Математические структуры и моделирование
2016. № 1(37). С. 97–102

УДК 004.424.8

РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ДЛЯ РАБОТЫ С ПРОГРАММАТОРАМИ

Д.Н. Лавров[1]
к.т.н., e-mail: dmitry.lavrov72@gmail.com
А.Ю. Воробьёв[2]
инженер-программист, e-mail: flyenj@gmail.com

[1]Факультет компьютерных наук, ОмГУ

[2]АО «Омский научно-исследовательский приборостроения»

Аннотация. В данной статье рассматриваются способы и приёмы разработки программного обеспечения для управления программаторами под операционные системы Windows и Linux. Каждый способ описывается подробно, и учитываются тонкости работы на каждой ОС. Представленное решение для ОС Windows может быть реализовано на любом языке, который может вызывать прикладной программный интерфейс этой ОС, например, C/C++ или Delphi. Полученные решения могут затем использоваться для создания кроссплатформенного ПО на Java с помощью технологии Java Native Interface.

Ключевые слова: программатор, usb flash, программирование, windows IO, Linux IO.

В настоящее время широко используются устройства, программирующие микросхемы ПЗУ, ПЛИС и микроконтроллеры. Такие устройства называются программаторами. Программатор — это аппаратно-программное устройство, предназначенное для записи и считывания информации в однократно записываемое постоянное запоминающее устройство [1]. Программатор обычно представляет собой внешнее устройство, подключаемое к компьютеру по интерфейсу USB. На компьютере это устройство идентифицируется как USB Mass Storage Device и может быть открыто как файл для чтения и записи.

В нашей организации ведётся разработка автоматизированного рабочего места (АРМ) для настройки и проверки кварцевых генераторов. Управление прожигом и мониторинг состояния кварцевого генератора осуществляется через программатор и внешний частотомер. Настройка осуществляется следующим образом. В оперативную память устройства через программатор записываются коэффициенты полинома вместе с некоторыми другими, такими как значение амплитуды, делителем частоты, типом входного буфера и т.д. Затем с помощью камеры тепла и холода изменяется температура и измеряется температурно-частотная характеристика. Если она соответствует норме, то подобранные коэффициенты прожигаются в постоянную память с помощью программатора.

Для создания такого АРМа необходимо создать модуль управления программатором. Требования к модулю следующие:

1) кроссплатформенность;
2) отсутствие проприетарных компонент сторонних производителей в коде;
3) возможность повторного использования кода;
4) так как основной код АРМа реализуется на Java, то должен быть обеспечен доступ к модулю через Java Native Interface (JNI).

В данной статье описывается реализация модуля для операционных систем Windows и Linux.

Разработка ПО для программатора на ОС Windows

Для того чтобы разработать ПО для управления программатором на ОС Windows, можно воспользоваться прикладным программным интерфейсом этой ОС, называемым WinAPI. Для взаимодействия с устройством достаточно следующих функций [2]:

1. CreateFile. Данная функция открывает устройство как файл для чтения и записи. В качестве имени устройства передаётся строка "////.//PhysicalDrive t где t представляет собой номер от 1. Следует различать 2 версии этой функции, а именно CreateFileAи CreateFileW. С помощью препроцессора, в зависимости от текущей кодировки, будет выбрана одна из этих версий. Первая поддерживает кодировку windows-1251, а вторая UTF-8. Каждая из этих версий может быть вызвана явно.

2. SetFilePointer. Устанавливает указатель для считывания и записи.

3. ReadFile. Считывает указанное количество байт в буфер.

4. WriteFile. Записывает указанное количество байт на устройство.

При использовании функции CreateFile необходимо указать некоторые параметры. Для взаимодействия с программатором это обычно «deviceName, GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL, OPEN_EXISTING, FILE_ATTRIBUTE_NORMAL, 0». Данные параметры позволяют получить необходимый уровень доступа к устройству.

После того как было найдено устройство, необходимо установить указатель для считывания с помощью функции SetFilePointer. У программатора обычно присутствует несколько секторов фиксированного размера. Для того чтобы считать определённый сектор, достаточно передать функции следующие параметры «hDevice, (SECTOR_NUMBER - 1) * SECTOR_SIZE, NULL, FILE_BEGIN», где hDevice устройство, SECTOR_NUMBER и SECTOR_SIZE соответственно номер и размер сектора.

