Notes for cs452
Busy-wait IO Documentation

Bill Cowan
University of Waterloo

The busy-wait I/O library, `bwio.a`, is a very simple implementation of, mostly output functions. Busy-wait means that the code spins testing ready bits in the hardware, producing output or acquiring input, when the hardware is ready. This means it is very robust, and keeps working when almost everything else is defunct. It cannot, however, share hardware with interrupt-driven IO, which you will later be implementing as part of your kernel. You need to change its implementation in order to do the first assignment.

You should observe one small peculiarity of these functions. My goal is to keep busy-wait IO as close to the hardware as possible. There it does not remove nulls, and it transmits line-feed (\n) and carriage return (\r) without modification. To get what you are used to in Unix you must output both of them. In other words, these functions assume that you know what you are doing and sometimes crap out when you do it wrong. That's because they are a temporary scaffold, later to be supplanted by real IO.

Here is a brief description of the most useful functions.

a. Initialization

a.1. Turn the Fifos off and on

Name. `bwsetfifo` – turn the fifos off and on.

Synopsis. `int bwsetfifo( int channel, int state )`

Description. `bwsetfifo` enables the fifo if the state argument is `ON`, and disables them if the state argument is `OFF`.

Returns. 0 for success, -1 if the channel is invalid.

a.2. Set the ocmmunication speed

Name. `bwsetspeed` – set the transmit/receive speed of a channel.

Synopsis. `int bwsetspeed( int channel, int speed )`

Description. `bwsetspeed` sets the transmit/receive rate of a channel in bits per second. At present only two speeds are provided: 115,200 bps for the monitor, and 2400 bps for the train controller.

Returns. 0 for success, -1 if the channel or the speed is invalid.

B. Output

b.1. Character at a Time Output

Name. `bwputc` - character at a time output.

Synopsis. `int bwputc( int channel, char ch )`

Description. `bwputc` tests the Transmit Buffer Empty bit in the `channel` UART until it sets, then puts `ch` in the buffer to be transmitted.

Returns. 0 for success, -1 if the channel is invalid, spins for ever if the UART is not turned on, etc.

b.2. Output a String

Name. bwputstr – output a string character by character

Synopsis. int bwputstr(int channel, char *str)

Description. `bwputstr` outputs a string character by character. Between characters it spins waiting for the Transmit Buffer Empty bit to set, then puts the next character into the buffer to be transmitted.

Returns. 0 for success, -1 if the channel is invalid, spins for ever if the UART is not turned on, if the string is not null-terminated, etc.

b.3. Output a Word

Name. `bwputr` – output a machine word in hexadecimal

Synopsis. `void bwputr( int channel, int reg )`

Description. `bwputr` outputs a 32 bit word in hexadecimal. Between characters it spins waiting for the Transmit Buffer Empty bit to set, then puts the next character into the buffer to be transmitted. It is most useful when you want to print the contents of a register or a memory location. It is easy to call from assembly: put the channel number into register 0, put the word to be printed into register 1, then branch with link to `bwputr`.

Returns. No return value, but spins forever if the UART is not turned on, if the string is not null-terminated, etc.

b.4. Formatted Output

Name. `bwprintf` – provide formatted output

Synopsis. `void bwprintf( int channel, char *fmt, <fmt arguments> )`

Description. `bwprintf` uses a format string like ordinary printf, with formatted escapes when variables are to be substituted in the format. Supported escapes are

- `%s` – null-terminated string,
- `%c` – ASCII representation of an unsigned character,
- `%d` – signed decimal integer, and
- `%x` – unsigned hexadecimal integer.

The formatted output is transmitted character by character. Between characters it spins waiting for the Transmit Buffer Empty bit to set, then puts the next character into the buffer to be transmitted.

Returns. No return value, but spins for ever if the UART is not turned on, if the format string is not null-terminated, etc, and crashes ignominiously if the types of arguments do not match the types of the format escapes.

c. Input

c.1. Character at a Time Input

Name. `bwgetc` – get a single character from an input channel

Synopsis. `char bwgetc( int channel )`

Description. `bwgetc` spins waiting for a character to appear in the input buffer.

Returns. The character that is read. Waits forever if no character is typed.