****************************************************************/
// This routine assumes that all the data bytes are in the array msg_buf[]
//   Starting with msg_buf[0] as the first byte.
// nbytes indicates how many bytes are in the array.
// The subroutine calculates both the checksum and the CRC byte as
//  defined by SAE and ISO specifications.
// Either the CRC or the Checksum is returned, depending on what
//  protocol is selected.

unsigned char crc(void)
{
        unsigned char crc_reg=0xff,poly,i,j, checksum=0;
        for (i=0, byte_point=msg_buf; i<nbytes; ++i, ++byte_point)
        {
                for (j=0, bit_point=0x80 ; j<8; ++j, bit_point>>=1)
                {
                        if (bit_point & *byte_point)            // case for new bit =1
                        {
                                if (crc_reg & 0x80) poly=1;     // define the polynomial
                                else poly=0x1c;
                                crc_reg= ( (crc_reg << 1) | 1) ^ poly;
                        }
                        else                                    // case for new bit =0
                        {
                                poly=0;
                                if (crc_reg & 0x80) poly=0x1d;
                                crc_reg= (crc_reg << 1) ^ poly;
                        }
```

```
                }

                checksum += *byte_point;                    // Calculate checksum
        }
        if (protocol==iso) return checksum;                 // Iso uses checksum,
        return ~crc_reg;                                    // Otherwise, use CRC
}
```

I will present some packets I have observed and note what I know about them (which is not very much at this stage).

Some command packets I have logged. All sent from The Jaguar HU.

NOTE: The Alpine commands have BIT 8 of BYTE 1 and 2 toggled, and the CRC byte changes.

PLAY CD.
B2 D0 D3 70 00 00 00 00 00 00 DF

MIX ON
B2 D0 E3 70 00 00 00 00 00 00 69

MIX OFF
B2 D0 E3 60 00 00 00 00 00 00 D0

REPEAT ON
B2 D0 E0 70 00 00 00 00 00 00 88

REPEAT OFF
B2 D0 E0 60 00 00 00 00 00 00 31

FORWARD ONE TRACK
B2 D0 D5 75 00 00 00 00 00 00 34

BACK ONE TRACK
B2 D0 D5 65 00 00 00 00 00 00 8D

FORWARD ONE DISC
B2 D0 D2 73 00 00 00 00 00 00 67
B2 D0 D3 70 00 00 00 00 00 00 DF

BACK ONE DISC
B2 D0 D2 63 00 00 00 00 00 00 DE
B2 D0 D3 70 00 00 00 00 00 00 DF

CHANGE TO DISC x (x= disc number)
B2 D0 D2 2x 00 00 00 00 00 00 crc e.g. disc 2 = B2 D0 D2 22 00 00 00 00 00 00 5D
B2 D0 D3 70 00 00 00 00 00 00 DF

The following packet was sent from CDC to HU while playing D2 T2. NOTE: Alpine not Jaguar.

7F 32 51 02 02 01 02 07 04 29 5E   From CDC, HU responds with 7F.

Byte 1
Byte 2
Byte 3
Byte4   --------Disc number
Byte 5 ---------Track number
Byte 6 ----------01 while playing disc, 00 while changing disc.
Byte 7 ----------Total playing time Mins.
Byte 8 ----------Total playing time Secs.
Byte 9 ----------Track playing time Mins.
Byte 10 ---------Track playing time Secs.
Byte 11 ---------CRC byte

Byte 12 ---------Response from HU after 32- 40us delay at end of packet.

Here is a log of the packets between HU and CDC (Jaguar) at initial power on i.e connection of power, not switch on.

```
FFFFBF2C000000000000E0 FF   From HU, CDC responds FF
FFFFBF2C000000000000E0 FF       "     "      "     "

B2B22110408000000000079          From CDC No responce

FF822000000000000000EA FF   From HU, CDC responds FF

FFB22000000000000000C8 FF   From CDC, HU responds FF

A2825000000000000000004D   --------
A2825000000000000000004D          |
8A825000000000000000020          |
8A825000000000000000020          |
B0825000000000000000C1          |
B0825000000000000000C1          |
B2825000000000000000026 B2        |--From hu, all sent twice
B4825000000000000000012          |  but one, only
B4825000000000000000012          |  one responce
B6825000000000000000F5          |
B6825000000000000000F5          |
C0825000000000000000CD          |
C0825000000000000000CD   --------

82B2906750000000000020 82   From CDC, HU responce 82

B2821090672005000000C5 B2   From HU, CDC responce B2

FFB2220100210000000080 FF ------|
FFB2422020102020202026 FF       |
FFB242202011202020204C FF       |--All from CDC, HU
FFB250260B20000000009D FF       |   responce, all FF
```

```
FFB2510100010000000048 FF       |
FFB25301400000000000CC FF ------|

B2D09067402100000000AC B2         From HU, responce B2

D0B2109067200500000067 D0-------| From CDC, HU responce D0
D0B24021019600000000C5 D0 ------|

B2D0906745000000000044 B2  From HU, CDC responce B2

D0B2109067200500000067 D0  From CDC, HU responce D0

B2D090675200000000009F B2  From HU, CDC responce B2

D0B245200000000000004C D0  From CDC, HU responce D0

B2D0906754000000000016 B2  From HU, CDC responce B2

D0B2109067200500000067 D0  From CDC, HU responce D0

B2825000000000000000026 B2  From HU, CDC responce B2

D0B25400000000000000E5 D0  From CDC, HU responce D0

B2D09067402000000000C6 B2  From HU, CDC responce B2

D0B2109067200500000067 D0  From CDC, HU responce D0

B2D090675200000000009F B2  From HU, CDC responce B2

D0B2402026003700000048 D0  From CDC, HU responce D0

B2D0D370000000000000DF B2  From HU, CDC responce B2

D0B210D370200000000056 D0  From CDC, HU responce D0

B2D090675200000000009F B2  From HU, CDC responce B2

D0B2109067200500000067 D0 -----|
FFB250260910000000003E FF      |--From CDC, HU responds
D0B252000000000000003A D0 -----|   with first byte

C082A0600000000000028     -----|
C082A0600000000000028          |
C082D06000000000000C0          |--from HU, no responce
C082D06000000000000C0          |
C082D66000000000000001F        |
C082D66000000000000001F   ------|
FF8252A000000000000072 FF------|--From HU, CDC Responce FF
FF825000000000000000002 FF------|

FFB24321D7012269275932 FF -----|
FFB2502708100000000025 FF      |
FFB2510101010000000234 FF      |
FFB25200D000000000007A FF      |
FFB25301D70122692759C4 FF      |--From CDC, HU responds FF
FFB2422020102020202026 FF      |
FFB242202011202020204C FF      |
```

```
FFB2422021102020202060 FF      |
FFB242202111202020200A FF -----|
```

To Be continued. I will update this document as soon as I have more info.

Last update 21/01/2009 at 0851.

Note: All the information presented here has been obtained by monitoring the AiNet bus. The equipment monitored consisted of a Jaguar XJ8 (2000MY) Head unit and 6 stack CDC and an Alpine CDA 7839E head unit with Alpine CHA1214 12 stack CDC.