I2CCHIP

DATA FORMATTING WITH BL233 AND I2C2PC

1 Introduction

This document summarises information in the Bl233B datasheet. You should read all the associated sections of the BL233B datasheet

The BL233B returns data in a simple default format.
It can be much easier to process the data if it is formatted to suit the application.

- Formatting into comma delimited tables will simplify reading with Excel or Matlab.
- Adding headers may make the data files self documenting
- Adding record numbers or timestamps to data points

This is especially true of data gathering applications where you will be capturing data to a file for post processing.

The most important point below is this. Use **J88** to suppress automatic EOL chars.
Then use comma and fullstop to format the data into CSV format.

2 EOL, Separator Chars, and Data formatting

EOL (LF 0x0A by default) is sent after every read command. eg

```
S41 R01 SA1 R02 P S41 R01 S42 R02 P
01[eol]
        FE02[eol]
                01[eol]
                        FE02[eol]
```

This makes for a messy data file. You want a file like this for Excel

```
01,FE02[eol]
01,FE02[eol]
```

Use the **J** command to set fControlSuppressEOL in the Control register allows, to suppress the automatic EOL character.
J88 will change it. It only needs to be sent once at the start.
fControlSuppressEOL will affect all operations that return data except d**U**mp, which always has an EOL.

Comma is echoed back and fullstop returns EOL. Now you can tabulate your data

```
J88
[set fControl to suppress EOL char]
S41 R01, SA1 R02 P. //read the first line
S41 R01, S42 R02 P. //read the next line
01,FE02[eol]
01,FE02[eol]
```

3 Typeback Chars and Strings

T types any chars back to the PC. It is very useful for formatting and synchronising data.

```
S407D T563D R01
V=7D[eol]
```
[set 8574 to 0x7D][Type "V=" back to PC][reads 8574: 0x7D]

- make return data more readable or parseable
- you don't have to wait for a specific reply

You can explicitly insert any char or string using the **T** command. These can include control chars. eg TAB, CR etc. The T command is followed by the hex value of the characters you want to send.

```
J88                         //set Control to suppress EOL char
T566F6C747320 R02, T20416D707320 R01.
Volts 5A5A, Amps 5A[eol]
```

Character	Hex	Hi Char	
Space	1 0	A0	
Quote	2 2	A2	
Tab	0 9	89	
CR	0 D	8D	
0x	3 0 7 8	B0 F8	
semicolon	3 B	BB	

Table 1: Special Typeback Character Codes

3.1 Typeback Synchronising Chars

If you use the typeback chars for synchronising the data, then remember that the I2C adaptor is going to return upper-case hex chars, and a few special upper case chars in certain cases (eg N,K)
Type different chars or lower case chars back, then it is very easy to separate them out of the data stream.

3.2 Long Strings

Each char typed back, requires 2 chars to be sent to the adaptor. If this is slowing your application down you can do two things.

3.2.1 Use EEProm Macro for String

The start-up welcome "Hi I2Cad VXXX" message works this way.

Store the **T** sequence in the EEProm. This will reduce a long sequence to a 3 char call. Try running the startup macro (at address 00) by typing

```
>00
```

The adaptor replies with the welcome message:

```
HI I2C v118
```

3.2.2 Use **HiCharsAsAscii.**

Add 0x80 or set bit 7 of any ascii char you want to send. Now you only send 1 char for each char typed back. Note that this also works inside macros.

3.3 Hex Formatting

The data read from the adaptor is always hex. Some applications will read the hex chars directly if they are preceded by "0x"

```
T3078 R02, T3078, R01
0x5A5A,0x5A
```

4 Translating Chars in data files

You do not necessarily need to completely format the data at the adaptor level. This may unnecessarily burden the comms with lots of strings. The unix/mac utility TR (available on the PC in UnixUtils package) is useful for translating single chars in files eg commas to spaces, LF to CR etc.

The SED utility, perl scripts etc can be used to translate a key char, into a header string. For example we could program our adaptor to return this compact data:

```
v5AA5a78r94
```

Then we could used SED or Perl to translate the chars v,a,r to verbose words and make our file:

```
Volts 5AA5, Amps 78, Ohms 94[eol]
```

5 Timestamps

For data logging it is frequently useful to prepend records with a timestamp.

- Use the T command to type the timestamp back.
- Use Realterm to capture the data to file. Realterm can automtically prepend timestamps in several formats, to each line.
- Capture each line to a file, prepend the system time. See http://www.i2cchip.com/linksys.html for an example where a unix system is collecting data into files and prepending the system time using *date*

6 Line Numbers

The BL233B has a message number facility that can be used to put line numbers or record numbers into a file. When it is enabled, each read command is preceded by a 1 byte message number. (see BL233B datasheet)

This is controlled through the Control register (**J** command) and the **M** command.
We can use this to make record numbers.
After each read the message number is incremented, unless it is disabled in the control register.

```
J88   //suppress eols
M R01. R01. R01. //M clears the message number
00A5[eol]
01A5[eol]
02A5[eol]
```

Note that unfortunately, you cannot get a comma between the message number and the data read. You can pair the message number with a read of the status register instead of an I2C read.