## Использование устройств с символьным режимом

Устройства с символьным режимом – это аппаратная периферия, которая последовательно передаёт и (или) принимает символы. Характерным устройством с символьным режимом является UART. Устройства с символьным режимом регистрируются в файловой системе HAL в качестве узлов. В основном, программа ассоциирует файловый дескриптор (идентификатор) с именем устройства, а затем пишет и читает символы в файл или из файла, используя операции с файлами ANSI Си, определённые в **file.h**. HAL также поддерживает концепт стандарта входа, стандарта выхода и стандарта ошибки, позволяя программам вызывать I/O функции из **stdio.h**.

### Стандарт входа, стандарт выхода и стандарт ошибки

Использование стандарта входа (stdin), стандарта выхода (stdout) и стандарта ошибки (stderr) – это простейший путь реализации простой I/O консоли. HAL закулисно управляет stdin, stdout и stderr, что позволяет вам посылать и принимать символы по этим каналам не касаясь напрямую управления файловыми дескрипторами. Например, HAL направляет выход от printf() на стандартный выход, а perror() на стандартный выход ошибки. Вы ассоциируете каждый канал с определённым аппаратным устройством, манипулируя настройками BSP.

Код в примере 6-3 отображает классическую программу "Hello World". Эта программа посылает символы в любое устройство, она ассоциирована с stdout на стадии компиляции.

**Example 6–3.** Hello World

```
#include <stdio.h>
int main ()
{
  printf ("Hello world!");
  return 0;
}
```

Когда используется API в стиле UNIX, вы можете использовать файловые дескрипторы stdin, stdout и stderr, определённые в **unistd.h**, для доступа соответственно к символьным I/O потокам стандартного входа, стандартного выход и стандартной ошибки. Файл **unistd.h** инсталлируется вместе с Nios II EDS в качестве части пакета библиотеки Си newlib.

### Общий доступ к устройствам с символьным режимом

Доступ к другим (не stdin, stdout или stderr) устройствам с символьным режимом также прост, как открытие и запись в файл. Код в примере 6-4 пишет сообщение в UART с именем uart1.

**Example 6–4.** Writing Characters to a UART

```
#include <stdio.h>
#include <string.h>

int main (void)
{
  char* msg = "hello world";
  FILE* fp;

  fp = fopen ("/dev/uart1", "w");
  if (fp!=NULL)
  {
    fprintf(fp, "%s",msg);
    fclose (fp);
  }
  return 0;
}
```

**Потоки C++**

Системы на базе HAL могут использовать C++ потоки API для управления файлами из C++.

**/dev/null**

Все системы имеют устройство **/dev/null**. Запись в **/dev/null** не имеет эффекта, а все данные теряются. Устройство **/dev/null** используется для сохранения I/O переадресации на стадии запуска системы. Это устройство прекрасно подходит для приложений, которые хотят принимать ненужные данные.

Это устройство – исключительно программная конструкция. Оно не относится ни к какому физическому устройству в системе.

**Упрощённая версия I/O в символьном режиме**

HAL предлагает несколько методов по уменьшению объёма кода драйверов устройств с символьным режимом. Подробнее в секции "Уменьшение размера кода" на странице 6-30.

**Функции записи Altera**

Функции записи Altera предлагают отдельные каналы для посылки записей и отладочной информации в устройства с символьным режимом, дополняющие stdout и stderr. Информация записи Altera может быть выведена в соответствии с несколькими состояниями. Записи Altera могут быть разрешены или запрещены независимо от любого обычного выхода stdio, усиливая средства отладки.

Когда записи Altera разрешены, ваша программа может выводить специальные сообщения в заданный порт с помощью вызова функций HAL. Порт записи, заданный в BSP, может быть устройством UART или JTAG UART. В своей конфигурации по умолчанию, записи Altera выводят загрузочные сообщения, отображающие каждый шаг процесса загрузки.

Избегайте использования устройства записи Altera в качестве устройства, использующего stdout или stderr. Если выход записи Altera посылает в stdout или stderr, то выход записи будет перемежаться с выходом stdout или stderr.

Несколько доступных опций записи контролируются символами предпроцессора Си. Вы можете также добавлять собственные сообщения записи.

Altera регистрирует изменения в поведении системы. Реализация записи разрабатывается насколько возможно просто, загружая символы прямо в регистры передачи. Это может иметь негативное влияние на характеристики программы.

