I-BUS Inside

Inside the BMW Cars entertainement Serial Bus

Written by Franck Touanen.
HackIBus@Yahoo.fr

2002, December.

Version DRAFT. REV 5.

A part of this information is based on research done by Thomas L. Wood, who has connected a Linux embedded PC in his E39 (without NAV) to play MP3 . After exchanges with Thomas, I built my own interface to start I-Bus analysis with the objective to interface a MP3 pic based player fully integrated with the board monitor.

Note that all this document is based on analysis done in a E39 (July 2000) with a MK II navigation system. I can't ensure if informations and/or device behaviours are identical for others models.

DISCLAIMER :

If you find errors or missing things, you can mail me at HackIBus@yahoo.fr.

A bus is basically a shared communication link.

It's a collection of wires through which data is transmitted from one part of a computer to another. You can think of a bus as a highway on which data travels within a computer.

In your computer, you can find e.g. a PCI bus to establish communication between extension cards and the CPU, the memory bus, the AGP bus for video, the Universal Serial Bus for high speed serial devices …

In your BMW, a bus is used to establish communication between security equipments, for engine control, or for entertainement devices (the famous IBus).

In automotive application, 2 mains bus architecture are used (others are often derivated from) :

The CAN bus and the LIN bus.

CAN Bus (Controller Area Network)

The CAN Bus was developed by Robert Bosch, GmbH. It combines extensive error checking with a high data throughput to create an extremely powerful platform for local communication. CAN has an event driven, priority based protocol which means that any device may attempt to communicate on the bus, but priority is given to the node with a greater priority “message identifier”.

Physically the CAN system is based around a single twisted pair of conductors, but can be realised on a number of mediums, including single wire and fibre optic. This configuration is quite robust and allows for 1 wire to be broken, shorted to ground or shorted to power (with reduced signal to noise).

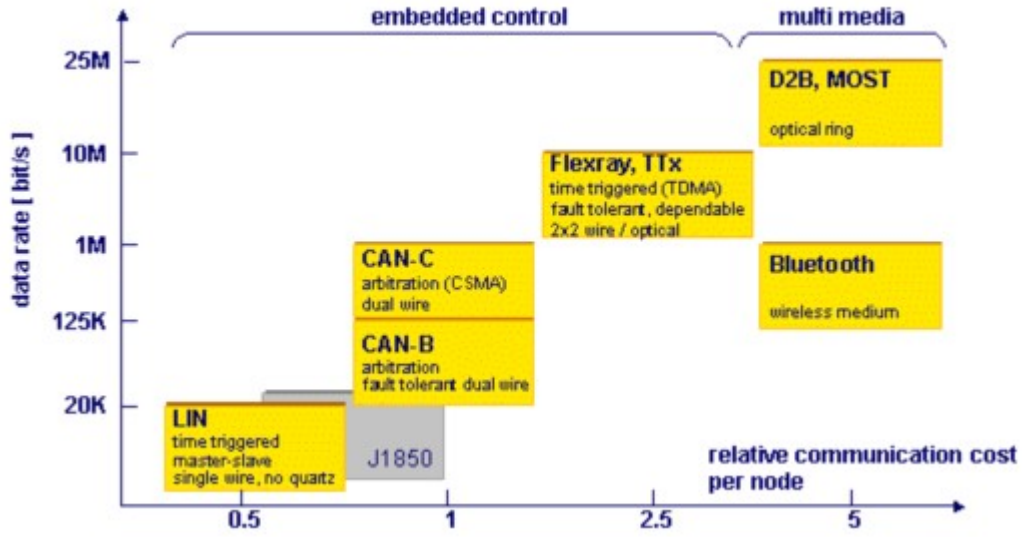
Currently CAN is very popular in Germany (it’s country of origin) and in Europe, but has not penetrated many other markets effectively. However, it is predicted that by the year 2002 US car manufacturers will be using CAN in volume.

LIN Bus (Local Interconnect Network)

LIN is a relatively new bus. It was developed through the collaboration of several companies, namely; Audi, BMW, Daimler-Chrysler, Motorola GmbH, Volcano Communications Technologies, Volvo and Volkswagen.

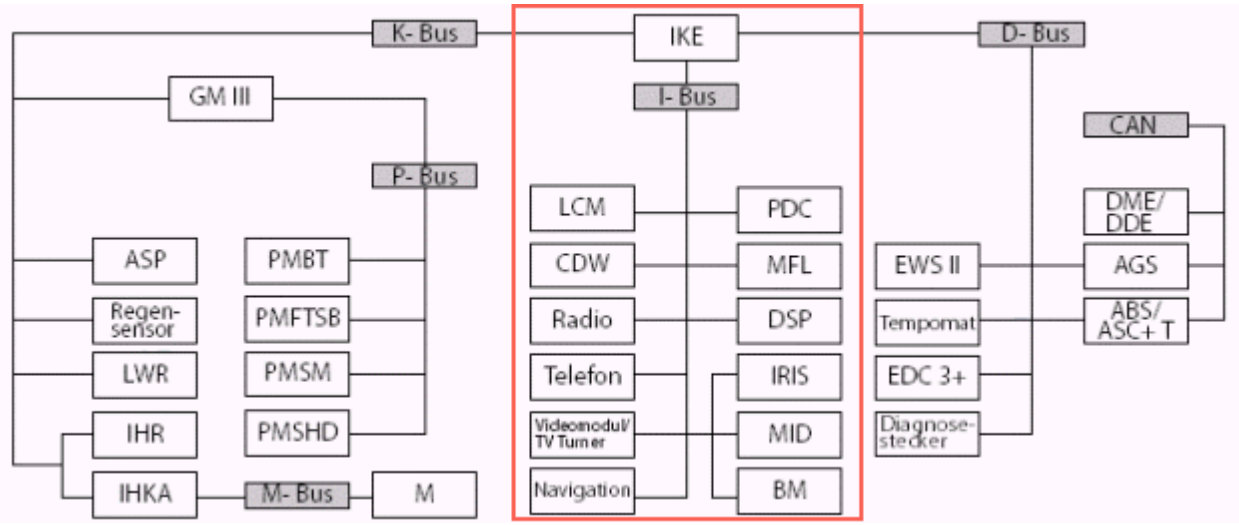
LIN is based on the ISO 9141 specification. The network layout of the system uses a single wire 12V bus, one master and several slave nodes. The slave nodes have no knowledge of the other nodes on the network except the master node, and the master node controls communication on the bus. LIN is optimal for applications such as doors, steering wheels, seats, climate regulation, lighting, rain sensors, or alternators.

