Интерфейс в стиле UNIX

HAL API предлагает несколько функций в стиле UNIX. Функции в стиле UNIX предлагают знакомую среду разработки для новичков в программировании под Nios II, и может упростить задачи переноса существующего кода для запуска в среде HAL. HAL использует эти функции изначально для предоставления системного интерфейса под стандартную библиотеку ANSI Си. Например, функциям, выполняющим доступ к устройству, необходимы функции библиотеки Си, определённые в **stdio.h**.

Следующий список содержит все необходимые функции в стиле UNIX:

- _exit()
- close()

- fstat()
- getpid()
- gettimeofday()
- ioctl()
- isatty()
- kill()
- lseek()
- open()
- read()
- sbrk()
- settimeofday()
- stat()
- usleep()
- wait()
- write()

Большинство часто используемых функций относятся к I/O файлам. Посмотрите секцию "Файловая система" на странице 6-6.

Подробнее об использовании этих функций, обратитесь к главе "Справочник по HAL API" в настольной книге программиста под Nios II.

Файловая система

HAL предлагает инфраструктуру для доступа к файлам в стиле UNIX. Вы можете использовать эту инфраструктуру для сборки файловой системы в любом устройстве памяти, доступном в вашем аппаратном проекте.

За примером обратитесь к главе "Доступная только для чтения файловая система Zip" в настольной книге программиста под Nios II.

Вы можете обращаться к файлам в файловой системе на основе HAL, используя либо функции для I/O файлов стандартной библиотеки Си (например, fopen(), fclose() и fread()), либо функции для I/O файлов в стиле UNIX, предлагаемые HAL.

HAL предлагает следующие функции стиле UNIX для управления файлами:

- close()
- fstat()
- ioctl()
- isatty()
- lseek()
- open()
- read()
- stat()
- write()

За подробной информацией об этих функциях, обратитесь к главе "Справочник по HAL API" в настольной книге программиста под Nios II.

HAL регистрирует файловую подсистему в качестве верхней точки глобальной файловой системы HAL. Попытка доступа к файлам ниже верхней точки перенаправляется прямо к файловой подсистеме. Например, если доступная только для чтения файловая подсистема zip (**zipfs**) установлена как **/mount/zipfs0**, то файловая подсистема **zipfs** обрабатывает вызовы fopen() для **/mount/zipfs0/myfile**.

Нет понятия *текущая директория*. Программа должна обращаться ко всем файлам, используя абсолютный путь.

Файловая инфраструктура HAL также позволяет вам управлять устройствами с символьным режимом с помощью имён путей в стиле UNIX. HAL регистрирует устройства с символьным режимом в качестве узлов файловой системы HAL. Условно, **system.h** определяет имя узла устройства в виде префикса **/dev/** плюс имя компонента, названного в SOPC builder. Например, периферия UART **uart1** в SOPC builder называется **/dev/uart1** в **system.h**.

Код в примере 6-2 читает символы из файловой подсистемы zip, доступной только для чтения, **rozipfs**, которая зарегистрирована в качестве узла файловой подсистемы HAL. Стандартные заголовочные файлы: **stdio.h**, **stddef.h** и **stdlib.h** устанавливаются вместе с HAL.

Example 6–2. Reading Characters from a File Subsystem

```
#include <stdio.h>
#include <stddef.h>
#include <stdlib.h>

#define BUF_SIZE (10)

int main(void)
{
  FILE* fp;
  char buffer[BUF_SIZE];

  fp = fopen ("/mount/rozipfs/test", "r");  if (fp == NULL)
  {
    printf ("Cannot open file.\n");
    exit (1);
  }

  fread (buffer, BUF_SIZE, 1, fp);

  fclose (fp);

  return 0;
}
```

За подробной информацией об использовании этих функций, обратитесь к документации на библиотеку Си newlib, установленной вместе с Nios II EDS. В меню **Пуск** в Windows кликните **Programs** > **Altera** > **Nios II** > **Nios II Documentation**.