U-Boot Standalone Application (Exports) API

1 Introduction

This document contains details on the U-Boot 1.1.4 Exports API used by standalone applications. Most of the details of this document were derived from reading the U-Boot source code relevant to the gumstix/PXA platform. While most of these functions are probably similar on other platforms, they may not be the same.

2 API Functions

2.1 get_version

`unsigned long get_version(void):`

Returns the version of the U-Boot Standalone ABI. Should be version 2.
See u-boot-1.1.4/doc/README.standalone for details.

2.2 getc

`int getc(void):`

Reads the next character from *stdin*. By default, this returns the least significant byte of the FFUART Receive Buffer Register if the Data Ready bit is set (that is, if a new character is available), otherwise the function blocks until a character is made available in the Receive Buffer Register.

This function always returns a value in the range 0–255 and it always returns with success.

2.3 tstc

`int tstc(void):`

Tests to see if a character is available on *stdin*. By default, this returns the value of the Data Ready bit in the FFUART's Line Status Register.

This function returns zero if a character is not yet available, and returns non-zero if a character is available. If this function returns non-zero, the next call to `getc()` is guaranteed not to block.

2.4 putc

`void putc(const char c):`

Writes the character `c` to *stdout*. By default, this writes the value of `c` to the FFUART's Transmit Holding Register as long as the Transmitter Empty bit of the Line Status Register is set, otherwise the function blocks until the Transmitter Empty bit is set (and all the data in the transmitter has been sent).

In other words, this function will always wait for the FFUART to finish transmitting data, and will not drop data on output.

2.5 puts

`void puts(const char *s);`

Writes the string `s` to *stdout*, generally by making successive calls to `putc()` internally. Note that, unlike the definition of `puts(3)` from the C standard library, U-Boot's `puts()` does not automatically append a trailing newline.

2.6 printf

`void printf(const char *fmt, ...);`

Writes the string described by `fmt` and succeeding arguments to *stdout*. Internally the string is formatted by `vsprintf()` and printed by `puts()` (and thus, follows `puts()`'s output behavior). The format of the `fmt` string follows that of `printf(3)` from the C standard library.

2.7 install_hdlr

`void install_hdlr(int vec, interrupt_handler_t *handler, void *arg);`

This function is not implemented for the PXA processor.

2.8 free_hdlr

`void free_hdlr(int);`

This function is not implemented for the PXA processor.

2.9 malloc

`void *malloc(size_t n);`

Allocates `n` bytes of memory and returns a pointer to it. Returns 0 if not enough space is available.

The heap size configured for `malloc` on the gumstix is approximately 128 kB, and is shared with other U-Boot datastructures (that is, it's not reserved exclusively for standalone application use).

See u-boot-1.1.4/common/dlmalloc.c for details.

2.10 free

`void free(void *p);`

Deallocates the portion of memory starting at the address contained in pointer `p`. Pointer `p` must be the result of a previous call to `malloc()`.

2.11 udelay

`void udelay(unsigned long usec);`

Sleeps for `usec` microseconds before returning.

2.12 get_timer

`unsigned long get_timer(unsigned long base);`

Reports the numbers of ticks elapsed as reported by the OS Timer (OSCR register, 3.6864 MHz) (3.25 MHz for `verdex-pro` boards) subtracted by `base`.

2.13 vprintf

`void vprintf(const char *fmt, va_list args);`

Writes the string described by `fmt` and argument list `args` to *stdout*. Internally the string is formatted by `vsprintf()` and printed by `puts()` (thus follows `puts()`'s output behavior). The format of the `fmt` string follows that of `printf(3)` from the C standard library.

2.14 do_reset

`void do_reset(void);`

Hard resets the CPU.