The LIN protocol was not designed to be a competitor to the controller area network (CAN) protocol or other high level system. It aims at applications that don’t require the flexibility and data rate of CAN. Thus it is possible for the two systems to be implemented in the one vehicle, and even for the LIN and CAN networks to share data through a gateway. LIN is designed for use in applications that will provide enhanced features in the automobile, but where the cost of using CAN implementations is prohibitive. See diagram below.



(credit to http://www.lin-subbus.org)

The LIN Consortium has grown beyond just the seven members of the steering circle to include over 20 companies worldwide, including most major automotive OEMs. LIN is already being implemented on cars in Europe. According to the LIN Consortium, the number of LIN nodes is expected to rapidly grow over the next decade to an average of twenty per vehicle, or a worldwide volume of approximately 1.2 billion LIN nodes per year.



(Credit to www.openbmw.org for figure and a part of the following explanation)

The I-BUS, as you can see on the figure , handle "multimedia" on-board peripherics. It is absolutely not linked to some security equipments, like airbags, ASP, AGS (so hacking will not hurt you…).

BMW's I-Bus is based on ISO 9141 and K-Bus, a bus largely used in automotive application. It is basically a second K-Bus in the car for handling interfacing between the radio, CD, navigation, and telephone systems, or any other system. In particular, the steering wheel controls for the radio utilize the I-Bus.

The I-Bus is a single wire bus. That means it uses only one wire to send and receive data.
In the car you can find the I-BUS wire at the CD changer connector in the rear, the Navigation system CD-ROM unit connector, the phone connector in the center console, etc.
It's a white/red/yellow wire (see picture below)

<IBUS PHOTO CD + NAV Sys >

You can see the RJ connector (telephone type with 4 pins) I have soldered to the Ibus for a clean and chip connection.

The bus' physical layer is an open collector setup pulled high (+12v) by the bus, and pulled low by the talker. This means that the normal voltage on the wire is +12v (the battery voltage, or Vbatt).

A bit is transmitted by pulling low or shorting the bus with the ground momentarily. This is the reverse of many digital signals where the normal voltage is 0v and is raised high (Vmax which in this case is Vbatt or +12v) to send a bit.

Serial communications on the bus are 9600 bps, 8 data bits, Even parity, 1 stop bit.

Typicals Device IDs

In the system, each device is identified by a unique code, 1 byte length.

Id	Device name
00	Broadcast
18	CDW - CDC CD-Player
30	?????
3B	NAV Navigation/Videomodule
3F	?????
43	MenuScreen
44	?????
50	MFL Multi Functional Steering Wheel Buttons
60	PDC Park Distance Control
68	RAD Radio
6A	DSP Digital Sound Processor
7F	?????
80	IKE Instrument Kombi Electronics
A8	?????
BB	TV Module
BF	LCM Light Control Module
C0	MID Multi-Information Display Buttons
C8	TEL Telephone
D0	Navigation Location
E7	OBC TextBar
E8	?????
ED	Lights, Wipers, Seat Memory
F0	BMB Board Monitor Buttons
FF	Broadcast

I-Bus Packet Structure

The structure of an Ibus packet is the folllowing :

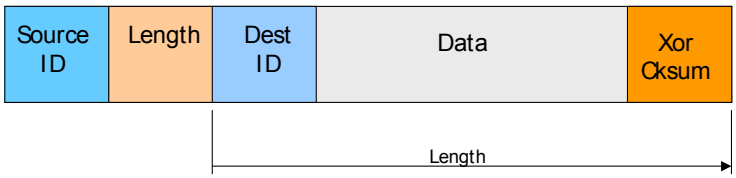
- *Source Device ID*
  The device which needs to send a message to another device

- *Length*
  The length of the packet whithout Source ID and length it-self.

- *Destination Device ID*
  The device which must receive the message

- *Data*
  The message to send to Destination ID

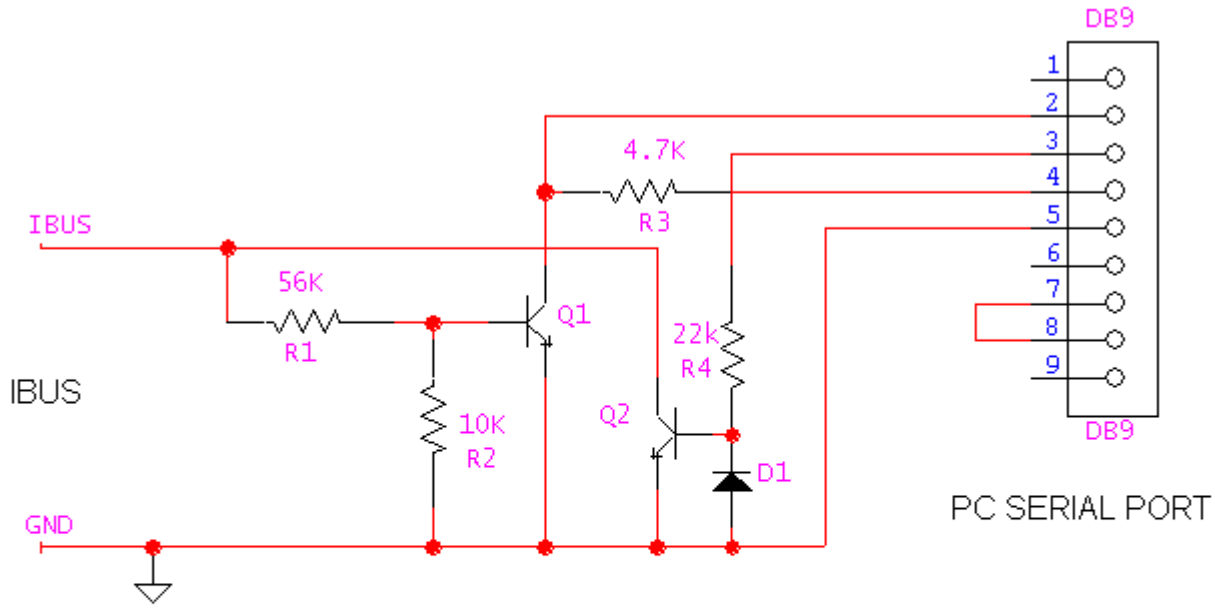
- *XOR CheckSum*
  This byte is used to check the integrity of the message. The receiver will compare that value with its own computation, and if not equal, will reject the packet.



