C7042 Scalextric

6 Car Power Base

C7042 Scalextric 6 Car Power Base SNC Communication Protocol

This document is prepared for Hornby plc

1 September 2009

Ref HL02

Document Approval

Document Number: **HO014-C7042**

Version: **01**

Date: **01/09/2009**

Total pages: **17**

Author:	**Trevor Tam**	**Date: 01/09/2009**
Reviewer:	**Sunny Chan/Kenny Mok**	**Date: 01/09/2009**
Approver:	**Scott Grubb**	**Date: 01/09/2009**

SAGENTIA

Contents

1 Introduction

This document describes the communication protocol between the C7042 - 6 Car Power Base in the Screen Not Connected mode (SNC mode) and host (it is such as PC).

Note that the protocol is designed with the specific hardware implementation taken into account and familiarity with this is assumed.

2 SNC Protocol

The SNC protocol provides PC or other communication devices with platform to communication with a 6 Car Power Base (6CPB) unit.

When the 6CPB entered SNC mode (i.e. the screen is not connected to 6CPB), communication using SNC protocol with an external device is allowed.
In this situation, an external device can:

1. Read the handsets status
2. Read the game timing
3. Read the track status (whether it is powered or not)
4. Read the Auxiliary Port Current
5. Conduct the game play by driving cars and the LEDs on 6CPB directly using the information it has collected.

The communication model is based on the serial communication standard, Universal Asynchronous Receiver/Transmitter (UART), using half duplex transmission.

The protocol is suitable for general application under standard communication such as EIA RS-485-Standard, RF-wireless and Infrared.

The UART configurations are:
1 start bit (0),
8 data bits,
1 stop bit (1),
No parity bit
Baudrate: 19,200bps

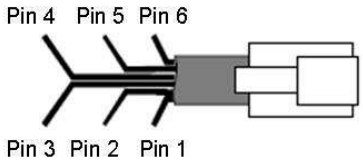
The protocol consists of 2 types of packet, incoming packet to 6CPB and outgoing packet from 6CPB.

When the 6CPB receives a valid incoming packet (defined in 2.3), it will reply the extern device with the outgoing packet (defined in 2.2). The communication cycle can be repeated after the 6CPB has finished its packet transmission.

All packets, either incoming packet or outgoing packet, are secured by an ***Error Detection Byte*** in its last byte. An ***Error Detection Byte*** is formed using 8-bit CRC polynomial x8+x2+x+1 of all preceding bits stream (see 2.5). If the ***Error Detection Byte*** received is different from the CRC calculated, the packet should be discarded.

2.1 RS485 Hardware connection

The RS485 communication uses 4 wires to interconnect the devices to an external device.
The configuration is described as below:

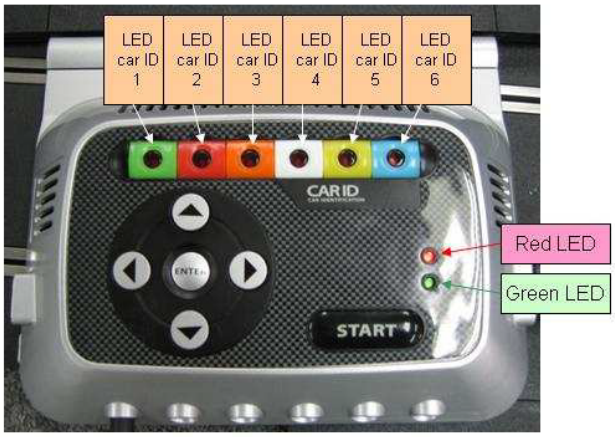


Pin 1	No used
Pin 2	Ground
Pin 3	RS-485 –V**(**Receive/Transmit inverting)
Pin 4	RS-485 +V**(**Receive/Transmit not inverting**)**
Pin 5	12 Volts
Pin 6	No used

2.2 Incoming Packet

The incoming packet carries the information for:

1. Driving signal for each car.
2. Driving signal for each LED on 6CPB unit.



2.2.1 Packet Structure

The packet from PC to 6CPB consists of 9 bytes of data.
The structure of the packet is as follows:

Byte	Item
1st	Operation Mode
2nd	Drive Packet #1
3rd	Drive Packet #2
4th	Drive Packet #3
5th	Drive Packet #4
6th	Drive Packet #5
7th	Drive Packet #6
8th	LED status
9th	Checksum

2.2.2 Operation Mode (1st Byte)

This 1 byte acts as the indictor to tell 6CPB whether it has successfully received the previous packet from 6CPB or not.

At the beginning of the communication (i.e. no previous packet from 6CPB), the 1st byte should be sent as 0xFF. Every success in receiving the last packet should also be feed backed to 6CPB by sending the 1st byte as 0xFF.

In case the host cannot recognize the previous packet from 6CPB, the 1st byte should be sent as 0x7F. 6CPB will re-send the same packet that PC cannot recognize for at most 2 times in this situation.

2.2.3 Drive Packet #1 - #6 (2nd Byte - 7th Byte)

Each byte carries the required driving signal for its corresponding car. These bytes are identical in format, and have to be sent in ones complement.(i.e. they are sent with changing all the bits that are 1 to 0 and all the bits that are 0 to 1)

Format **before ones complement** (same as 2.3.2):

Bit	bit 7	bit 6	bit 5 – bit 0
Item	Brake	Lane-Change	Power
Description	A value of "1" in this bit represents brake is applied to the car	A value of "1" in this bit represents the request of Lane Change operation to the car.	The six bits together contain the required power level (from 0 to 63) to be delivered to the car. The value of 63 indicates the maximum power level required while the value of 0 indicates the minimum power level.

2.2.4 LED status (8th Byte)

This byte carries the information for driving the LEDs on 6CPB.

Format:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Green LED	Red LED	LED 6	LED 5	LED 4	LED 3	LED 2	LED 1

A value of "1" in each bit indicates the corresponding LED is in "On" Status.

The status of Green and Red LED together carry the status of the Game Timer as describe as follow:

Game Timer Status	Green LED status	Red LED status
Not affected	0	0
Not affected	0	1
Started	1	0
Stopped/Reset	1	1

Whenever the 6CPB has received the LED status command: Game Timer Stopped/Reset (i.e. 8th Byte - bit7=1; bit6=1) the data of Game Timer and the data of SF-timer for all cars will be reset to their default value.

2.3 Outgoing Packet

The incoming packet carries the information of:

1. Current status of each handset;
2. Auxiliary port current consumed;
3. Game Timer information;
4. Car information updated, which includes car ID and passing SF-line time.

The packet from 6CPB to PC consists of 14 bytes of data.
The structure of the packet is as follows:

byte	Item
1st	Handset+Track Status
2nd	Handset #1
3rd	Handset #2
4th	Handset #3
5th	Handset #4
6th	Handset #5
7th	Handset #6
8th	Aux port current
9th	CarID / Track # updated
10th	Game or SF-line time(LSB)
11th	Game or SF-line time
12th	Game or SF-line time
13th	Game or SF-line time(MSB)
14th	Checksum

2.3.1 Handset+Track Status (1st Byte)

This byte contains the current status of the Handsets and the power delivered to the track.

Format :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
1	Handset # 6	Handset # 5	Handset # 4	Handset # 3	Handset # 2	Handset # 1	Track Power Status

Bit 7 is always equal to “1”.

Bit 6 to bit 1 represent the connection status of the handsets. Value of “1” indicates the corresponding Handset is connected properly.

Bit 0 indicates the power status in the track. Value of “1” represents power is delivered to the track properly while “0” means the track power is cut off.

