ScreenKey USB5000 Developer Kit

User's Guide 0.8

November, 2010

www.ScreenKeys.com

ScreenKey USB5000 Dev Kit

User's Guide

Information in this document is subject to change without notice. The latest revisions may be accessed on the SKI Web site.

    Web: www.ScreenKeys.com

Technical Support is available

    via Email   support@screenkeys.com

    via Web: www.screenkeys.com

DISCLAIMER:

ScreenKeys reserves the right to revise data file formats and functionality at any time.

Table of Contents

I N T R O D U C T I O N

ScreenKey USB5000 Dev Kit Overview

Introduction

The USB5000 Dev Kit allows developers to understand and experiment with the features and functionality of RGB and LC Series ScreenKeys.

The USB5000 Dev Kit contains two on-board ScreenKeys and includes source code firmware written in 'C' which can be easily modified and recompiled using a free 'C' compiler. Developers can quickly modify the provided source code to generate their own images, text and other functionality to test and prototype using RGB and LC Series ScreenKeys.

The USB5000 Dev Kit is reprogrammable via USB.

Software Updates

From time to time, SKI will issue new firmware updates as well as new programming tools. These can be downloaded from ScreenKeys website at www.ScreenKeys.com.

Sample Code

Sample code, in C, is provided but new versions may be released from time to time to illustrate how to drive and interface to ScreenKeys. Sample code can be accessed from the ScreenKeys web site at www.ScreenKeys.com.

Disclaimer

ScreenKeys reserves the right to revise this user manual or product specifications at any time.

H A R D W A R E

USB5000 Schematic Design

Dev Kit Hardware Overview

The USB5000 Dev Kit is a simple microcontroller unit that drives two RGB24 ScreenKeys using a parallel-to-serial converter arrangement. It is one demonstartion of how ScreenKeys may be driven by a microcontrolller. The two RGB24 ScreenKeys may be replaced by LC24 or LC16 ScreenKeys as required. The schematic of the USB5000 is shown on the following pages.

The MCU is an Atmel AT89C5131 which is reprogrammable in-circuit via USB. The power supply to the board is taken from the USB 5.0V supply which is also used to power the onboard ScreenKeys. The MCU has 32KB flash memory and 1Kb RAM.
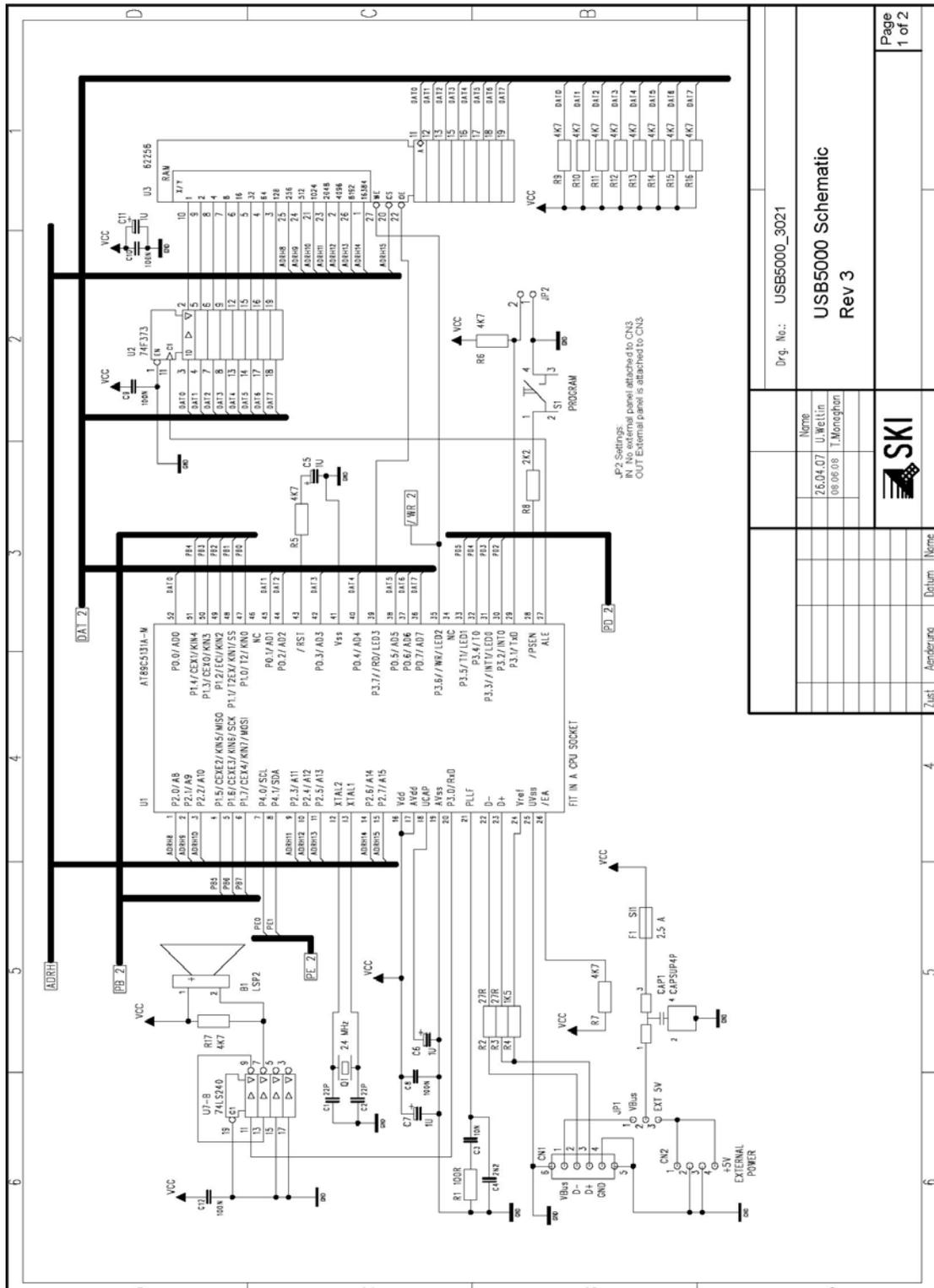
Currently, the USB interface is only used for programming and for power supply purposes. The USB5000 does not enumerate on the USB bus to offer direct control.

