AT90USBKey – Basic Programming Configuration for MS Windows Environments
(By Alexander Hoole, v0.1, September 2006)

Table of Contents

Introduction 1
Hardware summary 1
Getting started 2
Setup for programming 2
    Installing WinAVR 3
    Installing AVR Studio 4 3
    Installing FLIP 3
Final Preparation 4
Creating your first AVR Studio project 6
    Adding the source 6
    Building the project 8
Loading your first program on the key using FLIP 8
Closing Remarks 8

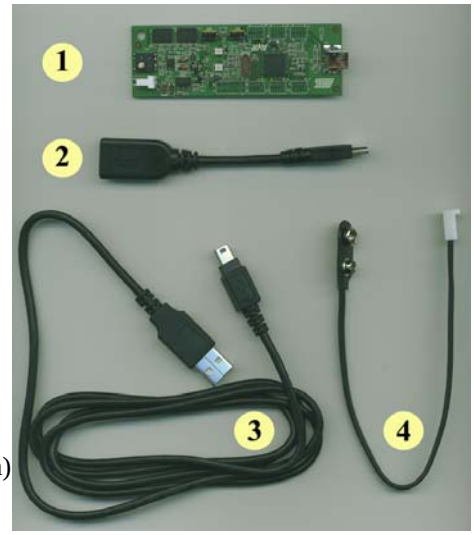
Introduction

This short document is intended to give a quick walkthrough of the tools and basic steps required to start working with the ATMEL AT90USBKey. There is a host of programs, tools and examples available, however, there did not seem to be any sort of documentation that gave a fast introduction to what is needed to get things up and running. Hopefully you will find this collection of steps helpful in getting your project up and going.

Hardware summary

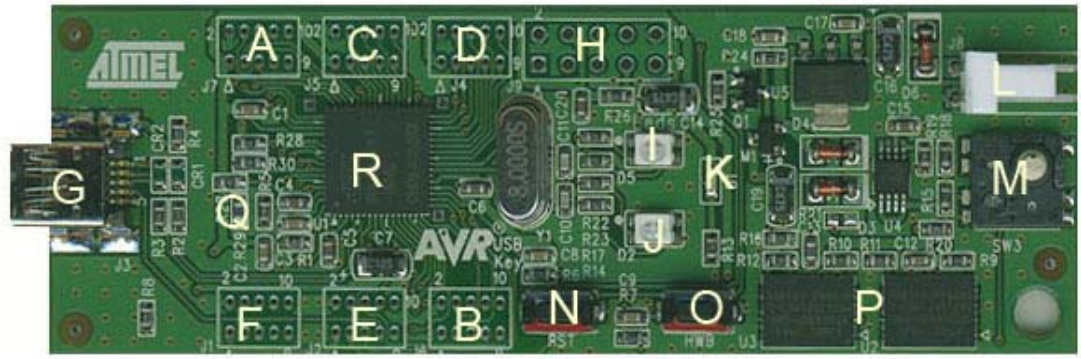
After ordering your own ATUSBKey you will probably be wondering what comes in the box. The AT90USBKey Demo-Kit contains the following components:

1. **AVR USB Key**
    - USB On-The-Go Microcontroller
    - 128KB Flash Program Memory
    - 8KB SRAM
    - 4KB EEPROM
    - 8 Channel 10-bit A/D-converter
    - JTAG interface for on-chip-debug
    - Up to 16MIPS throughput at 16MHz
    - 2.7 – 5.5 Volt operation
    - USB full/low speed device/host/OTG interface
    - 4+1-ways joystick
    - 2 Bi-Color LEDs
    - Temperature sensor
    - System clock: 8 MHz crystal
    - Onboard reset button
    - Onboard HWD button (force bootloader section execution)
2. **Host Adapter**
3. **USB Cable**
4. **Battery Clip**



The cables are pretty basic. Use (3) to connect your key to the development platform, and use (4) to connect the key to a power supply once you are done programming.