Затем можно приступить к считыванию с помощью функции ReadFile. Перед этим необходимо создать два буфера – для чтения и записи. Функция

вызывается со следующими параметрами «hDevice, bufferRead, SECTOR_SIZE, nb, NULL», где bufferRead – это буфер для чтения с размером, равным SECTOR_SIZE – 1, а nb – это переменная, в которую будет записано количество считанных байт. После считывания необходимо скопировать буфер чтения в буфер для записи. Перед повторным считыванием указатель должен быть выставлен повторно.

Для записи необходимо установить указатель в начало записываемого сектора, а затем, с помощью функции WriteFile с параметрами «hDevice, bufferWrite, SECTOR_SIZE, nb, NULL» осуществить запись. На записываемом секторе обычно присутствуют места в памяти, запись в которые представляют собой команды для программатора. Таким образом, можно осуществлять взаимодействие с программатором на ОС Windows. Пример кода для соединения с устройством под Windows:

```
bool connectToDevice()
{
  char deviceName[19] = "\\\\.\\PhysicalDrive ";
  for (int i = 1; i < 6; ++i) {
    deviceName[17] = i + '0';
      hDevice = CreateFileA(
        deviceName,
          GENERIC_READ | GENERIC_WRITE,
          FILE_SHARE_READ | FILE_SHARE_WRITE,
          NULL,
          OPEN_EXISTING,
          FILE_ATTRIBUTE_NORMAL,
          0
      );
      if (hDevice != INVALID_HANDLE_VALUE) {
        std::cout << "Some Device found!" << std::endl;
        SetFilePointer(hDevice, (SECTOR_NUMBER - 1) *
             SECTOR_SIZE, NULL, FILE_BEGIN);
        ReadFile(hDevice, bufferRead, SECTOR_SIZE, nb, NULL);
        for (int i = 0; i < sizeof(bufferRead); ++i) {
          bufferWrite[i] = bufferRead[i];
        }
        return testSignature(bufferRead, signature);
      }
      std::cout << "Can't find device!" << std::endl;
  }
  return false;
}
```

Разработка ПО для программатора на ОС Linux

Для того чтобы разработать ПО управления программатором на ОС Linux, можно воспользоваться прикладным программным интерфейсом этой ОС. Для

взаимодействия с устройством достаточно следующих функций [3]:

1) open. Осуществляет соединение с устройством. Она определена в заголовочном файле "fcntl.h";
2) lseek. Устанавливает указатель для считывания и записи;
3) read. Считывает указанное количество байт;
4) write. Осуществляет запись.

Как можно заметить, данные функции очень похожи на функции WinAPI.

Для соединения с блочным устройством обычно указываются следующие параметры: «deviceName, O_RDWR, O_SYNC, O_DIRECT». Флаг «O_RDWR» позволяет осуществлять чтение и запись. А флаги O_SYNC и O_DIRECT предоставляют прямой синхронный доступ к устройству.

Аналогично, перед тем как считывать какую-либо информацию необходимо установить указатель для чтения. Для этого используется функция lseek с параметрами: «hDevice, (SECTOR_NUMBER - 1) * SECTOR_SIZE, SEEK_SET», где SEEK_SET означает начало файла. Чтение и запись работают аналогично функциям в WinAPI. Все они определены в заголовочном файле «unistd.h».

Этот способ работает только в том случае, если у нас в системе установлен подходящий драйвер, через который возможно взаимодействие. В случае если таковой отсутствует, можно обратиться напрямую к устройству через протокол UAS[4] и найти то место, в которое необходимо осуществлять чтение и запись.

Стыковка C++ и Java

Для стыковки кода, написанного на C или C++ можно воспользоваться готовым генератором, который автоматически генерирует стыковочный код по заданному интерфейсу. Или написать этот код самостоятельно, используя функции из Java Native Interface. Далее будет описан пример создания стыковки с помощью генератора под названием «Simplified Wrapper and Interface Generator» (SWIG).