There are several ways in which ScreenKeys may be driven. ScreenKeys receive data serially synchronised with a clock signal. To minimise overheads on the MCU, the USB5000 writes data in parallel from the CPU into a parallel-to-serial converter which then clocks this data serially to the ScreenKeys.
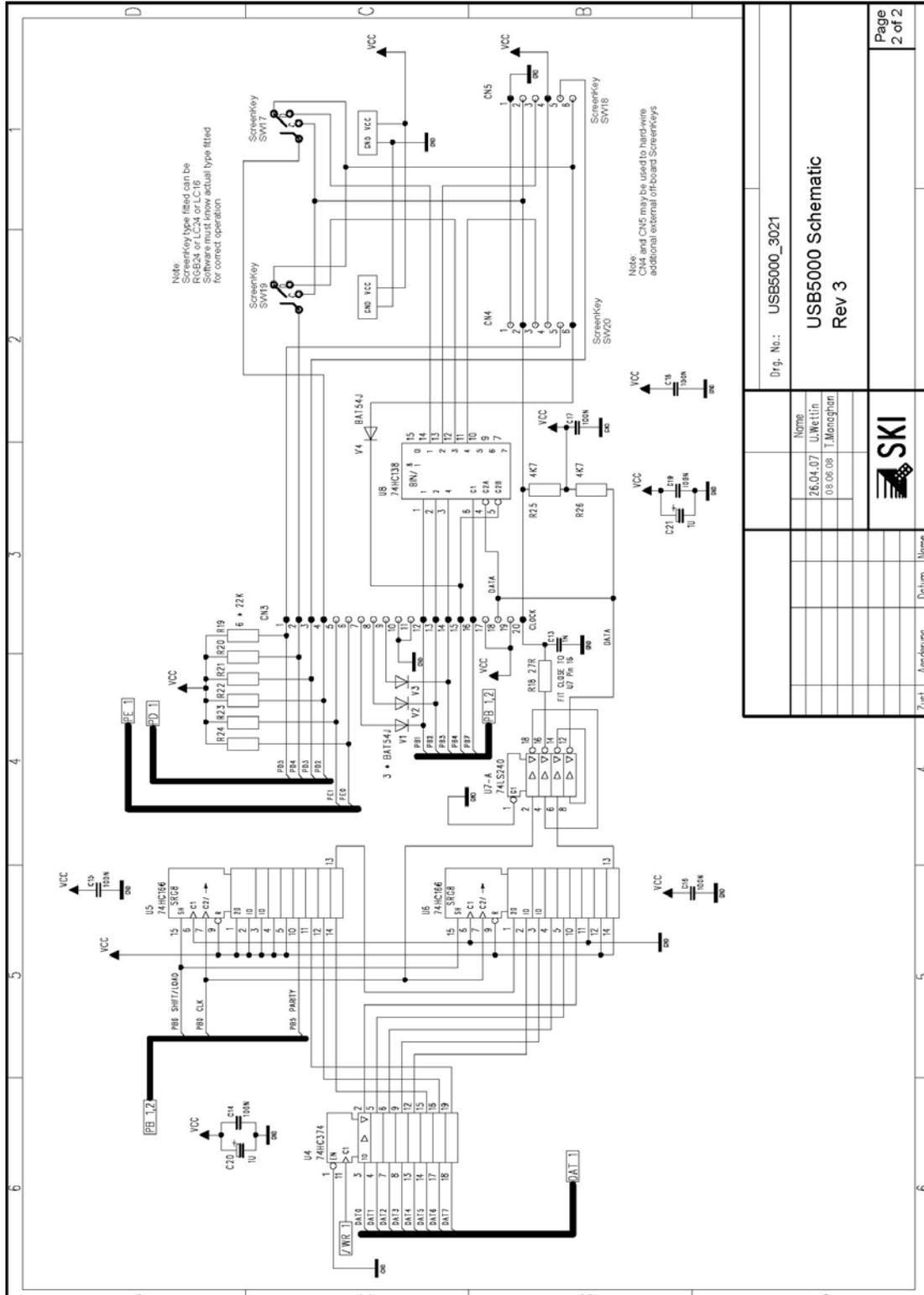
Alternative approaches can use general purpose IO lines for both the clock and data lines and to drive these directly from the MCU. As the ScreenKeys require a continuous clock even when data is not being written, this approach can utilise additional MCU resources. Alternatively, the clock line can be taken from a PWM output of the MCU which runs freely when data is not being written to the key and then directly control this PWM output for data writes.