The Xor checksum is a XOR byte per byte for all bytes contained in the packet.
The Xor checksum must be set to 00H before the computation.

Basic Interface

Do not detect contention but the most basic circuit you can find.



Q1,Q2 Generic NPN transistor e.g. 2N2222, 2N3904 etc

D1 Generic silicon diode e.g. 1N4148, 1N914

Q2, D1 and R4 only needed for transmit

Title IBUS Interface
Date 01/31/03
Kevin White

Contention Detection Interface

Thomas L. Wood has built a specific circuit, with a contention detection system, mainly based on a RS latch, and a MAX232A.

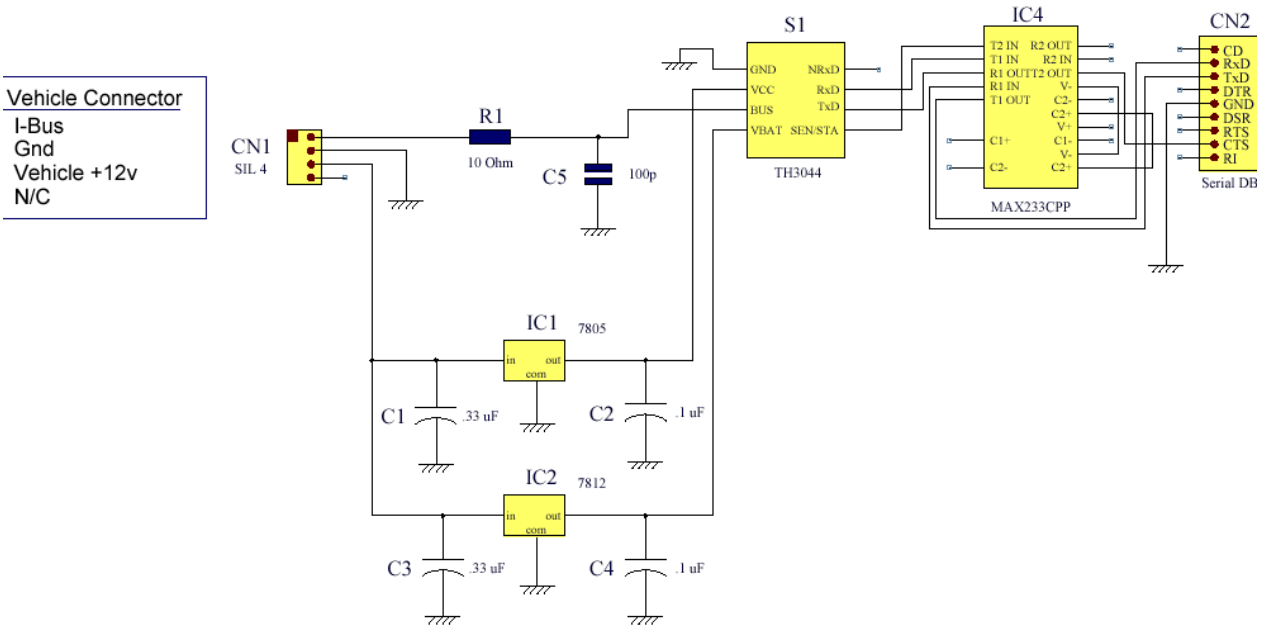
The MAX232A CI from Maxim converts RS232 levels (-12V, +12V) from a PC to TTL levels (0V, +5V).
The rest of the circuit is built with classical logical gates (74HC27, 74HC08, 74HC04).
Note that 74LS type are also working even if they are slower.

To use this circuit, the software must first assert the RTS line, and wait to get the CTS line before sending a message. If you loose CTS right after sending your message, there was a contention.
If your message has been sent, check it in the receive buffer. You must get the echo.

Just for delay : at 9600 Bauds, send one byte take about 1 ms.