The USB key, however, is more complicated and deserves to have its key components spelled out. The AVR USB Key has the following layout:



LABEL	Description
A-F	Ports A through F
G	USB MiniAB connector
H	JTAG port
I-J	Bi-colored LEDs (Red/Green LEDs)
K	Power LED
L	External power connector
M	4+1 way joystick (four directions + push button)
N	On board reset button
O	On board HWB button to force bootloader section execution on reset
P	Data flash memory
Q	Temperature sensor
R	AT90USB1287

Note: For additional information refer to the "AT90USBKey Hardware User Guide".

While the kit does not contain connectors for ports [A-F] or the JTAG port, you will hopefully be able to obtain them during the course.

In this short tutorial we will make use of the following features:

- Use (G) to connect to our development machine
- Use (N) and (O) to change the operating mode of the key and to run our software once it is uploaded onto the key
- Use our software to have (I) and (J) LEDs light up.

Getting started

First, attach the USB key (1) to the USB port of your PC/Laptop using the USB cable (3). Once the USB key is attached, the operating system should detect the presence of a new device and grant access to the USB key via a new drive letter on your machine. You should start by opening this new drive and then double-click on *index.html* to start learning the new device.

Setup for programming

To program our AT90USBKey we will need a combination of software. The following setup was performed on a Intel based Windows XP Professional machine.

In general, the software available for programming the AT90USBKey is listed at the following URL: http://www.atmel.com/dyn/products/product_card.asp?part_id=3875. We will make use of WinAVR, which provides the GCC compiler that we will use; AVR Studio 4, which provides the integrated development environment we will develop on; and FLIP 3, which will be used to load our new programs onto the USB key. In the following section we will quickly follow the necessary steps to configure the development machine.

Installing WinAVR

Download and install WinAVR from the following URL: http://winavr.sourceforge.net.

Note: Available for all 32-bit MS Windows (95/98/NT/2000/XP).

Installing AVR Studio 4

Download and install AVR Studio 4 and Service Pack 3 from the following URL: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=2725. Make sure you choose the "Install/upgrade USB Driver" option if you plan to use JTAG later on.

Note: Available for MS Windows 95/98/NT/2000/XP.

Installing FLIP

Download and install FLIP 3 from the following URL: http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886. Once completed you will need to reboot your machine.

FLIP stands for Flexible In-system Programmer and it allows us to program all of the FLASH C51 parts. During installation the following list of features is provided is provided:

- Performs In-System Programming through RS232, USB or CAN interfaces.
- May be used through its intuitive Graphical User Interface or launched from a DOS window (see the batchisp manual), from an embedded software IDE like KEIL's uVision2, or even from your own application (see the ISP Functions Library manual).
- Runs under Windows 9x / Me / NT / 2000 / XP.
- Supports Intel MCS-86 Hexadecimal Object, Code 88 file format for data file loading and saving.
- Buffer editing capabilities: fill, search, copy, reset, modify, goto.
- Target device memory control: erase, blank check, program, verify, read, security level and special bytes reading and setting.
- Parts serialization capability (from batchisp only).
- AutoISP allows ISP hardware conditions to be set by software.
- The demo mode emulates ISP operations without any target hardware.
- The debug mode lets you log the traffic on the selected medium.
- On-line help.

Final Preparation

When you are ready to start programming the device you will need to do a hard reset of the AT90USBKey. To do a hard reset of the key do the following:

- Press both "RST" and "HWB" push buttons down simultaneously
- First release the "RST" push button
- Then release the "HWB" push button

When this is done one of the following will likely occur:

a) You will be prompted for drivers for the new device.
b) The system will already support the device.