The Atmel AT89C5131 on the USB5000 provides an integrated clock generator output which is basically a 50% duty-cycle PWM output. This is connected to the ScreenKeys to provide the continuous clock required by ScreenKeys. For serial data writes to the keys, the clock is stopped and the clock line is directly driven to the parallel-to-serial converters to provide the clock/data combination to write information into the key.

The USB5000 supports connection of a separate panel of ScreenKeys via on an onboard connector. This panel can support a grid of 3x4, 4x4 or 5x4 ScreenKeys.

USB5000 Schematic
Rev 3

Drg. No.: USB5000_3021

Page
1 of 2

ScreenKey Driver Design

ScreenKeys use four pin connections for LCD and LED backlighting control and two pins for the switch:

1. Gnd

2. Vcc (+5V)

3. CLK

4. DATA

5. Switch

6. Switch
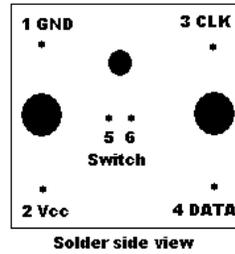
The two switch pins implement an electrically isolated single-pole momentary action switch circuit.

Serial Data Generation

Data to the ScreenKey is serially transmitted (DATA) synchronised to a user supplied clock signal (CLK).

The format of the DATA signal is 12 bits, transmitted as one start bit followed by data (8 bits, low-bit first), one parity bit and two stop bits:

Start Bit	low
Data Bits (d0-d7)	low/high
Parity Bit	low/high
Stop Bits (2 bits)	high

It is possible to generate this data stream serially in software and to coordinate its transmission with a software generated clock. However, such an approach can be resource intensive. A continuous clock signal is required by the ScreenKey even when data is not being written and this can cause additional overheads to produce this in software.

As most MCU devices provide some configurable PWM output, using a PWM output is the most efficient method for generating the clock. A PWM output can be setup for continuous operation and this can be interrupted and directly driven whenever it is required to write data to the key.