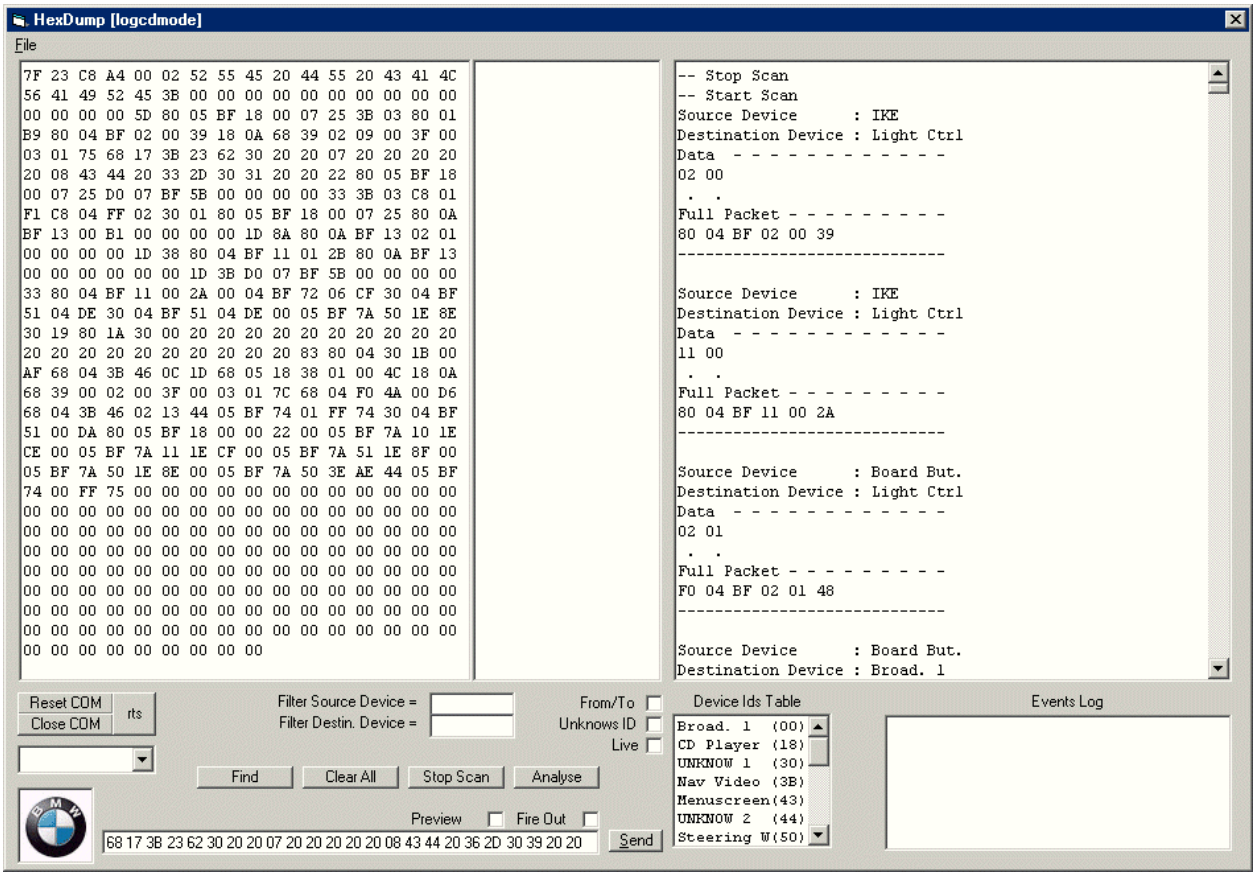
You can find the PDF file for the shematic here :
http://autos.groups.yahoo.com/group/HackTheIBus/files/IBUS%20Interfaces/232TOOC_REV1.pdf

Melexis Chip Interface



The contention is detected by the chip itselft. If the CTS is low, then there is some traffic on the BUS.

The I-Bus Analyser Software. (V1.00)



The I-Bus software analyser allows you to display I-BUS messages in clear text for everybody understanding.You have the choice to scan in real time the COM PORT and I-BUS conversations, to load a binary (.bin), an hexa file, or copy and paste to the hexa area some hexa bytes from a dump window in another software. A big part of my analysis of Ibus messages was done with that software.

Menu "FILE"

Open : Open a binary file or an hexa bytes file. Note that the binary file must be a ".bin" type to be interpreted as is. Hexa bytes can be text files or RTF with colors.

Save : Save the hexa text box to a file. This allows you to save a log from COM port as a text file.

"RTS/rts" button

This button enables or disable RTS. If RTS is enable, the button caption is "RTS" in uppercase letters, else it's "rts". If the text "(1)" appears after the rts/RTS text, that means you hold CTS : you can transmit because it's **C**lear **T**o **S**end.

"Reset COM" button
This will reset (close and reopen) the current COM port. If you have some trouble to communicate with I-BUS, it can help.

"Close COM" button
This will simply close the COM port you were using for IBA.
The "Start Scan" button will automatically re-open the port.
This is usefull is you are using another sofware on the same serial port.

"Stop Scan / Start Scan" button
This button will stop/start to dump bytes from the COM port.
The "Stop Scan' will not close the communication.

If you let your PC running IBA, the log can grow dramatically. The scan mode reduce it self the size to 10% every 30 000 bytes to avoid memory (and windows) crash.

"Clear All" button
This will clear all windows : hexa bytes, ascii , analysed bytes, log windows.

"Find" button
This will hightligth some hexa string you want to search in the hexa windows.

"Analyse" button
This will start the message analysis. The analyser will display clear text message description in the right part of the main window. You can copy and paste results to another software like a text editor.

Analyser options

Filtering packets:

You can use "Filter Source Device", and/or destination device fields. You can enter many devices ID, separated by a comma. e.g. "68,3B" or only one.

e.g. : If you want to see only packet sent by the radio, enter 68 in the "Src id" field then click on "Analyse".

The "FROM/TO" check box allows you to filter only message from and to a specific device.
e.g. If you want to see every packet FROM/TO the CD Player, enter "18" in the "Filter Source Device" field, check "FROM/TO" checkbox then click on the "Analyse" button. This will show you a diagram like the following :

```
Source Device      : CD Player
Destination Device : Broad. 2
Data  - - - - - - - - - - - -
02 01
 .  .
Full Packet - - - - - - - - -
18 04 FF 02 01 E0
-----------------------------

<---Source Device      : Radio
<---Destination Device : CD Player
<---Data  - - - - - - - - - - - -
<---Poll CD Player

<---Full Packet - - - - - - - - -
<---68 03 18 01 72
<-------------------------------
```

Unknow devices ID

As we didn't guess (not yet !) what are all devices ID, by checking this option , the analyser will assume that every bytes from 00 to FFH can be a valid device ID. This will avoid sometimes a great numbers or errors when decoding.
Note : The error counter is the number of byte the analyser couldn't decode as belonging to a valid message.

