USB загрузчик для отечественных микроконтроллеров MDR32F9Q2I с использованием программы 1986wsd.

Петров А. А.[1], Будин Д. И.[2]

[1]Петров Алексей Александрович / Petrov Alexey Aleksandrovich – студент;
[2]Будин Дмитрий Иванович / Budin Dmitry Ivanovich – студент,
кафедра систем автоматического управления и контроля, факультет интеллектуальных технических систем,
Национальный исследовательский университет,
Московский институт электронной техники, г. Зеленоград

Аннотация: *в данной статье рассматривается альтернативный вариант создания USB загрузчика на основе программы 1986wsd. Основные преимущества и недостатки по сравнению со стандартным загрузчиком от компании производителя.*

Ключевые слова: *USB, прошивка, загрузчик, микроконтроллер, программа 1986wsd, MDR32F9Q2I, программатор.*

Прошивка – это встроенное программное обеспечение, недоступное пользователю, взаимодействующая с аппаратной частью устройства, позволяющая использовать различные программы и выполнять различные функции.

Одним из самых простых способов загрузки прошивки в память микроконтроллера, является использование программатора. Данный метод прост и удобен. Но в связи с тем, что программаторы - удовольствие не из дешевых, приходится искать выход. Одним из таких решений является использование программы 1986WSD, которую можно скачать на официальном форуме компании Milandr. Данная программа использует стандартный UART загрузчик. Прошивка загружается в микроконтроллер через интерфейс uart. Тоже неплохое решение, но если вы сами решили проектировать свою плату на основе микроконтроллера MDR32F9Q2I и создавать специально для прошивки пины для программатора - встраивать UART интерфейс просто нет необходимости? Но есть usb вход. Микро-usb или мини - разницы не имеет. Об этом и пойдет дальнейшая речь.

Во время создания данного проекта использовалась отладочная плата LDM-K1986BE92QI, программа 1986WSD, готовый проект VCOM_Echo, поставляемый на диске производителем отладочной платы.

Производителями данного микроконтроллера предусмотрен стандартный протокол загрузки прошивки. Он описан в документации. Рассмотрим подробнее передачу данных от программы 1986wsd к микроконтроллеру, и что в ответ должен послать микроконтроллер. На рисунке 1 изображено окно программы 1986wsd.

