BLE Software Spec

Pat Walp, KSI -- 2023 April 17

# Introduction:

This document is my concept of software that will allow us to use the CYBLE in our system. Ideally Arrow/Infineon would develop all of this, leaving very minor tweaks for us to do. In the event that this is too much, the more important part is the software running on CYBLE. Please have a look at this, and then let's talk about it.

The level of detail here is intended to be a balance of full "specification" and reasonable brevity. I can think of a lot of things that should be added to specify things completely, but for the most part these are things I don't really care about. Questions and discussion are welcome.

# Two Components:

1. Running on CYBLE
   1. Carry out commands from host
   2. Collect data from 18 bits of digital inputs (GPIO)
   3. Collect data from 17 ADC channels
   4. Talks to 8 i2c slaves within target
   5. Talks to FPGA UART within target
2. Running on PC ("host")
   1. Accepts commands from command line
   2. Returns results of those commands in the same terminal
   3. Provides options to log output to
      1. standard output
      2. a text file
   4. Provides option to accept input from
      1. standard input
      2. a text file
   5. Command-line editing: The usual
      1. up/down arrows for command history
      2. left/right arrows to move cursor within a command
      3. type to insert, backspace to delete

# Host (PC) User Interface

This is a command line application, compiled to an executable "dble.exe". It is executed within a DOS window by typing "dble". Within the application each command is entered on a command line. Output is displayed immediately after. Commands that interact with the target (other than uart) start with an optional time stamp.

Commands:

**rb GPIOs\_SPEC** -- read bits and display

GPIO\_SPEC is a list of ports, e.g., "P5.0,P0.4,P6.7"

Output is a single line of text, starting with a time stamp in milliseconds, followed by a bit vector with the states of the bits in GPIO\_SPEC, terminated with a newline character. For example:  
"1,234.560ms 10\_1110\n"  
Long bit vectors have "\_" inserted every four characters for readability.

**rbl** -- repeat the last **rb** command in a loop. Ctrl-C stops.

**lp PERIOD** -- set loop period in milliseconds. If an iteration takes more than **lp** milliseconds, execute the next iteration immediately, prepending its output with "\*". Default=0

**bin** -- set the radix to binary (for **rb** command)

**hex** -- set the radix to hexadecimal (for **rb** command)

**toff**/**ton** -- turn time stamp display off/on for future commands

**ri BUS ADDR** -- read from an i2c register. BUS is a number from 0 to 7 specifying the i2c switch port. ADDR is the 7-bit i2c address (hexadecimal).

**wi BUS ADDR DATA** -- write to an i2c register. BUS is a number from 0 to 7 specifying the i2c switch port. ADDR is the 7-bit i2c address (hexadecimal). DATA is the data to be written (hexadecimal).

**ril** -- repeat the last **ir** command in a loop. Ctrl-C stops.

**wil** -- repeat the last **iw** command in a loop. Ctrl-C stops.

**ad ADC\_SPEC** -- read ADCs specified in ADC\_SPEC and display in volts. For example, if ADC\_SPEC is "10,11,16,0", output has the format of "11.974ms 0.234 1.189 0.074 3.333\n".

**adl** -- repeat the last **ad** command in a loop. Ctrl-C stops.

**rt** -- reset time to 0, and don't restart it until the next loop command (**rbl**, **ril**, **wil**, **adl**)

**uart** -- change to UART-mode, during which keyboard strokes are echoed to the FPGA UART, and characters received from that UART are echoed to the output. Ctrl-alt-C exits this mode.

**logfile** FILENAME -- start logging output to text file. If "FILENAME" is omitted, open a dialog box to choose location and file name.

**logstop** -- stop logging output to text file and close the file

**infile** FILENAME -- execute all the commands in text file

**help** -- print a summary of commands (maybe just the contents of this "spec")