Live Mode :

The live mode is not a really "live" analyser. It's just activating a 1 seconde timer trigging the "Analyse" button. Only the the last 1000 bytes of the log are tkaen in consideration. The useful point, is that it will synchronize the "virtual" navigation screen (see below).

"Send" button

The Send text box is used to send message you enter directly in hexadecimal value in the input box on the left side of the button. The message must be a valid IBUS message, but the checksum is computed automatically : DO NOT ADD THE CHECKSUM BYTE. IBA will add it ! !

The len of the message is also automatically computed. Simply enter 00 just to let room for the right len in the message whren sending.

You can also send TEXT by using quotes. Only one text string can be sent.

e.g.

-> 68 00 18 01
Will poll the CD player

-> 68 00 3B 23 62 30 'Hello World'
Will display Hello world to the area 0 of the nav screen.

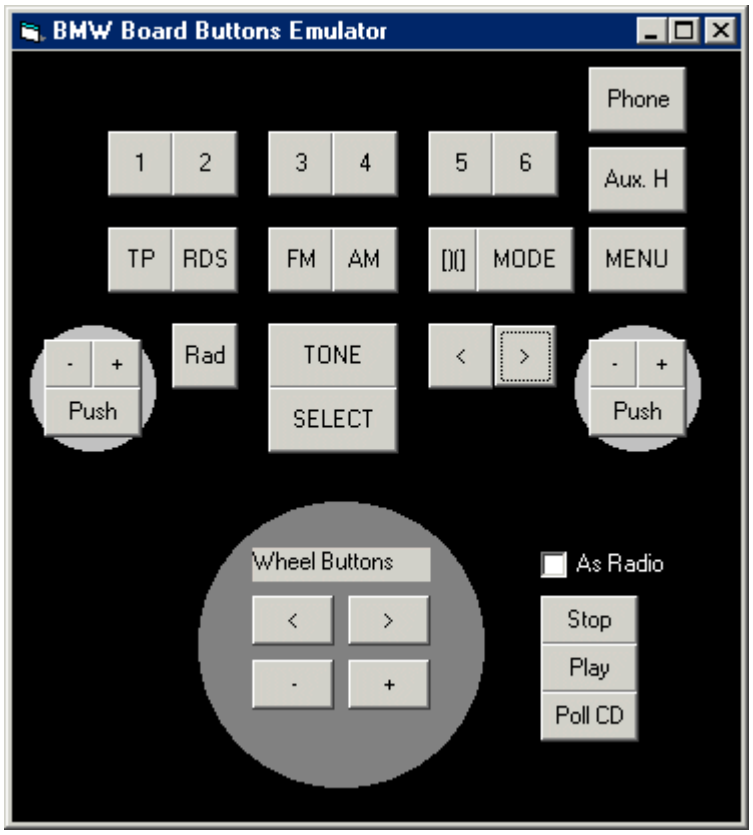
The log is updated each time you send something in the hexa dump window.

Preview
The preview check box is used to see the message before sending it
(e.g. get the checksum and len).

Fire Out
If you want to send a message every second, without having to click each time on the "Send" button, just use "Fire Out". It's usefull for testing circuits, POLL Messages, or to stress a device for simulating bus traffic.

Button Board Emulator.

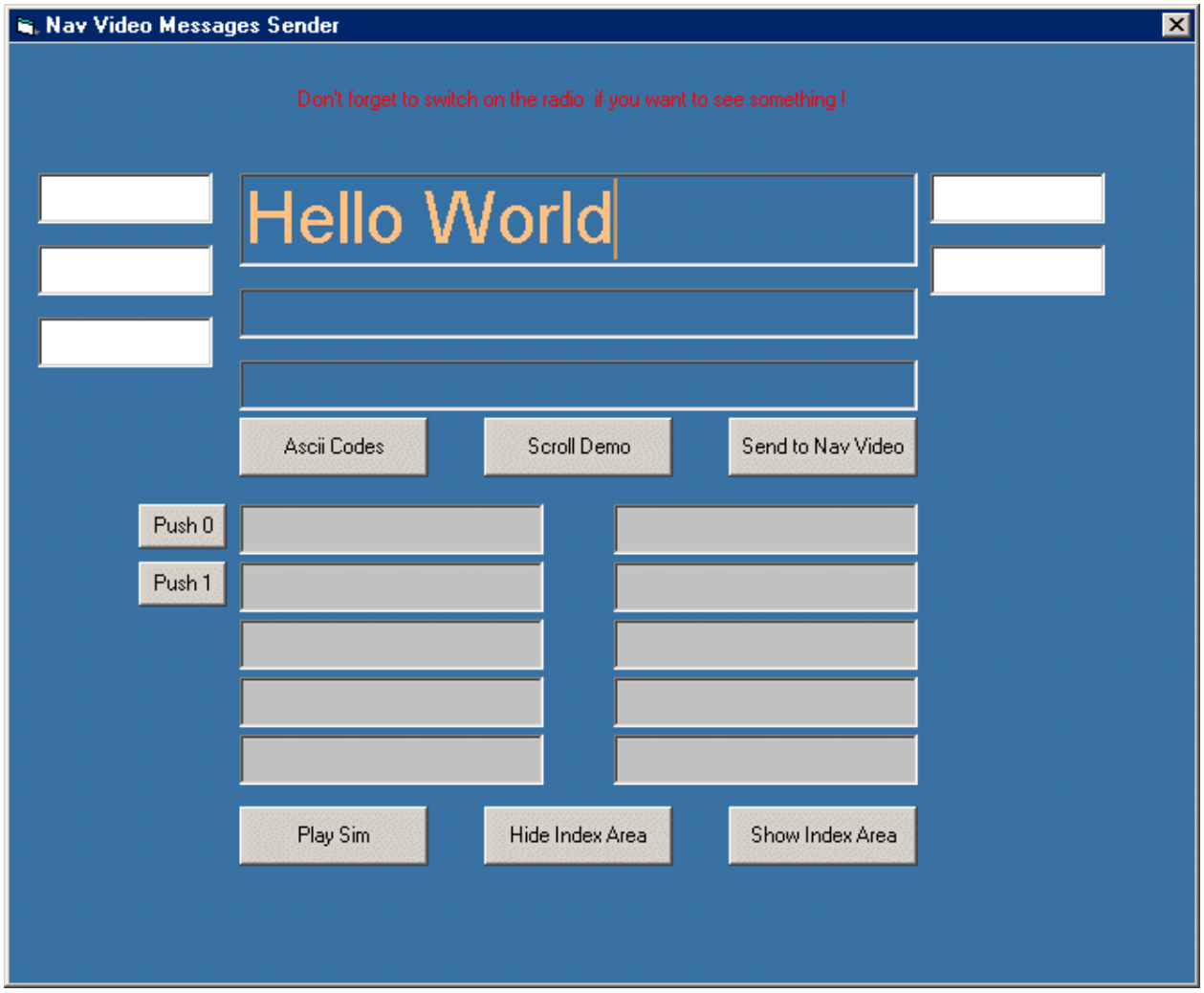
To get the BMB emulator, click on the BMW logo (you wil get also the NAV Screen emulator).