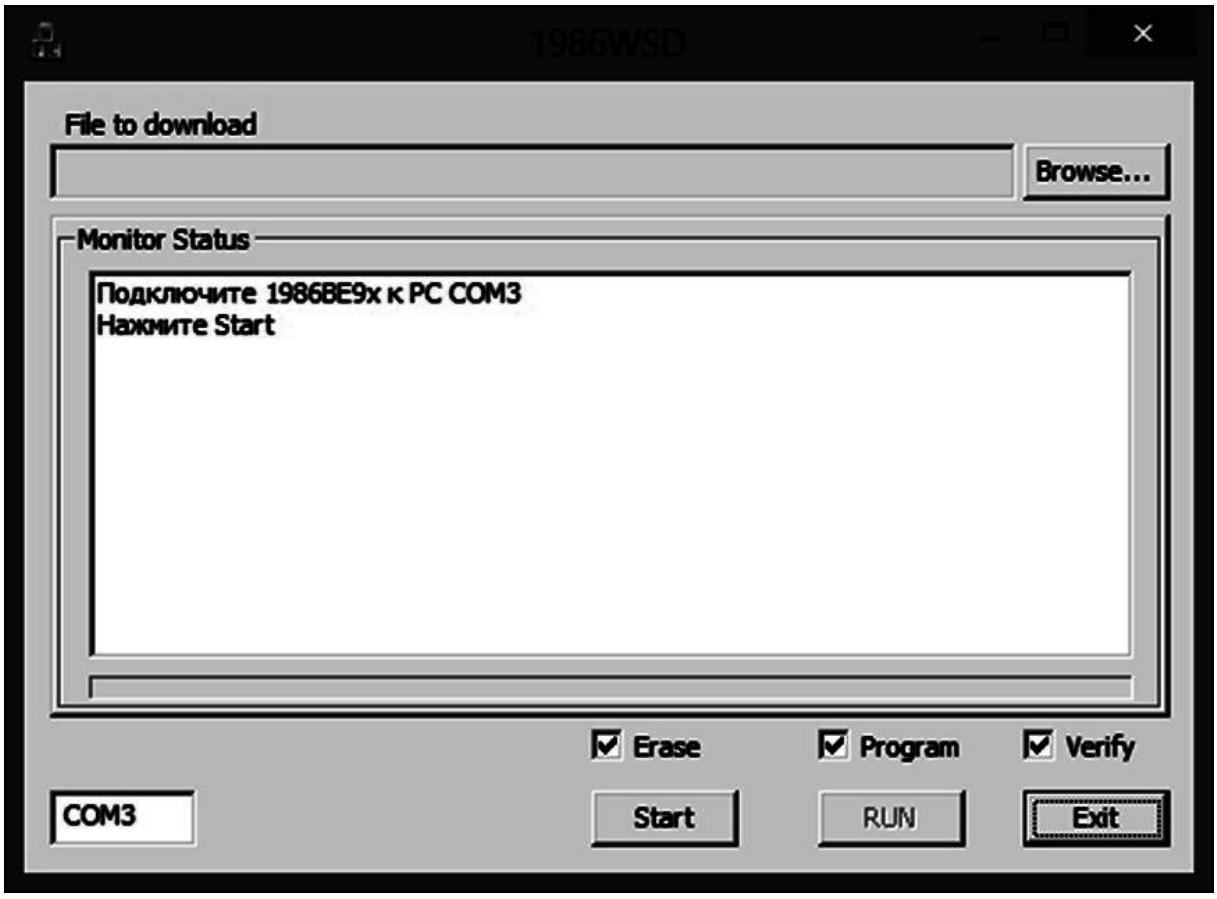

Рис.1. Программа 1986WSD

Интерфейс программы прост и очевиден. В полном объяснении не нуждается. Стоит лишь пояснить, что программа передает данные по указанному COM порту.

1. Первым сообщением передается синхросимвол 0x00. Служащий для синхронизации с нашим микроконтроллером. Символ передается порядка 1024 раз. Никоим образом на него отвечать не нужно.
2. Сразу же после синхросимвола передается команда 0x0D, на которую микроконтроллер должен ответить массивом из трех элементов это 0x0D, 0x0A, 0x3E.
3. Третья посылка от программы состоит из 9 байт, и состоит она из следующих чисел: 0x4C, 0x00, 0x78, 0x00, 0x20, 0xE3, 0x03, 0x00, 0x00. Микроконтроллер же должен отправить 0x4C. После взаимной передачи данных программа передает прошивку.
4. Передается прошивка, которую мы успешно указали в программе. Передается одним файлом. С интерфейсом uart все будет работать хорошо. Но с USB вы столкнетесь с проблемой. Часть данных будет просто напросто потеряна.
5. Проверка переданной прошивки. Команда состоит из 9 байт. Рассмотрим ее как массив данных. Первым элементом данного массива является 0x59, следующие 8 байт - это адрес таблицы векторов загруженной программы [1]. Т. е. мы получаем адрес. В нашу задачу входить найти по этому адресу соответствующие переменные и отправить их программе. На данную команду микроконтроллер должен отправить уже массив, состоящий из 10 байт. Первый элементом массива является символ 0x59, а последним 0x4B. Остальные же элементы содержат информацию по указанному адресу.
6. Следующей командой является команда 0x52. Запуск программы на выполнение. И содержащий параметр адрес таблицы векторов [1]. Микроконтроллер просто обязан отправить в ответ число 0x52.
7. Программа посылает число 0x49. На которое микроконтроллер должен отправить массив из байт: 0x31, 0x39, 0x38, 0x36, 0x42, 0x4F, 0x4F, 0x54, 0x55, 0x41, 0x52, 0x54, 0x1C. Данный массив в кодировке ASCII означает 1986BOOTUART.

