I2CCHIP

DATA FORMATTING WITH BL233 AND I2C2PC

1 Introduction

This document summarises information in the B233B datasheet. You should read all the associated sections of the BL233B datasheet.

The BL233B returns data as hex 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.
BL233C adds the ability to manipulate data and do decimal conversions.

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.

Table of Contents

ph +64 21 623-402 www.i2cchip.com sales@i2cchip.com

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, 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]
```

2.1 Reading large blocks and formatting

If you want to read large blocks of data, without sending the address repeatedly, you can use the manual nack mode.
See **BL233B** datasheet section ***5.92 Reading more than 255 bytes / Read without NACK***
eg Read 256 bytes in 64 byte lines

```
J8C
S41 40.40.40.N40 P
J88
```

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
- put sync characters and markers in so you don't have to wait for a specific reply
- insert format codes for HEXCSV2BIN (below)

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		Char	Hex	Hi Char
Space	1 0	A0		d	64	E4
Quote	2 2	A2		u	75	F5
Tab	0 9	89		v	76	F6
CR	0 D	8D		s	73	F3
0x	3 0 7 8	B0 F8		t	74	F4
semicolon	3 B	BB		f	66	E6
				g	67	E7

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
```

The welcome message is simply a T sequence macro.

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" (eg javascript parseInt)

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

4 *HEXCSV2DEC:* Converting HEX files to Decimal

The I2CCHIP command line utility ***HEXCSV2DEC.EXE*** converts a CSV file containing hex numbers, to decimal. HexCSV2DEC is a commandline program included with Realterm V2.0.0.65+, in the same directory as Realterm

While Excel, OpenOffice, Matlab and other applications have hex handling routines, using HEXCSV2DEC may be quicker and easier.

- Makes data file readable and meaningful without processing
- avoids lots of hex commands in spreadsheets

Hex2Dec only converts valid *uppercase* hex. Quoted strings, floats (numbers with a decimal point) and lower case will be passed through unchanged, except for explicit decimals as below.
Note that strings are best quoted to avoid problems.

- Hex will be converted as Unsigned, Big Endian, by default. Format chars s,t,u,v can be used on the command line to set the default format.
- A Format String on the commandline can specify different formats for each column
- Max length of each hex number is 8 chars (32bit numbers)
- Each individual number must be delimited with a comma
- Floats, eg Matlab Timestamps (below) will be passed through
- Quoted strings will pass through
- CRLF and LF terminated lines (pc and unix files).

4.1 Format Codes

Remembering the format codes is easy. The big-endian formats *s,u,f* are obvious. The equivalent little endian code is the next letter in the alphabet after the big-endian one.

Format Char	Meaning	Endian	Notes	Echo
d	BCD/HEX	n/a	Passed through unchanged	T64
u	Unsigned	Big		T75
v	unsigned	Little		T76
s	signed	Big		T73
t	signed	Little		T74
f	IEEE float	Big	4byte singles only	T66
g	IEEE float	Little	4byte singles only	T67
b	binary	n/a		T62

Table 2: Format Codes for HexCSV2Dec

4.2 Setting the Default Format

The first parameter can be an option to set the default format for all hex fields.
Eg -u -s -v -t

```
HEXCSV2DEC -s <input file> <output file>
```

4.3 Using a Format String

If there is more than one char in the format option, then it is specifying each hex field with each character.
If there are more fields on a line that format chars, then the remaining fields will default to using the first format char.

```
HEXCSV2DEC -sufgvt <input file> <output file>
```

The format chars are only used for hex fields. Fields with the explicit format character, quoted strings, and floating point strings don't count in the format string.
e.g.

```
HEXCSV2DEC -svb <input file> <output file>
```

will convert:

```
80,0080,A,8000
```

to:

```
-128,32768,1010,-32768
```

4.4 Explicitly specifying the format (in file)

You can explicitly precede hex with a single format char from the list below. The “T” command is used to echo the format char into the data string. Eg “T64” will put “d” into the data stream.

```
T64R02,T73R02
d1234,s1234
```

Obviously using lots of T commands and format chars will make comms slower, and files larger. Using the format string on the command line is a more efficient choice as long as the data is the same on every line which has hex.

4.4.1 BCD in Hex Files

In some cases you might have BCD data. Realtime Clock chips commonly store their data as BCD bytes not binary. As these are already decimal, you do not want to convert them as if they were hex.
Precede BCD/decimal values with "d" eg "d1234" will pass through as "1234"
The T64 command can be used to prepend a decimal value with a "d"

4.5 Examples

Usage:
HEXCSV2DEC <input file> <output file>
will convert:

```
81,FFFF,"Tues",1.234,d1234,s8000,t0080
82,FFFE,"Wed",1.234,d1234,bA5
```

to:

```
129,65535,"Tues",1.234,1234,-32768,-32768
130,65534,"Wedn",1.234,1234,10100101
```

4.6 Making Header Lines for Spreadsheets

If you plan to open the file with a spreadsheet, then you might want to start the file with a header line. As long as the column headings are in quotes, they will pass through unchanged.

5 Translating Chars in data files

You do not necessarily need to completely format the data at the BL23 / I2C2PC 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]

6 Timestamps

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

- Use Realterm to capture the data to file. Realterm can automatically prepend timestamps to each line, in several common formats.
- Use the T command to type the timestamp back. In this case the PC is sending the time to the adaptor to send back.
- 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 with a script using *date*
- Use an I2C realtime clock chip, and read the time from that at each line.

7 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.

8 Interpreting & Converting HEX

If you need to see what various values look like in hex for integers/floats/big-little endian/signed etc, try http://www.i2cchip.com/realterm/Keiths_Bitometer.exe

9 Revisions

Rev	Date	Change
14	04/11/08	1st Release
17	06 Feb 15	Added section on reading more than 256 bytes, NACK
22	15 Jan 17	Add links, SignPDF

ph +64 21 623-402 www.i2cchip.com sales@i2cchip.com