2.3.2 Handset #1 - #6 (2nd Byte – 7th Byte)

Each byte carries the status of the corresponding handset, which also indicates different control requests, such as driving power, braking and lane change operation to the corresponding car by the user. These bytes are identical in format , and have to be sent in ones complement.(i.e. they are sent with changing all the bits that are 1 to 0 and all the bits that are 0 to 1)

Format **before ones complement** (same as 2.2.3):

Bit	bit 7	bit 6	bit 5 – bit 0
Item	Brake	Lane-Change	Power
Description	A value of “1” in this bit represents brake is applied to the car	A value of “1” in this bit represents the request of Lane Change operation to the car.	The six bits together contain the required power level (from 0 to 63) to be delivered to the car. The value of 63 indicates the maximum power level required while the value of 0 indicates the minimum power level.

2.3.3 Aux port current (8th Byte)

This byte contains the value from 0-255, which indicates the auxiliary port current (in mA) consumed.

2.3.4 CarID / Track # updated (9th Byte)

This byte carried the ID of the car which has first passed the SF-Line.

Format :

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2- bit 0
1	1	1	1	1	Car ID

Bit 7 to bit 3 are always equal to ” 1”

Bit 2 to bit 0 carries the Car ID. The corresponding information is to be reported in section 2.3.5 and 2.3.6

The value of “000’ in Bit2-Bit0 represents the Game-Timer, and the value of “111” should be regarded as invalid ID.

2.3.5 Game or SF-line time (10th Byte - 13th Byte)

These 4 bytes data contain the 32-bit timer/counter value (relative to the game starting moment) when the car (with the ID specified in 2.3.4) passing the SF line.

Every 6.4µ sec will cause an increment to the count er. Hence, the actual passing time can be calculated by

6.4 µ sec X 32-bit counter value.

The value 0xFFFFFFFF will be transmitted in case the game is not started or the car ID is invalid.

2.4 Communication performance/Sampling

For a complete communication, 6CPB would receive 9bytes incoming packet, and then reply with 14 bytes outgoing packet.

Number of byte per communication cycle= 9+14= 23 bytes

Number of bits byte per communication cycle= 23 * (8 bits data +1 stop bit +1 start bits) =230 bits

Max number of communication cycle per second
= Baud rate/ Total Packet bits
=19,200/230
= 83.5 cycles/sec