Функции записи Altera условно компилируемы. Когда запись запрещена, они не влияют на размер кода или характеристики.

Уменьшенные драйверы устройств Altera не поддерживают записи Altera.

### *Разрешение записи Altera*

Nios II SBT имеет настройку для разрешения записи Altera. Эта настройка называется **hal.log_port**. Она похожа на **hal.stdout**, **hal.stdin** и **hal.stderr**. Для разрешения записи Altera, вам необходимо установить **hal.log_port** для устройства JTAG UART или UART. Настройка позволяет HAL посылать сообщения записи в заданные устройства, когда вызывается макрос записи.

Когда записи Altera разрешены, Nios II SBT определяет ALT_LOG_ENABLE в **public.mk** для разрешения сообщений записи. Инструмент сборки также устанавливает значения ALT_LOG_PORT_TYPE и ALT_LOG_PORT_BASE в файле **system.h** для указания на заданное устройство.

Когда записи Altera разрешены без специальных опций, HAL выводит загрузочные сообщения в выбранный порт. Для обычной программы, которая использует стандартный **alt_main.c** (такая как "Hello World"), появляются сообщения как в примере 6-5.

**Example 6–5.** Default Boot Logging Output

```
[crt0.S] Inst & Data Cache Initialized.
[crt0.S] Setting up stack and global pointers.
[crt0.S] Clearing BSS
[crt0.S] Calling alt_main.
[alt_main.c] Entering alt_main, calling alt_irq_init.
[alt_main.c] Done alt_irq_init, calling alt_os_init.
[alt_main.c] Done OS Init, calling alt_sem_create.
[alt_main.c] Calling alt_sys_init.
[alt_main.c] Done alt_sys_init.  Redirecting IO.
[alt_main.c] Calling C++ constructors.
[alt_main.c] Calling main.
[alt_exit.c] Entering _exit() function.
[alt_exit.c] Exit code from main was 0.
[alt_exit.c] Calling ALT_OS_STOP().
[alt_exit.c] Calling ALT_SIM_HALT().
[alt_exit.c] Spinning forever.
```

Операция записи в устройство записи Altera останавливается в ALT_LOG_PRINTF(), пока все символы не будут прочитаны из выходного буфера устройства записи Altera. Чтобы проследить за тем, что приложение Nios II завершило инициализацию, запустите команду **nios2-terminal** из командной строки Nios II, чтобы получить доступ к выходу записи Altera.

### *Специальные опции записи*

В дополнении к загрузочным сообщениям по умолчанию, имеются дополнительный опции записи Altera. Каждая опция контролируется символом Си предпроцессора. Подробное описание каждой опции приведено в табл. 6-3.

**Табл. 6-3.** Опции записи Altera (часть 1 из 2)

| **Имя** | **Описание** | |
|---|---|---|
| Записи системного такта | Назначение | Выводит сообщение обработчика прерываний системного такта через определённый интервал. Этим показывается остановка запуска системы. Интервал по умолчанию – 1 секунда. |
| | Символ предпроцессора | ALT_LOG_SYS_CLK_ON_FLAG_SETTING |
| | Модификаторы | Записи системного такта имеют два модификатора, дающих два различных способа задания интервала записи.<br>▪ ALT_LOG_SYS_CLK_INTERVAL – задаёт интервал записи в системных тактах. По умолчанию - *один такт в секунду*, что означает – 1 секунду.<br>▪ ALT_LOG_SYS_CLK_INTERVAL_MULTIPLIER – задаёт интервал записи в секундах. По умолчанию – 1. Когда вы модифицируете ALT_LOG_SYS_CLK_INTERVAL_MULTIPLIER, то пересчитывается ALT_LOG_SYS_CLK_INTERVAL. |
| | Пример выхода | System Clock On 0<br>System Clock On 1 |
| Запись эхо | Назначение | Всегда вызывается alt_write() (обычно всякий раз, когда символы посылаются в stdout), первые <*n*> символов возвращаются эхом в сообщении записи. Сообщение начинается со строки "Write Echo:". Число <*n*> символов задаётся в ALT_LOG_WRITE_ECHO_LEN. По умолчанию – это 15 символов. |
| | Символ предпроцессора | ALT_LOG_WRITE_ON_FLAG_SETTING |
| | Модификаторы | ALT_LOG_WRITE_ECHO_LEN – количество символов в эхо, по умолчанию – это 15 символов. |
| | Пример выхода | Write Echo: Hello from Nio |
| Записи запуска JTAG | Назначение | При инициализации драйвера JTAG UART, выводится строка с количеством символов в программном буфере передачи, следующая за содержимым контрольного регистра JTAG UART. Количество символов в строке, начинающейся с "SW CirBuf", может быть отрицательным, поскольку они считаются как (*"конечная точка"- "начальная точка"*) в циклическом буфере. За подробной информацией о полях контрольного регистра JTAG UART обратитесь к секции "Периферия интерфейсов с внешними устройствами" в руководстве пользователя IP встроенной периферией. |
| | Символ предпроцессора | ALT_LOG_JTAG_UART_STARTUP_INFO_ON_FLAG_SETTING |
| | Модификаторы | нет |
| | Пример выхода | JTAG Startup Info: SW CirBuf = 0, HW FIFO wspace=64<br>AC=0 WI=0 RI=0 WE=0 RE=1 |