Предположим, что имеется исходный файл на C. Для его стыковки с Java сначала необходимо создать файл-интерфейс с расширением «i». В этом файле указывается та информация, которая будет доступна из Java, а именно функции, классы и т.д. Пример файла-интерфейса для программатора:

```
%module Programmator
%{
#include "programmator.h"
%}

struct Coeff {
  int inf, lin, scale, cub, four, fifth,
  vc, sc, ampl, fch, div, cf, cc, offset;
};

char* test();
```

```
bool connectToDevice();
bool isConnected();
bool setUc(int uc);
bool powerOn();
bool powerOff();
const char* readBurnedValues();
const char* readCurrentValues();
int send(struct Coeff coeff);
int burn(struct Coeff coeff);
const char* readKs();
```

После создания интерфейса необходимо сгенерировать стыковочный файл. Для этого можно воспользоваться следующей командой

`swig -c++ -java -package mypackage.wrapper interface_file.i`,

где `mypackage.wrapper` — это название пакетов, в которых будут располагаться сгенерированные исходные файлы на Java, а `interface_file.i` — это название файла-интерфейса. Следует уделить особое внимание пакетам, в которых будут располагаться исходные файлы на Java, так после их генерирования нельзя перемещать эти файлы в другие пакеты.

После выполнения такой команды будет сгенерирован стыковочный файл на C++ и исходные файлы на Java. Перед компиляцией необходимо подключить заголовочные файлы из Java Development Kit, а именно `/jdk/include` и `/jdk/include/operating_system`, где `operating_system` — название операционной системы, под которую делается стыковка. После выполнения всех описанных выше действий оба C++ файла скомпилируются в динамическую библиотеку. Для работы стыковки необходимо, во-первых, разместить динамическую библиотеку в `java.library.path` или в системную переменную `PATH`. Во-вторых, в Java коде сделать вызов `System.loadLibrary("libraryName")`, где `libraryName` — это название динамической библиотеки без указания её расширения.

Заключение

В результате проделанной работы было получено описание взаимодействия с программаторами, которые реализованы в виде USB flash. Путём инкапсуляции низкоуровневой логики была получена возможность на программном уровне абстрагироваться от особенностей реализации аппаратной части. Таким образом, код, написанный на C или C++ можно использовать в объектном стиле на Java. Благодаря этому от разработчика, пользователя, сгенерированного Java кода скрывается сложность низкоуровневой работы по USB с программатором.

Полученные результаты могут применяться для взаимодействия с любым устройством, которое поддерживает эти функции. Разработанный модуль может быть обобщён и использован как программная библиотека для высокоуровневых языков, таких как Java. Это позволит разрабатывать подобные программы только с помощью высокоуровневых языков. Тем самым ускоряя разработку и уменьшая количество возможных ошибок.

Литература

1. Гелль П. Как превратить компьютер в универсальный программатор. ДМК Пресс, 1992.
2. File Management Functions. URL: `https://msdn.microsoft.com/en-us/library/windows/desktop/aa364232(v=vs.85).aspx` (дата обращения 15.04.2015).
3. The Open Group Base Specifications Issue 7 // IEEE Std 1003.1, 2013 Edition. URL: `http://pubs.opengroup.org/onlinepubs/9699919799/functions/V2_chap02.html` (дата обращения 15.04.2015).
4. USB Attached SCSI (UAS). URL: `http://www.t10.org/members/w_uas-.htm` (дата обращения 21.04.2015).

SOFTWARE DEVELOPMENT FOR CHIP PROGRAMMER

D.N. Lavrov[1]
Ph.D., e-mail: dmitry.lavrov72@gmail.com
A.Y. Vorobyev[2]
Software Engineer, e-mail: flyenj@gmail.com

[1]Omsk State University n.a. F.M. Dostoevskiy
[2]Omsk Institute of Instrument

Abstract. . This article discusses the methods and techniques of software development for controlling chip programmers for Windows and Linux operating systems. Each method is described in detail and takes into account subtleties for each OS. The presented solution for Windows can be implemented in any language that can call the application programming interface of the OS, such as C/C++ or Delphi. The resulting solution can then be used to create cross-platform software in Java with the help of Java Native Interface technology.

Keywords: chip programmer, usb flash, programming, windows IO, linux IO.