The 12-bit word may be created and output serially in sync with the directly driven clock line. In the USB5000, the 12-bit word is written in parallel into a paralllel-to-serial

converter (P2S). The data is then serially clocked out of this P2S to the ScreenKey. This reduces the software overhead on the MCU so that it does not have to serialise ScreenKey data. It should be noted that it is not particularly cumbersome to directly produce the serial data stream in software and to drive the serial line directly to the ScreenKey (thereby eliminating the need for the P2S logic).

The example opposite demonstrates how the USB5000 outputs data to the ScreenKey. U5 serial out (pin 13) is shifted directly into U6. U6's serial out feeds the DATA line to the ScreenKey. This means that we actually clock 16 bits per word. Our 12-it word is book-ended by HIGHs (equivalent to STOP bits. Serial data is shifted out by the CLK signal which is also fed directly to the key.

In this example, parallel data is written with an initial two high bits (U6 – 14,12) then the low START bit (U6 – 11). The 8-bit data byte is next with lsb first (U6 – 10,5,4,3,2 and U5 – 14,12,11). The data bits are followed by the PARITY bit (U5 – 10) and then 2 high STOP bits (U5 – 5,4). Highs are forced onto the remainder of the 16 bits (equivalent to additional STOP bits).

To use this hardware, prepare the data word (8 data bits and appropriate parity – see ScreenKey datasheet for parity usage) and set these inputs. Load this word into the shift register by toggling SHIFT to low and back to high. Enable this data to be serially shifted into the ScreenKey by setting LOAD to low.

Toggle the clock line from low-to-high 16 times to ensure that the full 12-bit word is clocked out (including the bookends).
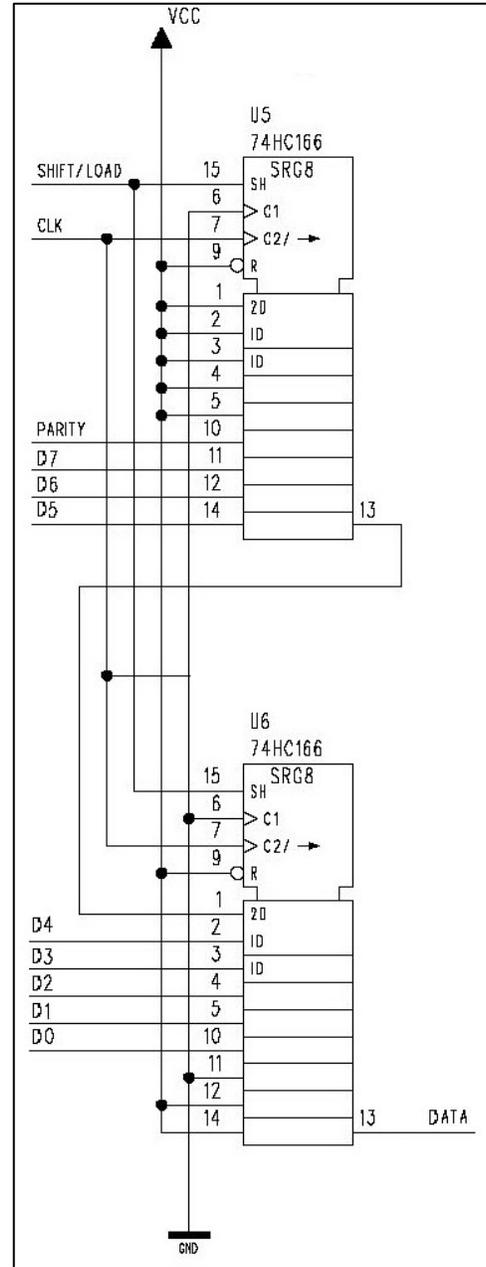
CLK Signal

The ScreenKey datasheet specifies that the CLK signal can be anywhere between 50KHz and 4MHz. With a microcontroller circuit there is usually some CPU related clock signal that can be tapped to generate this CLK signal. For example, most microcontrollers offer a PWM output which can be set to 50% duty-cycle to produce an acceptable clock source.

The clock source does not need to be a crystal. Similarly the clock frequency is not critical and does not need to remain constant. However it is necessary that the clock signal remains operational whenever power is applied to the ScreenKey. It is acceptable for the clock to be delayed during bootup of the microcontroller.

The clock line output can then be overtaken by the MCU under direct software control to synchronise transmissions to the key (via CLOCK and DATA line syncing).

