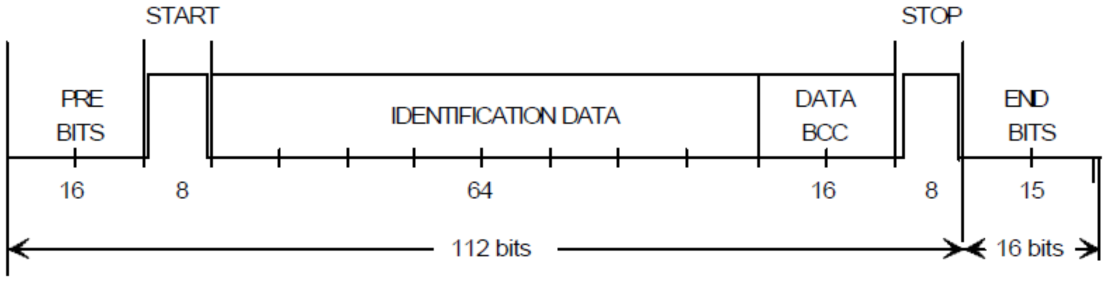
Digital Kaos

4C Transponder Information

The 4C transponder ID can be broke down into 5 parts;

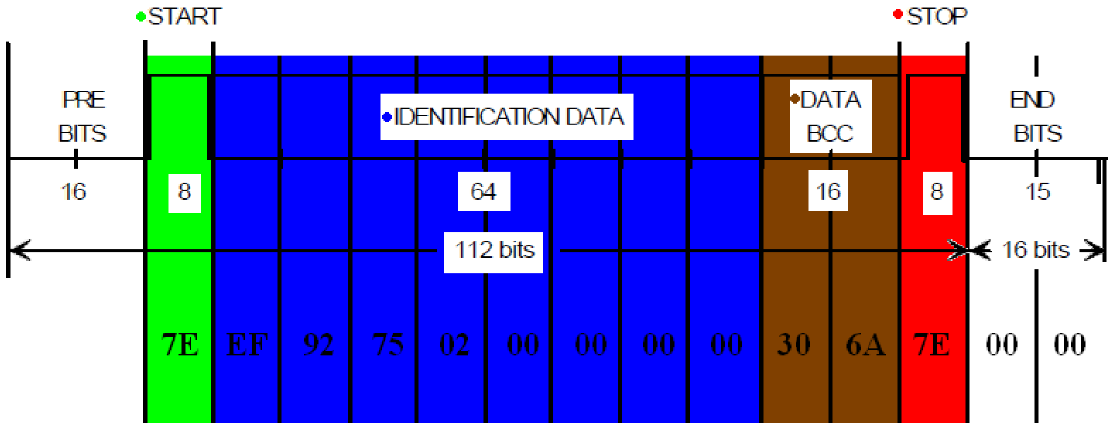
• Start bit	(1 byte)
• Data	(8 bytes)
• Checksum	(2 byte)
• Stop bit	(1 byte)
• End bits	(2 bytes)

As seen here;



Example 4C ID in TIRIS format (Texas Instruments Registration and Identification System)

7E EF 92 75 02 00 00 00 00 30 6A 7E 00 00

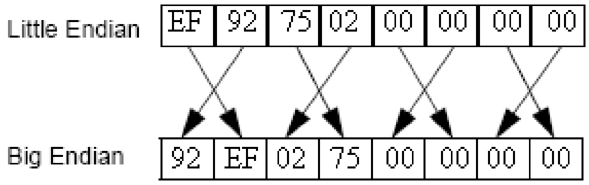


- The start and stop bit are always 7E.
- Only the first 4 bytes of the data are used the remainder being ‘00’.
- The checksum of the TIRIS format uses CRC-CCITT (Kermit) (0x5EF3) and is calculated from the **8 bytes** of Data.
- The End bits are always “00 00”

Getting Key Data From EPROM Files

Only the 8 data bytes are stored in eprom and are often repeated multiple times.