You can use buttons exactly as you use BMB in your car. The release event is simulated.

If you check the "As Radio" box, this will simulate order sent by the radio to the CD player instead of sending BMB events. e.g. If you click on the "1" button, that will send "Play CD#1", and not "BMB Button 1".

Nav Video Messages Sender and Emulator



To get the NAV emulator screen, click on the BMW logo (you will get also the BMB emulator).
(TO BE CONTINUED….)

DEVICE ID = 50H

The cruise control buttons messages are not seen on the I-Bus.
Only audio and telephone buttons send messages.

Messages to Radio (68H)

Button "-" (Volume Down)

32 10

Button "+" (Volume Up)

32 11

Button ">" (Next Track)

3B 01	Push	3B 21	Release

Button "<" (Previous Track)

3B 08	Push	3B 28	Release

Messages to Telephone (C8H)

Dial Number Button

3B 80	Push	3B A0	Release

Button "R/T"

3B 40

DEVICE ID = 80H

Send (urgent ?) Message to OBC Text bar

1A 35 00 <Text>

```
Source Device      : (Unknow ID =30)
Destination Device : IKE
Data Len (Checksum): (83)
Data  - - - - - - - - - - - - -
1A 35 00 20 20 43 48 45 43 4B 20 43 4F 4E 54 52 4F 4C 20 4F 4B 20 20
 .  5  .        C  H  E  C  K     C  O  N  T  R  O  L     O  K
Full Packet - - - - - - - - -
30 19 80 1A 35 00 20 20 43 48 45 43 4B 20 43 4F 4E 54 52 4F 4C 20 4F 4B 20 20 83
----
```

30 19 80 1A 37 18 20 20 20 20 48 45 55 52 45 20 31 39 3A 35 39 20 20 20 20 20

Send 3 bips

30 19 80 1A 37 10
DONG sound

30 19 80 1A 37 01 'TEST'
Display 'TEST' between 2 red arrows.

30 19 80 1A 37 03 'TEST'
Display 'TEST' between 2 blinking arrows

30 19 80 1A 37 **04** 'TEST'
DING sound with arrows
05 : instead of 04 : without arrows

30 19 80 1A 37 08 'TEST'
15 DING

Messages to OBC (E7H)

Set OBC Parameter

24 **nn** 00 <Text>

Where nn is the parameter identifier :

Nn	Parameter	Sample Text
01	Time	" 9:50 "
02	Date	"19.11.2002"
03	Out. Temp.	" +6.5.C"

04	Consumption 1	"9.4 L/100"
05	Consumption 2	"9.4 L/100"
06	Range	"541 KM "
07	Distance	"- 405 KM"
08		"--:-- "
09	Limit	"170 KM/H"
0A	Average Speed	"50.7 KM/H"
0E	Timer 1	" 0.0 SEC"
0F	Aux.Heating Timer 1	"--:-- "
10	Aux.Heating Timer 2	"--:-- "
1A	Timer 2	" 0.0 SEC"

OBC Parameter table

Backward signal. Used for retrovisors auto moving and probably for PDC.

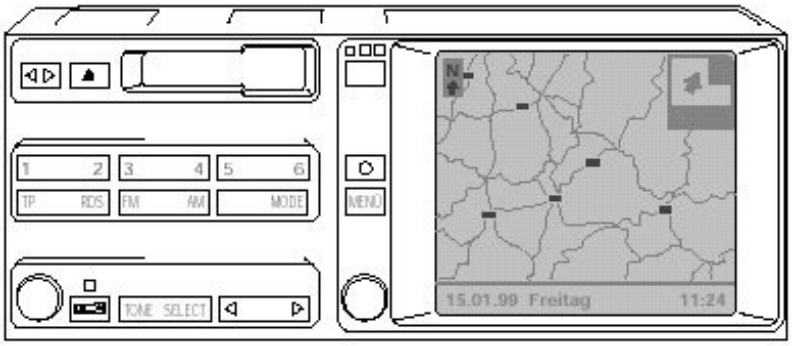
80 0A BF 13 00 13 00 00 00 00 20

Speed Signal

```
80 05 BF 18 06 0E  Display security Message on the TV screen
80 05 BF 18 00 07  Nothing displayed (the car is running lower than 5 KM/H)
```