The USB5000 utilises a dedicated clock output from its Atmel MCS89C5131 microcontroller. This port output is allowed to run freely when the key is not being written to. The clock is then stopped and the port IO line is driven directly under software control to synchronise data transmissions to the key.

Driving Multiple ScreenKeys

Driving more than one ScreenKey is simply a matter of enabling the DATA line to be routed separately to each of the desired number of ScreenKeys.

For example, to drive four ScreenKeys a simple 2-4 decoder (74HC139) could be used to route the DATA signal to a selected key. The USB5000 supports up to 20 keys so it has a slightly more complicated arrangement. A 3-8 line decoder (74HC138) is used to select between the two onboard ScreenKeys and the additional two ScreenKeys that can be connected via CN4 and CN5. Separate decoding circuitry is used on the ScreenKey matrix panels that can be connected via CN3.
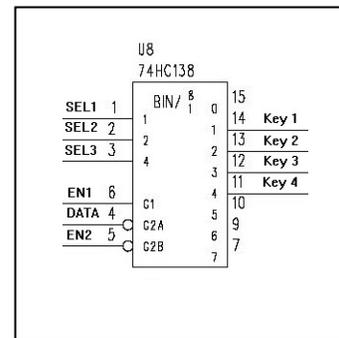
There is no need to selectively apply the CLK signal to each ScreenKey. The CLK signal may be applied continuously to all ScreenKeys at the same time. Standard line buffering techniques should be employed if more than four ScreenKeys are to be driven from the same CLK line.

Reading Switch Presses

The ScreenKey switch mechanism is an electrically isolated circuit and operates as a single-pole normally open momentary action.

Any standard approach to detecting key switches can be employed. Usually this involves setting one side of the switch to a certain state and reading the other side of the switch to see if the signal is being passed across the switch. This requires some form of key scanning to alternately apply this test to each key individually.

This approach is implemented here with the USB5000 using a grid matrix of scan and read lines to detect a switch press. To reduce the number of required IO lines from the MCU, the key select lines that drive the decoder are multi-tasked with also operating as the switch scan lines. Distinct switch read lines are allocated solely for reading a switch press and these lines use a 22K pull-up resistor connected to Vcc.

The arrangement in the diagram opposite shows a simple way to detect the switch state. One side of the switch is connected to VCC via a pull-up resistor and this is monitored by the MCU. The other side is connected to a scan line from the MCU. In the cas eof the USB5000, this is a key select line via a fast switching diode - BAT54J. The key select line is set low and if the 'SW Read' line reads as a low then the switch is pressed. If not pressed, the MCU line will read back as high.
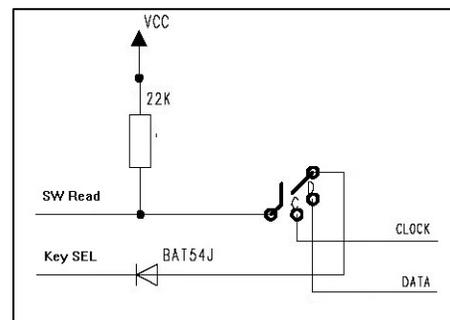
G E T T I N G   S T A R T E D

Programming Tools

Atmel FLIP

The onboard MCU on the USB5000 is an Atmel AT89C5131. This is an 8051-based CPU which is reprogrammable via USB.

Atmel provide a programming utility called FLIP, provided on the USB5000 CD or which can be downloaded from Atmel's website:

http://www.atmel.com/dyn/products/tools_card.asp?tool_id=3886

To run FLIP also requires the Java Runtime Environment. Atmel offer a FLIP download that includes the Java Runtime Environment or a FLIP download without this for users who already have Java installed. There is also a Java Runtime install included on the accompanying USB5000 Dev Kit CD.

Use of the FLIP program is beyond the scope of this document as the "ScreenKey USB Programmer" provides a simple functional subset of FLIP for use with the USB5000. Installation of FLIP and Java Runtime is a necessary precursor to using ScreenKey USB Programmer.

FLIP is offered for both Linux and Windows.

ScreenKeys USB Programmer

The accompanying USB5000 Dev Kit CD includes a folder called "ScreenKey USB Programmer". This includes a command-line utility which identifies when the USB5000 is attached via USB and programs a specified hex file automatically.