**Табл. 6-3.** Опции записи Altera (часть 2 из 2)

| Имя | Описание | |
|---|---|---|
| Записи интервала JTAG | Назначение | Создаёт объекты тревоги для вывода той же информации от JTAG UART, что и при записи запуска JTAG, но через повторяющиеся интервалы. Интервал по умолчанию – 0,1 секунда, или 10 сообщений за секунду. |
| | Символ предпроцессора | ALT_LOG_JTAG_UART_ALARM_ON_FLAG_SETTING |
| | Модификаторы | Записи интервала JTAG имеют два модификатора, дающих два различных способа задания интервала записи.<br>▪ ALT_LOG_JTAG_UART_TICKS – задаёт интервал записи в системных тактах. По умолчанию - *такты в секунду*/ 10.<br>▪ ALT_LOG_JTAG_UART_TICKS_DIVISOR – задаёт количество записей в секунде. По умолчанию – 10. Когда вы модифицируете ALT_LOG_JTAG_UART_TICKS_DIVISOR, то пересчитывается ALT_LOG_JTAG_UART_TICKS. |
| | Пример выхода | JTAG Alarm: SW CirBuf = 0, HW FIFO wspace=45 AC=0 WI=0 RI=0 WE=0 RE=1 |
| Записи программы обработки прерываний (ISR) JTAG | Назначение | Выводит сообщение всегда, когда срабатывает прерывание близкого к пустому JTAG UART. |
| | Символ предпроцессора | ALT_LOG_JTAG_UART_ISR_ON_FLAG_SETTING |
| | Модификаторы | нет |
| | Пример выхода | JTAG IRQ: SW CirBuf = -20, HW FIFO wspace=64 AC=0 WI=1 RI=0 WE=1 RE=1 |
| Записи загрузки | Назначение | Выводит сообщение о протекании процесса загрузки. Запись загрузки включена по умолчанию при разрешении записи Altera. |
| | Символ предпроцессора | ALT_LOG_BOOT_ON_FLAG_SETTING |
| | Модификаторы | нет |
| | Пример выхода | Обратитесь к секции "Разрешение записи Altera" на странице 6-10. |

Установка флага предпроцессора в 1 разрешает соответствующую опцию. Любое значение, отличное от 1, запрещает опцию.

Некоторые опции имеют модификаторы, которые дополняют символы предпроцессора, контролируя детали работы этой опции. Например, модификаторы записи системного такта контролируют интервал записи. Модификаторы опций описаны в табл. 6-3. Модификаторы опций имеют смысл только, когда эта опция разрешена.

### *Уровни записи*

Дополнительный символ предпроцессора ALT_LOG_FLAGS может быть использован для некоторой группировки специальных опций записи. ALT_LOG_FLAGS реализует уровни записи, основываясь на их влиянии на характеристики. При высоких уровнях записи, опции записи Altera занимают больше процессорного времени. Уровни ALT_LOG_FLAGS определены в табл. 6-4.

**Табл. 6-4.** Уровни записи Altera

| Уровень записи | Запись (регистрация) |
|---|---|
| 0 | Запись загрузки (по умолчанию) |
| 1 | Уровень 0 плюс запись системного такта и запись запуска JTAG |
| 2 | Уровень 1 плюс запись интервала JTAG и запись эхо |
| 3 | Уровень 2 плюс запись JTAG ISR |
| -1 | Режим молчания – нет записи Altera |

Примечание к табл. 6-4.