2.5 CRC Checksum

2.5.1 Sample Code for standard 8-bit CRC in Visual Basic

```
Dim sRs232Buffer() As Byte = New Byte(1023) {}
Dim crc8 As Byte
Dim CRC8_LOOKUP_TABLE() As Byte = New Byte(255)
{ _
  &H0, &H7, &HE, &H9, &H1C, &H1B, &H12, &H15, &H38, &H3F, &H36, &H31, &H24, &H23, &H2A, &H2D, _
  &H70, &H77, &H7E, &H79, &H6C, &H6B, &H62, &H65, &H48, &H4F, &H46, &H41, &H54, &H53, &H5A, &H5D, _
  &HE0, &HE7, &HEE, &HE9, &HFC, &HFB, &HF2, &HF5, &HD8, &HDF, &HD6, &HD1, &HC4, &HC3, &HCA, &HCD, _
  &H90, &H97, &H9E, &H99, &H8C, &H8B, &H82, &H85, &HA8, &HAF, &HA6, &HA1, &HB4, &HB3, &HBA, &HBD, _
  &HC7, &HC0, &HC9, &HCE, &HDB, &HDC, &HD5, &HD2, &HFF, &HF8, &HF1, &HF6, &HE3, &HE4, &HED, &HEA, _
  &HB7, &HB0, &HB9, &HBE, &HAB, &HAC, &HA5, &HA2, &H8F, &H88, &H81, &H86, &H93, &H94, &H9D, &H9A, _
  &H27, &H20, &H29, &H2E, &H3B, &H3C, &H35, &H32, &H1F, &H18, &H11, &H16, &H3, &H4, &HD, &HA, _
  &H57, &H50, &H59, &H5E, &H4B, &H4C, &H45, &H42, &H6F, &H68, &H61, &H66, &H73, &H74, &H7D, &H7A, _
  &H89, &H8E, &H87, &H80, &H95, &H92, &H9B, &H9C, &HB1, &HB6, &HBF, &HB8, &HAD, &HAA, &HA3, &HA4, _
  &HF9, &HFE, &HF7, &HF0, &HE5, &HE2, &HEB, &HEC, &HC1, &HC6, &HCF, &HC8, &HDD, &HDA, &HD3, &HD4, _
  &H69, &H6E, &H67, &H60, &H75, &H72, &H7B, &H7C, &H51, &H56, &H5F, &H58, &H4D, &H4A, &H43, &H44, _
  &H19, &H1E, &H17, &H10, &H5, &H2, &HB, &HC, &H21, &H26, &H2F, &H28, &H3D, &H3A, &H33, &H34, _
  &H4E, &H49, &H40, &H47, &H52, &H55, &H5C, &H5B, &H76, &H71, &H78, &H7F, &H6A, &H6D, &H64, &H63, _
  &H3E, &H39, &H30, &H37, &H22, &H25, &H2C, &H2B, &H6, &H1, &H8, &HF, &H1A, &H1D, &H14, &H13, _
  &HAE, &HA9, &HA0, &HA7, &HB2, &HB5, &HBC, &HBB, &H96, &H91, &H98, &H9F, &H8A, &H8D, &H84, &H83, _
  &HDE, &HD9, &HD0, &HD7, &HC2, &HC5, &HCC, &HCB, &HE6, &HE1, &HE8, &HEF, &HFA, &HFD, &HF4, &HF3
}
crc8 = CRC8_LOOKUP_TABLE(sRs232Buffer(0))
For sBufPtr = 1 To 7
  crc8 = CRC8_LOOKUP_TABLE(crc8 Xor sRs232Buffer(sBufPtr))
Next
```

2.5.2 Sample Code for standard 8-bit CRC in C

```
unsigned char crc8;
unsigned char i;
unsigned char srs[1024];
const unsigned char CRC8_LOOK_UP_TABLE[256] =
{
0x00,0x07,0x0e,0x09,0x1c,0x1b,0x12,0x15,0x38,0x3f,0x36,0x31,0x24,0x23,0x2a,0x2d,
0x70,0x77,0x7E,0x79,0x6C,0x6B,0x62,0x65,0x48,0x4F,0x46,0x41,0x54,0x53,0x5A,0x5D,
0xE0,0xE7,0xEE,0xE9,0xFC,0xFB,0xF2,0xF5,0xD8,0xDF,0xD6,0xD1,0xC4,0xC3,0xCA,0xCD,
0x90,0x97,0x9E,0x99,0x8C,0x8B,0x82,0x85,0xA8,0xAF,0xA6,0xA1,0xB4,0xB3,0xBA,0xBD,
0xC7,0xC0,0xC9,0xCE,0xDB,0xDC,0xD5,0xD2,0xFF,0xF8,0xF1,0xF6,0xE3,0xE4,0xED,0xEA,
0xB7,0xB0,0xB9,0xBE,0xAB,0xAC,0xA5,0xA2,0x8F,0x88,0x81,0x86,0x93,0x94,0x9D,0x9A,
0x27,0x20,0x29,0x2E,0x3B,0x3C,0x35,0x32,0x1F,0x18,0x11,0x16,0x03,0x04,0x0D,0x0A,
0x57,0x50,0x59,0x5E,0x4B,0x4C,0x45,0x42,0x6F,0x68,0x61,0x66,0x73,0x74,0x7D,0x7A,
0x89,0x8E,0x87,0x80,0x95,0x92,0x9B,0x9C,0xB1,0xB6,0xBF,0xB8,0xAD,0xAA,0xA3,0xA4,
0xF9,0xFE,0xF7,0xF0,0xE5,0xE2,0xEB,0xEC,0xC1,0xC6,0xCF,0xC8,0xDD,0xDA,0xD3,0xD4,
0x69,0x6E,0x67,0x60,0x75,0x72,0x7B,0x7C,0x51,0x56,0x5F,0x58,0x4D,0x4A,0x43,0x44,
0x19,0x1E,0x17,0x10,0x05,0x02,0x0B,0x0C,0x21,0x26,0x2F,0x28,0x3D,0x3A,0x33,0x34,
0x4E,0x49,0x40,0x47,0x52,0x55,0x5C,0x5B,0x76,0x71,0x78,0x7F,0x6A,0x6D,0x64,0x63,
0x3E,0x39,0x30,0x37,0x22,0x25,0x2C,0x2B,0x06,0x01,0x08,0x0F,0x1A,0x1D,0x14,0x13,
0xAE,0xA9,0xA0,0xA7,0xB2,0xB5,0xBC,0xBB,0x96,0x91,0x98,0x9F,0x8A,0x8D,0x84,0x83,
0xDE,0xD9,0xD0,0xD7,0xC2,0xC5,0xCC,0xCB,0xE6,0xE1,0xE8,0xEF,0xFA,0xFD,0xF4,0xF3
};
//Routine for the CRC
crc8= CRC8_LOOK_UP_TABLE [srs[0]  ];                    //first byte
// for incoming packet, data length is 9 byte, the routine should be should be looped for 7 times
// for outgoing packet, data length is 14 byte, the routine should be should be looped for 14 times
for (i=1;i<=7;i++)   //loop for 7 times for incoming packet
{
    crc8= CRC8_LOOK_UP_TABLE [ crc8 ^ srs[i]  ];
}
```