Also the data is often byte swopped;

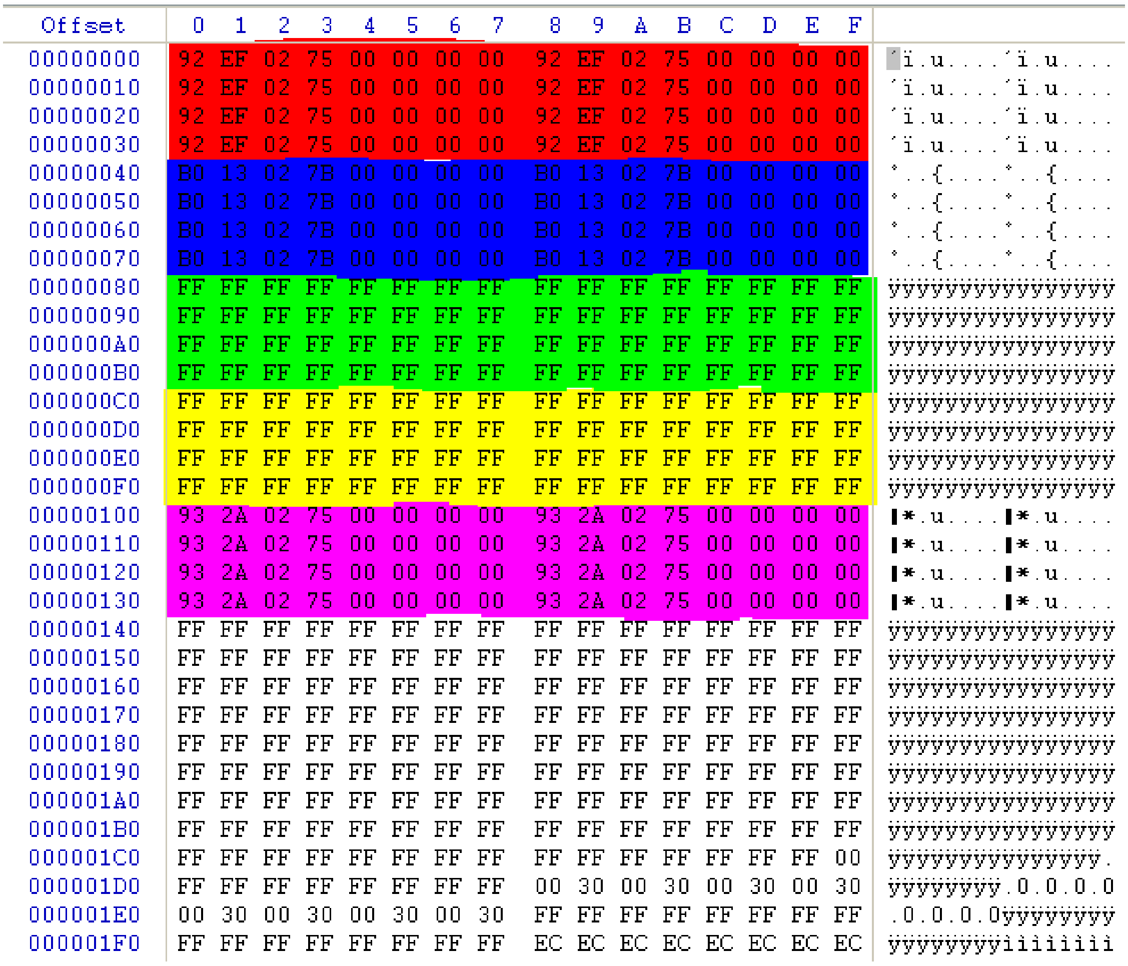


Using the above example with the start/stop/checksum/end bytes all removed;

EF 92 75 02 00 00 00 00

Will often be in the eprom as;