I have tried to fire the previous message every seconds running in my car, and the TV was working 1 second then I get the message 1 second, then again TV…and so on. (Use the FIRE option of the IBA software)

DEVICE ID = F0H



Messages to Radio (68H)

Button Label	Push Message	Push More 1 Second	Release Message	Button Label	Push Message	Push More 1 Second	Release Message
< > (Reverse Tape)	48 14	48 54	48 94	**^** (Eject Tape)	48 24	48 64	48 A4
1	48 11	48 51	48 91	**2**	48 01	48 41	48 81
3	48 12	48 52	48 92	**4**	48 02	48 42	48 82
5	48 13	48 53	48 93	**6**	48 03	48 43	48 83
TP	48 32	48 72	48 B2	**RDS**	48 22	48 62	48 A2
FM	48 31	48 71	48 B1	**AM**	48 21	48 61	48 A1
(Dolby)	48 33	48 73	48 B3	**MODE**	48 23	48 63	48 A3
TONE	48 04	48 44	48 84	**SELECT**	48 20	48 60	48 A0
< (Previous Track)	48 10	48 50	48 90	**>** (Next Track)	48 00	48 40	48 80
(radio menu)	48 30	48 70	48 B0				

For Radio On/OF unlknow : 4B 05

Radio turnknob

48 06 Push 48 46 Push More 1 Second 48 86 Release

32 n0 Rotate Left 32 n1 Rotate Right

n is between 1,9. It is the step related with the rotation speed (message optimization)

Messages to Navigation / Video (3BH)

Nav turnknob

48 05	Push	48 45	Push More 1 Second	48 85	Release

49 0**n**	Rotate Left	49 8**n**	Rotate Right

n is between 1,9. It is the step related with the rotation speed (message optimization)

48 1A
48 1B
48 1C
48 1D
48 1E
48 0C
48 0D
48 0 E
48 0F
49 21
49 23
49 A1
49 A3
49 E1
49 E3
49 61
49 63

Broadcast Messages (FFH)

Button '(phone)'

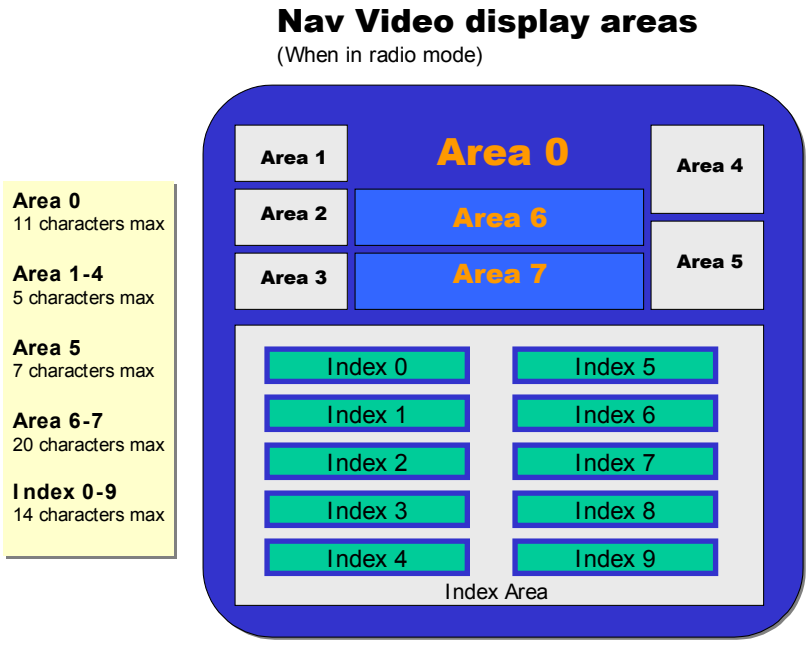
48 08	Push	48 48	Push More 1 Seconde	48 88	Release

Button '(aux Heating timer)'

48 07	Push

Button 'MENU'

48 34	Push	48 74	Push More 1 Seconde	48 B4	Release



DEVICE ID = F0H

Messages to Navigation / Video (3BH)

Write to Area 0 (Title)

23 62 **<attribute> <field value>**

The attribute can be :

30 Normal
FF Blinking

The field value can be a text value, or a combination of sub-field values.

Note : the area is not erased before displaying text. It is to the sender to pad the text with spaces.

Write to Area n

A5 62 01 **nn** <text>

Valid areas are 01,02,03,04,05,06,07

Note : Areas are not erased before displaying text. It is to the sender to pad the text with spaces.

Radio to Nav

After switching off the radio

46 0C

Fill the "index" area (the menu displaying CD1, CD 2, CD3, …)

21 60 00 60 06 <TEXT 1> 06 <TEXT2> 06

21 60 00 43 <TEXT 3> 06 06 06 <TEXT 4> 06

07 <TEXT 5> 06 <TEXT 6> 06 06

Field
Tag
05 Previous Field ??
06 Next Field
7 Next +1 field ??
8 Next +2 field ??

21 60 00 **60** ← 60 could be the global index area ?

Index screen is structured as following (it's a menu) :

0 @ 5 E
1 A CD1 **6 F CD4**
2 B CD2 **7 G CD5**
3 C CD3 **8 H CD6**
4 D 9 I

Set Text Index n

21 60 00 **nn** <text>

Index	nn	Index	nn
0	40	**5**	45
1	41	**6**	46
2	42	**7**	47
3	43	**8**	48
4	44	**9**	49

Text is 14 chars max.

Refresh the index area

68 06 3B A5 60 01 00

A5 60 01 00

NAV TO RADIO
3B 06 68

31 60 00 **nn**

where nn is

Index	Push	Release	Index	Push	Release
0	00	40	**5**	05	45
1	01 (CD 1)	41	**6**	06 (CD 4)	46
2	02 (CD 2)	42	**7**	07 (CD 5)	47
3	03 (CD 3)	43	**8**	08 (CD 6)	48
4	04	44	**9**	09	49

Note that index 1,2,3 and 6,7,8 do the radio asking a CD change.