2.5.3 Sample Code for standard 8-bit CRC in Microsoft Visual C# 2005 Express Edition

```
byte[] from6CPB_Msg  = new byte[16];
byte crc8Rx = 0;
int i=0;

byte[] CRC8_LOOK_UP_TABLE=new byte[256]
{

0x00,0x07,0x0e,0x09,0x1c,0x1b,0x12,0x15,0x38,0x3f,0x36,0x31,0x24,0x23,0x2a,0x2d,

0x70,0x77,0x7E,0x79,0x6C,0x6B,0x62,0x65,0x48,0x4F,0x46,0x41,0x54,0x53,0x5A,0x5D,

0xE0,0xE7,0xEE,0xE9,0xFC,0xFB,0xF2,0xF5,0xD8,0xDF,0xD6,0xD1,0xC4,0xC3,0xCA,0xCD,

0x90,0x97,0x9E,0x99,0x8C,0x8B,0x82,0x85,0xA8,0xAF,0xA6,0xA1,0xB4,0xB3,0xBA,0xBD,

0xC7,0xC0,0xC9,0xCE,0xDB,0xDC,0xD5,0xD2,0xFF,0xF8,0xF1,0xF6,0xE3,0xE4,0xED,0xEA,

0xB7,0xB0,0xB9,0xBE,0xAB,0xAC,0xA5,0xA2,0x8F,0x88,0x81,0x86,0x93,0x94,0x9D,0x9A,

0x27,0x20,0x29,0x2E,0x3B,0x3C,0x35,0x32,0x1F,0x18,0x11,0x16,0x03,0x04,0x0D,0x0A,

0x57,0x50,0x59,0x5E,0x4B,0x4C,0x45,0x42,0x6F,0x68,0x61,0x66,0x73,0x74,0x7D,0x7A,

0x89,0x8E,0x87,0x80,0x95,0x92,0x9B,0x9C,0xB1,0xB6,0xBF,0xB8,0xAD,0xAA,0xA3,0xA4,

0xF9,0xFE,0xF7,0xF0,0xE5,0xE2,0xEB,0xEC,0xC1,0xC6,0xCF,0xC8,0xDD,0xDA,0xD3,0xD4,

0x69,0x6E,0x67,0x60,0x75,0x72,0x7B,0x7C,0x51,0x56,0x5F,0x58,0x4D,0x4A,0x43,0x44,

0x19,0x1E,0x17,0x10,0x05,0x02,0x0B,0x0C,0x21,0x26,0x2F,0x28,0x3D,0x3A,0x33,0x34,

0x4E,0x49,0x40,0x47,0x52,0x55,0x5C,0x5B,0x76,0x71,0x78,0x7F,0x6A,0x6D,0x64,0x63,

0x3E,0x39,0x30,0x37,0x22,0x25,0x2C,0x2B,0x06,0x01,0x08,0x0F,0x1A,0x1D,0x14,0x13,

0xAE,0xA9,0xA0,0xA7,0xB2,0xB5,0xBC,0xBB,0x96,0x91,0x98,0x9F,0x8A,0x8D,0x84,0x83,

0xDE,0xD9,0xD0,0xD7,0xC2,0xC5,0xCC,0xCB,0xE6,0xE1,0xE8,0xEF,0xFA,0xFD,0xF4,0xF3

};
```