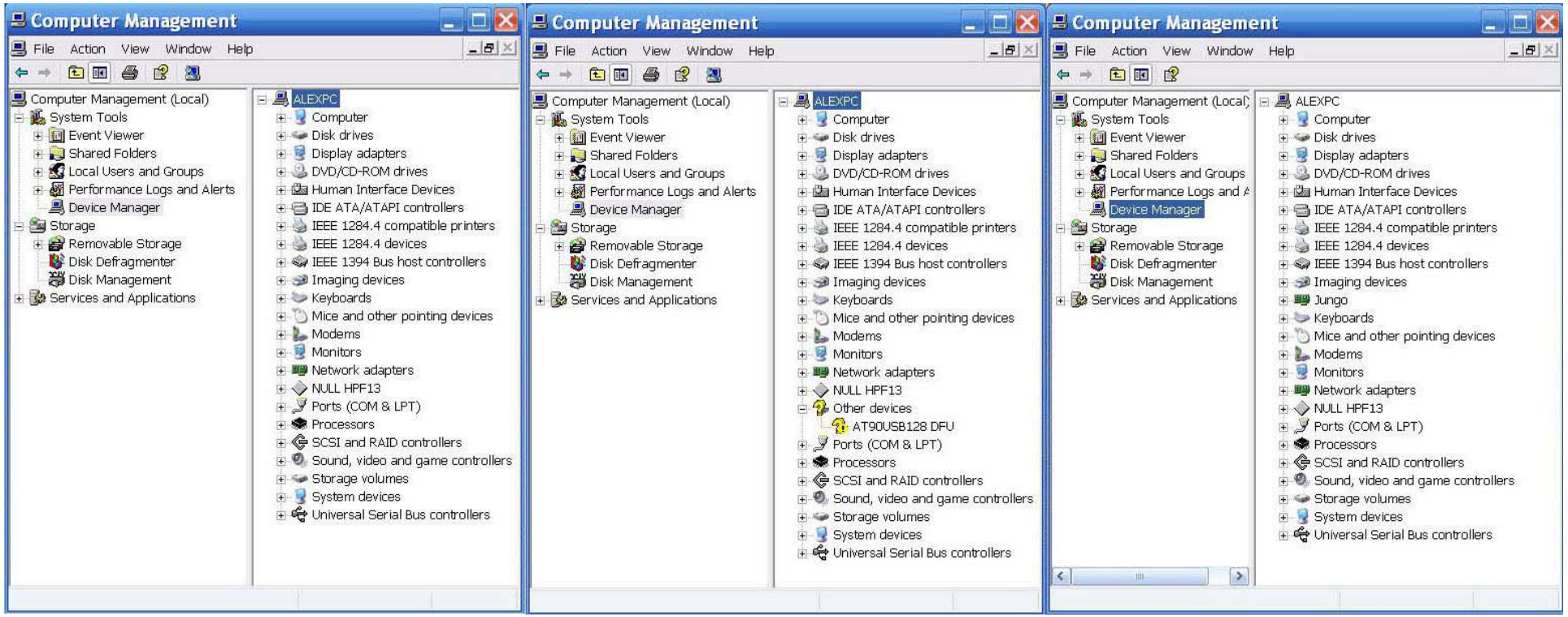
Case (b) is most likely going to occur if the device is left in its original configuration. To ensure that the device is in fact connected and detected, check the listing of devices on your machine. To do this on Windows XP, do the following:

- Select: **Start→Run**
- Type: **compmgmt.msc /s**
- Click: **Ok**
- Select: **Device Manager**

Shown below are three screen captures depicting what your device listings will look like in the three possible states when working with the AT90USBKey. The figure on the left shows the system when the key has initially been plugged in. The figure in the center shows a machine that does not have the device detected properly for programming (after a hard reset of the key). Notice the other device with the "?" beside it. The figure on the right does have the device detected properly for programming (after a hard reset of the key). Notice the addition of the "Jungo" device. If you do not have this listed, the drivers are not properly installed.

The drivers are located in the subdirectories of the FLIP 3 installation (Ex. *C:\Program Files\Atmel\FLIP 3.0.4\usb*). Use this location to provide the drivers for the USB key when prompted as in (a). From now on you should no longer be prompted for drivers.