92 EF 02 75 00 00 00 00



```
Offset      0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000   92 EF 02 75 00 00 00 00  92 EF 02 75 00 00 00 00   ´ï.u....´ï.u....
00000010   92 EF 02 75 00 00 00 00  92 EF 02 75 00 00 00 00   ´ï.u....´ï.u....
00000020   92 EF 02 75 00 00 00 00  92 EF 02 75 00 00 00 00   ´ï.u....´ï.u....
00000030   92 EF 02 75 00 00 00 00  92 EF 02 75 00 00 00 00   ´ï.u....´ï.u....
00000040   B0 13 02 7B 00 00 00 00  B0 13 02 7B 00 00 00 00   °..{....°..{....
00000050   B0 13 02 7B 00 00 00 00  B0 13 02 7B 00 00 00 00   °..{....°..{....
00000060   B0 13 02 7B 00 00 00 00  B0 13 02 7B 00 00 00 00   °..{....°..{....
00000070   B0 13 02 7B 00 00 00 00  B0 13 02 7B 00 00 00 00   °..{....°..{....
00000080   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000090   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000A0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000B0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000C0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000D0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000E0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000000F0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000100   93 2A 02 75 00 00 00 00  93 2A 02 75 00 00 00 00   ▌*.u....▌*.u....
00000110   93 2A 02 75 00 00 00 00  93 2A 02 75 00 00 00 00   ▌*.u....▌*.u....
00000120   93 2A 02 75 00 00 00 00  93 2A 02 75 00 00 00 00   ▌*.u....▌*.u....
00000130   93 2A 02 75 00 00 00 00  93 2A 02 75 00 00 00 00   ▌*.u....▌*.u....
00000140   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000150   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000160   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000170   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000180   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
00000190   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001A0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001B0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001C0   FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF 00   ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ.
000001D0   FF FF FF FF FF FF FF FF  00 30 00 30 00 30 00 30   ÿÿÿÿÿÿÿÿ.0.0.0.0
000001E0   00 30 00 30 00 30 00 30  FF FF FF FF FF FF FF FF   .0.0.0.0ÿÿÿÿÿÿÿÿ
000001F0   FF FF FF FF FF FF FF FF  EC EC EC EC EC EC EC EC   ÿÿÿÿÿÿÿÿìììììììì
```

In the above picture we can see the example data repeated 8 times in **red**.

Highlighted in **blue** and **purple** are two more keys, with two unused keys in **green** and **yellow** and the possibility of a couple of unused keys after that.

Your eprom may have a different number of repeats and different locations, you are looking for 4 bytes of data followed by 4 bytes of “00 00 00 00” probably repeated.

If you have a known key you can isolate the key data and try to find the bytes (you may have to byte swop) to find the area where keys are stored, then you will have to figure out the pattern and alter as required.

Keys can be removed (‘FF’’s), added or changed.

TIRIS Format to TPX1/Electronic Head Format

- The TXP1 format has the same start/stop/end bytes as TIRIS.

- The checksum is calculated with CRC-CCITT (0x0000) which is the same CRC routine but has a starting value of 0x0000 as apposed to TIRIS 0x5EF3.

The Data for TPX1 format can be worked out from the TIRIS format.

Take the TIRIS format data : EF 92 75 02 00 00 00 00

Now we need to change each byte into binary (can use window calc in scientific mode)

Type the byte in hex mode “EF” now click bin -> 11101111

NOTE: Windows calc removes leading 0’s we need them so if there isn’t 8 bits pad the number with 0’s at the START to make 8 bits.

Now take 11101111 and reverse the bits to get 11110111 convert this back to hex -> F7

This is our first byte in TPX1 format. Do the same for each byte you will get F7 49 AE 40 00 00 00 00

You can now use this along with a CRC calculator such as http://www.lammertbies.nl/comm/info/crc-calculation.html

To get the CRC of “0C 56” CRC-CCITT (0x0000)(XModem)

Add this to the data along with the start/stop/end bits to get the TPX1 data;
7E F7 49 AE 40 00 00 00 00 0C 56 7E 00 00