А теперь рассмотрим программу 1986wsd подробнее. Все исходники программы имеются на официальном сайте производителя микроконтроллера. Все, что нам необходимо изменить, для того чтобы мы могли осуществить загрузку данных через USB, это внести изменения в файл 1986WSDDlg.cpp. Находим в этом файле функцию Start(). В теле программы введем следующие изменения:

1. Добавим переменную EOT (End of transission) типа Byte и присвоим ей значение N. Она необходима для того, чтобы сообщить микроконтроллеру о конце передачи данных о прошивке.

```
Byte Eot = 'N'.
```

2. Находим следующий участок кода.

```
if      ((!com.ReadBlock(rxdbuf,1))||(rxdbuf[0]!='L'))
{
str = "ошибка обмена";
m_list.InsertString(m_list.GetCount(),str);
com.Close();
return;}
```

И сразу же после него нам необходимо прописать следующее.

```
int l = 0, indFE = 0, indEE = 0;
for (int k = 0; k < 32768; k++)
{
if (bufram[k] != 0xff)
{
indFE = k;
break;
}
}
// First index of element which not equal 0xff
for (int k = 32767; k > indFE; k--){
if (bufram[k] != 0xff)
{
indEE = k;
break;
}
}
while (indFE < indEE)
{
buffArr[0] = 'H';
for (int k = 0; k < 31; k++)
{
buffArr[k + 1] = bufram[indFE + k];
}
com.WriteBlock((LPSTR)(buffArr), 32);
indFE += 31;

}
com.WriteBlock((LPSTR)eot, 1);
```

Этот участок кода позволяет отправлять данные прошивки по 31 байту. 1 байта занимает символ «H», который сообщает микроконтроллеру о начале прошивки. Но изменений одной лишь программы 1986 WSD недостаточно, чтобы можно было получить прошивку по частям. Для этого также требуется внести изменения в прошивку микроконтроллера. Первоначально добавив в файл main.c функцию, которая передает и принимает данные в соответствии с вышеописанным протоколом.

```
USB_Result Protocol_Of_Transmit_Data(uint8_t* Buffer, uint32_t Length)
{
// Объявление всех необходимых переменных. Можно сделать их глобальными
uint8_t zero=0x00, countByte=0, sync_sym=0x01, buffer=0xFF, buadRateCorrect=0xFF,
id_str[13]={0x31,0x39,0x38,0x36,0x42,0x4F,0x4F,0x54,0x55,0x41,0x52,0x54,0x1C}, // 1986BOOTUART
cmdLoadArr[]= {0x59, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x4B},
buadRate[] = {0x42, 0x00, 0xC2, 0x01, 0x00},
USB_Result result;
switch(Buffer[0])
{
case 0x48:
for(countByte=0;countByte<31;countByte++)
```

```
{
Firmware[count]=Buffer[countByte+1];
count++;
}
result= USB_SUCCESS;
break;
case 0x00:
if(count<1021)
{
count+=1;
result=USB_SUCCESS;
}
else
{
result = USB_CDC_SendData(&zero,1);
}
break;
case 0x59:
for(countByte=1;countByte<9;countByte++)
{
cmdLoadArr[countByte]=Firmware[fCount];
fCount++; // fCount глобальная переменная типа uint32_t;
}
USB_CDC_SendData(cmdLoadArr, 10);
break;
case 0x42:
result = USB_CDC_SendData(&buadRateCorrect, 1);
break;
case 0x52:
zero=0x52;
result=USB_CDC_SendData(&zero, 1);
break;
case 0x0D:
if(flag==0)
{
count=0;
result = USB_CDC_SendData(cmdInv,3);
}
break;
case 0x4C:
result = USB_CDC_SendData(errorArr,1);
break;
case 0x4E:
result= USB_CDC_SendData(&errorArr[1],1);
break;
case 0x49:
result= USB_CDC_SendData(id_str,13);
break;
}
return result;}
```

После того, как все заготовки выполнены, необходимо загрузить прошивку в микроконтроллер. Запустить программу и все. USB загрузчик готов.

Основным плюсом данного проекта является то, что вам уже самим лично не придется разбираться в коде программ и вносить в них изменения. Все сделано за вас. Вам лишь придется разобраться в этой статье.

Литература

1. Спецификация. Серия 1986BE9x, К1986BE9x, К1986BE92QI, К1986BE92QC, К1986BE91H4, высокопроизводительных 32-разрядных микроконтроллеров на базе процессорного ядра ARM Cortex-M3.