<u>Note:</u> Before placing your new program on the key, first copy the existing files from the memory of the key to a backup folder. This will ensure that you can reload existing HEX files back onto the device if your program does not work (see files on key: starterkits\STK525-USBKEY\firmware\hex).*



Note: The first capture shows the system with the key acting as a normal USB key. The second caption displays the system after the hard reset of the key (without the drivers installed). The third caption shows the system after the driver from FLIP has been installed for the USB key.

Creating your first AVR Studio project

One of the key features of this IDE is the ability to simulate the different AVR controllers. To setup a programming project for the AT90USBKey follow the following example (remember you could just do this at the command line using WinAVR tools):

-Project->Project Wizard
-New Project
-Select: **AVR GCC**
-Project name: **FLASH_LED_DEMO**
-Click **Next**
-Select Debug Platform: **AVR Simulator**
-Select Device: **AT90USB1287**
-Click **Finish**.

Adding the source

Once the project is created you can start writing code to control the hardware on the AT90USBKey. Here is a simple C program that will turn on all the LEDs (read the document *doc7627.pdf* for information on why we use port D).

```
#include <stdlib.h>
#include <avr/io.h>

int main (int argc, char *argv[])
{
        DDRD  = 0xff;           /* Set all pins on port D for output
        PORTD = 0xff;           /* Initialize port to high --- turn on LED

        return 0;
}
```

Note: The header files that you should start learning are located in the following location: C:\WinAVR\avr\include\avr.

The following Makefile is derived from the CDC project on the CD provided by Mantis. It is documented to define most of the rules, however, you will probably need to read additional documentation from the following sources:

- http://www.gnu.org/software/make/manual/make.html
- *C:\WinAVR\doc\binutils\index.html*
- *C:\WinAVR\doc\gcc\gcc.pdf*

To add the Makefile, right-click on the *Other Files* folder in the *AVR GCC* tab (located by default at the left-side of your screen) and choose *Create New File.* Ensure that you have the *Makefile* displayed in your *AVR GCC* tab under *Other Files*.

```
#--- General Flags
PROJECT = FLASH_LED_DEMO                #Project name
MCU = at90usb1287                       #Target architecture
TARGET = FLASH_LED_DEMO.elf             #Target file
CC = avr-gcc.exe                        #Compiler for C files

#--- Options common to compile, link and assembly rules
COMMON = -mmcu=$(MCU)                   #Compile for target arch.
```

```
#--- Compile options common for all C compilation units.
#           -D                      Predefine as macro
#           -Wall                   Enable all optional warnings
#           -gdwarf-2               Produce debugging information in DWARF format (version 2)
#           -Os                     Optimize for size
#           -fsigned-char           Let type char be signed
#           -MD -MP –MT -MF         Specify where dependencies are written
CFLAGS = $(COMMON)
CFLAGS += -D AVRGCC -Wall -gdwarf-2  -Os -fsigned-char
CFLAGS += -MD -MP -MT $(*F).o -MF dep/$(@F).d

#--- Assembly specific flags
ASMFLAGS = $(COMMON)

#--- Linker flags
LDFLAGS = $(COMMON)

#--- Intel Hex file production flags (used by objcopy
#           -R                      Remove named section from output file
#           -j                      Copy only the named section from input to output
#           --set-section-flags     Set flags for named section
#           --change-section-lma    Set the address for the section to be loaded at
HEX_FLASH_FLAGS = -R .eeprom
HEX_EEPROM_FLAGS = -j .eeprom
HEX_EEPROM_FLAGS += --set-section-flags=.eeprom="alloc,load"
HEX_EEPROM_FLAGS += --change-section-lma .eeprom=0

#--- Objects that must be built in order to link
OBJECTS =  FLASH_LED_DEMO.o

#--- Build
all: $(TARGET) FLASH_LED_DEMO.hex

#--- Compile
FLASH_LED_DEMO.o: FLASH_LED_DEMO.c
        $(CC) $(INCLUDES) $(CFLAGS) -c  $<

#--- Link, pull it all together
$(TARGET): $(OBJECTS)
         $(CC) $(LDFLAGS) $(OBJECTS) $(LINKONLYOBJECTS) $(LIBDIRS) $(LIBS) -o $(TARGET)

#--- Building Intel MCS-86 Hexadecimal Object
%.hex: $(TARGET)
        avr-objcopy -O ihex $(HEX_FLASH_FLAGS)  $< $@

#--- Building EEPROM initialization file
%.eep: $(TARGET)
        avr-objcopy $(HEX_EEPROM_FLAGS) -O ihex $< $@

#--- Create extended listing file
%.lss: $(TARGET)
        avr-objdump -h -S $< > $@

#--- Clean target
clean:
        -rm -rf $(OBJECTS) FLASH_LED_DEMO.elf dep/* FLASH_LED_DEMO.hex  \ FLASH_LED_DEMO.eep
```