//Routine for the CRC

```
crc8Rx = CRC8_LOOK_UP_TABLE[from6CPB_Msg[0]];        //first byte

for (i = 1; i <= 14; i++)     //loop for 14 times for incoming packet
{
      crc8Rx = CRC8_LOOK_UP_TABLE[crc8Rx ^ from6CPB_Msg[i]];

}
```

Revision History

Revision	Issue Date	Prepared by	Authorized by	Reason/Description
01	1Sept09	Trevor Tam	Scott Grubb	Initial release

Sagentia SGAI Ltd
Realising innovation

Sagentia SGAI provides design, development and manufacture of high quality products backed by a substantial and innovative technology capability in Europe and the US – we realise innovative ideas.

Sagentia SGAI is a joint venture between Sagentia and Automatic International. Automatic is a Hong Kong based Chinese manufacturer specialising in high quality manufacturing. SGAI works with both Automatic and other manufacturers within China and the Far East depending upon the product to be made.

Sagentia
Many minds make bright work

We create value from technology. We do this by developing richer solutions with our clients that enable them to realise better business opportunities.

We operate in six market sectors developing new technologies, products and services that change the basis of competition. We assist business leaders and policy makers to create strategies for technology, innovation and growth.

Our Collective Technology Wisdom – the unique characteristic of our company – guides how we work. We form highly creative teams that draw on individuality and collective experience. And we take a multi-dimensional approach to opportunity discovery and problem solving, drawing on our combined technical expertise, business acumen and industry experience.

We can work with you wherever you are in the world. Our teams are situated in state-of-the-art facilities in Cambridge UK, Frankfurt and Stockholm in Europe, Boston and Baltimore in the USA, and Hong Kong and Shanghai in China.

www.sagentia.com
info@sagentia.com

Sagentia Ltd
Harston Mill
Harston
Cambridge
CB22 7GG
UK

T. +44 1223 875200

Sagentia GmbH
Westend Carree
Grüneburgweg 18
D-60322 Frankfurt
Germany

T. +49 69 9550 4500

Sagentia Inc
Reservoir Place
1601 Trapelo Road
Suite 154
Waltham MA 02451
USA

T. +1 781 290 0500

Sagentia Inc
11403 Cronhill Drive
Suite B
Owings Mills
Baltimore MD 21117
USA

T. +1 410 654 0090

Sagentia Catella AB
Veddestavägen 7
SE-175 62 Järfälla
Sweden

T. +46 8 445 7960

Sagentia SGAI Ltd
Unit 6-7, 13/F
Wah Wai Industrial
Centre
38-40 Au Pui Wan Street
Fotan
Hong Kong

T. +852 2866 8701

SAGENTIA

many minds make bright work®