Вы можете использовать уровень записи -1 для выключения записи без изменения размера кода. Код записи останется в исполняемом образе, что определено выбором других опций. Это подходит для того, чтобы вы могли включать и выключать выход записи без разрушения схемы распределения памяти.

Поскольку каждая опция записи контролируется независимым символом предпроцессора, отдельные опции в уровнях записи могут быть подменены.

### *Пример создания BSP с записью*

В примере 6-6 создаётся HAL BSP с разрешёнными записями Altera и следующими опциями в дополнении к записи загрузки по умолчанию:

- Запись системного такта
- Запись запуска JTAG
- Запись интервала JTAG, регистрируется дважды в секунду
- Эхо не пишется

**Example 6–6.** BSP With Logging

```
nios2-bsp hal my_bsp ../my_hardware.sopcinfo \
  --set hal.log_port uart1 \
  --set hal.make.bsp_cflags_user_flags \
  -DALT_LOG_FLAGS=2 \
  -DALT_LOG_WRITE_ON_FLAG_SETTING=0 \
  -DALT_LOG_JTAG_UART_TICKS_DIVISOR=2↵
```

Аргумент -DALT_LOG_FLAGS=2 добавляет -DALT_LOG_FLAGS=2 к ALT_CPP_FLAGS, создавая переменную в **public.mk**.

### *Собственные сообщения записи*

Вы можете добавить собственные сообщения записи, которые будут посылаться в устройства записи Altera. Для определения возможности собственных сообщений, включите заголовочный файл **alt_log_printf.h** в ваш исходный файл Си:

#include "sys/alt_log_printf.h"

Затем используйте следующую макро функцию:

ALT_LOG_PRINTF(const char *format, ...)

Этот макрос предпроцессора Си является сокращённой версией printf(). Формат аргумента поддерживает большинство опций printf(). Он поддерживает: %c, %d %I %o %s %u %x и %X, а также с некоторой точностью модификаторы интервалов, такие как %-9.3o. Он не поддерживает форматы с плавающей точкой, такие как %f или %g. Эта макро функция не компилируется, если не разрешены записи Altera.

Если вы хотите, чтобы ваши собственные сообщения записи контролировались опциями предпроцессора записи Altera, используйте соответствующую опцию флага предпроцессора из табл. 6-4 или 6-3 на стр. 6-11. В примере 6-7 показаны два способа реализации опций записи вместе с собственными сообщениями записи.

**Example 6–7.** Using Preprocessor Flags

```
/* The following example prints "Level 2 logging message" if
   logging is set to level 2 or higher */
#if ( ALT_LOG_FLAGS >= 2 )
    ALT_LOG_PRINTF ( "Level 2 logging message" );
#endif

/* The following example prints "Boot logging message" if boot logging
   is turned on */
#if ( ALT_LOG_BOOT_ON_FLAG_SETTING == 1)
    ALT_LOG_PRINTF ( "Boot logging message" );
#endif
```

***Файлы записи Altera***

В табл. 6-5 перечислены исходные файлы HAL, в которых реализованы функции записи Altera.

**Табл. 6-5.** HAL файлы реализации функций записи Altera

| **Размещение (1)** | **Имя файла** |
|---|---|
| components/altera_hal/HAL/inc/sys/ | alt_log_printf.h |
| components/altera_hal/HAL/src/ | alt_log_printf.c |
| components/altera_nios2/HAL/src/ | alt_log_macro.S |

Примечание к табл. 6-5

(1) Все файлы находятся относительно <*Nios II EDS install path*>.

В табл. 6-6 перечислены исходные файлы HAL, использующие функции записи Altera. Эти файлы реализуют опции записи, перечисленные в табл. 6-3 на стр. 6-11. Они служат примером использования записи.

**Табл. 6-6.** HAL файлы - примеры использования записи Altera

| Размещение (1) | Имя файла |
|---|---|
| components/altera_avalon_jtag_uart/HAL/src/ | altera_avalon_jtag_uart.c |
| components/altera_avalon_timer/HAL/src/ | altera_avalon_timer_sc.c |
| components/altera_hal/HAL/src/ | alt_exit.c |
| components/altera_hal/HAL/src/ | alt_main.c |
| components/altera_hal/HAL/src/ | alt_write.c |
| components/altera_nios2/HAL/src/ | crt0.S |

Примечание к табл. 6-6

(1) Все файлы находятся относительно <*Nios II EDS install path*>.