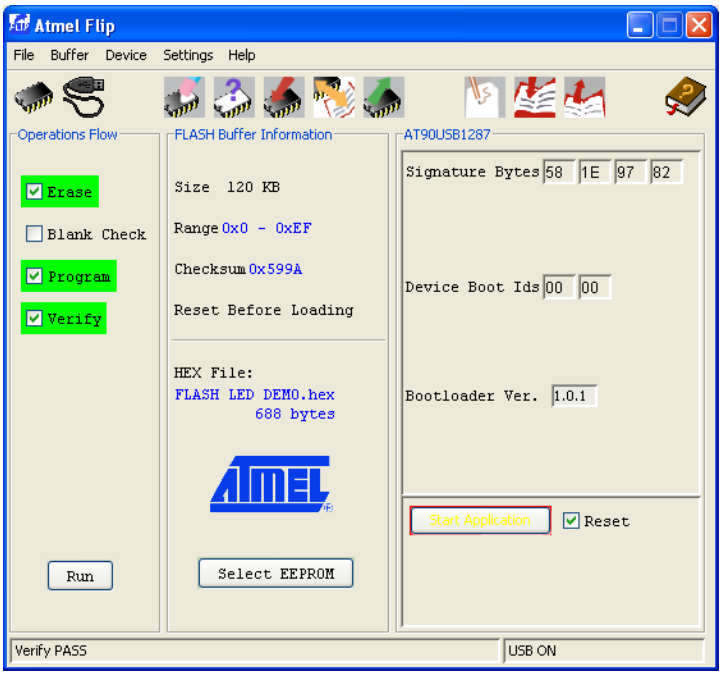
Building the project

To build the project, simply choose ***Build→Rebuild All*** from the menu bar at the top of the IDE (remember to save all your changed files first). You also need to ensure that you are building with the makefile you created above. This is done through ***Project→Configuration Options→Use External Makefile***. Once you get the code to build cleanly we are ready to load the program onto the key.
Note: You can also use this environment to debug in a simulator environment.

Loading your first program on the key using FLIP

To load a program onto the AT90USBKey, perform the following steps:

1. Connect the AT90USBKey to the development machine using the USB cable.
2. Do a hard reset of the key: **RST** and **HWB** buttons held down, then release **RST**, finally release **HWB**. Only the power light should be on.
3. Start FLIP.
4. Select: **Device→Select…**
5. Choose the **AT90USB1287** controller.
6. Click on **OK**.
7. Select: **Settings→Communications→USB**.
8. Click on **Open**.
9. Select: **File→Load HEX File…**
10. Select your **FLASH_LED_DEMO.hex**
11. Click on **OK.**
12. Click on **Run** (bottom left corner of FLIP).
13. Press the **RST** button once. The LED lights should now light up.



Closing Remarks

Use the HEX file *starterkits\STK525-USBKEY\firmware\hex\storageUSBKEY.a90* to restore the key to a state where it can act like a normal USB key using the FLIP utility.

Good luck!