Après

46 02 Come back to the main menu ?

46 0C : Come back to the On board Computer screen ?

```
Source Device      : Radio
Destination Device : Nav Video
Data  - - - - - - - - - - - -
21 60 00 20
 !   `   .
Full Packet - - - - - - - - -
68 06 3B 21 60 00 20 34
```

Messages to Instrument Kombi Electronics – IKE (80H)

```
Display CD title in the OBC text bar

Source Device      : Radio
Destination Device : IKE
Data Len (Checksum): (1)
Data  - - - - - - - - - - - - -
23 62 30 07 20 20 20 20 20 08 43 44 20 32 2D 30 34 20 20 BC 20
 #  b  0  .                 .  C  D     2  -  0  4        .
Full Packet - - - - - - - - -
68 17 80 23 62 30 07 20 20 20 20 20 08 43 44 20 32 2D 30 34 20 20 BC 20 01
----------------------------
```

Special characters : BC = " ▶ "

Messages to CD Player (18H)

Poll CD Player

01

Request Current CD and Track Status

38 00 00

Stop Playing

38 01 00

Start Playing

38 03 00

Fast Scan

38 04 **nn**

Where nn is :

00 Forward
01 Backward

Change CD

38 06 **nn**

Where nn is the CD number to load (usually between 01,06)

Scan Intro Mode

38 07 **nn**

Where nn is :

01 Scan mode ON
00 Scan mode OFF

Random Mode

38 08 **nn**

Where nn is :

01 Random mode ON
00 Random mode OFF

Change Track

38 0A **nn**

Where nn is :

00 Next track
01 Previous track

DEVICE ID = 18H

CD Player behaviour

When the I-Bus wakes up, the CD player starts to announce it-self ("02 01" msg).
It's seems the CD announce is not mandatory.

The CD must send immediatly a poll response ("02 00") when polled by the radio every 10 secondes.

If the CD doesn't respond to 2/3 polls, the radio considers that there is no CD Player (or not anymore).

Broadcast Messages (FFH)

Poll Response

02 00

Announce

02 01

Messages to Radio (68H)

- dd is the disk index (usually between 01,06).
- tt is the track index.

CD and Track Status Not Playing Response

39 00 02 00 3F 00 **dd tt**

CD and Track Status Playing Response

39 00 09 00 3F 00 **dd tt**

Track Start Playing

39 02 09 00 3F 00 **dd tt**

CD Status Scan Forward

39 03 09 00 3F 00 **dd tt**

CD Status Scan Backward

39 04 09 00 3F 00 **dd tt**

Track End Playing

39 07 09 00 3F 00 **dd tt**

CD Seeking

39 08 09 00 3F 00 **dd tt**

DEVICE ID = 3BH

Messages to IKE (80H)

OBC Parameters Get/Reset

41 **nn cc** <Text>

- nn is the parameter identifier.
- cc is the command :

01 Get Value
10 Reset Value

See the OBC parameter table for "nn" value.

3B 04 F0 4F 00 : Switch off nav screen

3B 04 F0 4F 00 : Switch off

```
Source Device      : Nav Video
Destination Device : IKE
Data  - - - - - - - - - - - -
15 01 04 80 4A
 .  .  .  .  J
Full Packet - - - - - - - - -
3B 07 80 15 01 04 80 4A 66
------------------------------

Source Device      : IKE
Destination Device : Light Ctrl
Data  - - - - - - - - - - - -
15 01 04 80 4A
 .  .  .  .  J
Full Packet - - - - - - - - -
80 07 BF 15 01 04 80 4A E2
------------------------------

Source Device      : Nav Video
Destination Device : IKE
Data  - - - - - - - - - - - -
15 06 04 80 4A
 .  .  .  .  J
Full Packet - - - - - - - - -
3B 07 80 15 06 04 80 4A 61
------------------------------
```

```
Source Device      : IKE
Destination Device : Light Ctrl
Data  - - - - - - - - - - - -
15 06 04 80 4A
 .  .  .  .  J
Full Packet - - - - - - - - -
80 07 BF 15 06 04 80 4A E5
----------------------------
```

00 German

01,02,05 TIME GB

03 ORA Esp ?

04 HORA IT ?

06 FR HEURE

DEVICE ID = C8H

Messages to IKE (80H)

TEL

C8 04 E7 2C 11 : Display the phone symbol

Source device ID is 7F, but this message (called SPTI) was sent by navigation system.

It have the following format:

7F 14 C8 A2 - src, len, destination and message type.
00 00 - constant
52 33 19 40 - Y coordinate in BCD format.
00 13 11 51 70 - X coordinate in BCD format.
X can vary from 0° to 180°, so the storage is
one byte bigger.

coordinate sign is stored in the lowest nibble.

00 00 00 - constant
11 00 - current time in BCD format, (hh:mm, 24h format)
00 00 - constant
0B - XOR checksum

```
7F 14 C8
A2 00 00
47 55 48 00
00 02 07 37 81
00 00 00
09 43
00
A2
```

Nav system have only one message with BB as destination ID:
STVS 0 = 3B LL BB 4F 02 00 XX
STVS 1 = 3B LL BB 4F 01 00 XX (LL - len, XX - CRC)

3B LL BB 4F 01 00 XX (switch to TV mode)
3B LL BB 4F 00 00 XX (switch back)

Thanks a lot for your fast answer - you were right, I have checked everything yesterday in the car. Here is the message:
80 0A BF 17 0D 5E 00 20 CC 72 CC 23
It seems the total kilometers are coded into the data bytes 5-7 (0D 5E 00); the total kilometer count can be calculated as followed:
km = Byte7 * 65536 + Byte6 * 256 + Byte 5
In my case total kilometers were 24077 yesterday, which result in 00 * 65536 + 5E * 256 + 0D = 24077.