To use this utility, copy the contents of the "ScreenKey USB Programmer" utility to a local folder on your own computer. Included in this folder is the current version of the USB5000 firmware (called USB5Kv0_8.hex).

Run the utility by running the included batch file (USBProg.bat) or opening a command window, navigating to the local folder and typing:

<drive>:\<local folder>\"ScreenKey USB Programmer" USB5Kv0_8.hex

i.e. pass the name of the hex file to be programmed to the utility on the command line.

Note that *ScreenKey USB Programmer* only runs under Windows.

Programming Mode

The USB5000 does not automatically boot into USB programming mode. Its default startup mode is to run the user programmed application.

To enter programming mode, hold the PRG button on the USB5000 while plugging it into a USB socket.

First-time Device Identification

The first time the USB5000 is plugged in, it will identify that an unknown device has been attached. Depending on your operating system, specify that you want to install a specific driver from a specific location.

Browse to the accompanying CD, and open the folder "USB DFU Driver". This will begin installation of the Atmel AT89C5131 programming driver files.

After installation completes, open *Device Manager* and you should see the following entry that confirms the device has been installed successfully:

LibUSB-Win32 Devices
AT89C5130/AT89C5131
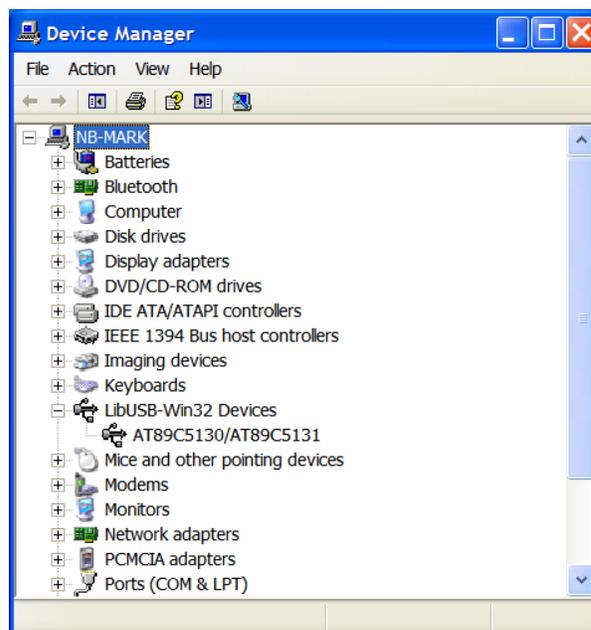
SDCC Compiler

The accompanying USB5000 demonstration source code is written in "C" for the SDCC compiler.

SDCC is a free compiler which can be downloaded from the web:

http://sdcc.sourceforge.net/

Go the download page and download the relevant install for your operating system.

The supplied source code compiles correctly with version 2.9 of SDCC. A distributable install of 2.9 for Windows is included on the accompanying CD.

Startup Sequence

Before beginning programming of the USB5000, the following sequence should be adhered to:

1. Attach USB5000 to USB socket and ensure pre-programmed application runs correctly. Press the onboard ScreenKeys and note that displayed images change.

2. Disconnect the USB cable and then while holding the PRG button, reconnect the USB cable. Release the PRG button.

3. Point your operating system to the Atmel USB DFU drivers on the accompanying CD in folder "USB DFU Driver". Wait for computer to successfully finish driver installation.

4. Check *Device Manager* for a new entry called "LibUSB-Win32 Devices" and a sub-entry called "AT89C5130/AT89C5131".

5. Install FLIP from web download or use install provided on CD.

6. Install SDCC from web download or use install provided on CD.

7. Copy CD folder called "ScreenKey USB Driver" to a suitable local drive/folder.

8. Copy the USB5000 demo source code from the folder called "USB5000 Source" to a suitable local drive/folder.

You are now ready to begin modifying the supplied source code and to make the USB5000 perform.

D E M O   F I R M W A R E

Demonstration Source Code

File Listing

The source code for the USB5000 is supplied on the accompanying CD in the folder "USB5000 Source".

The included files are:

Main.c	Main function with executive loop
LCDKeyMgr.c	Routines to drive ScreenKeys including text to graphics
KbdMgr.c	Key scanning and recording of keypresses
TimerMgr.c	Software timers
ErrorMgr.c	Recording and access to error list
DemoMgr.c	Implements demonstration of images/text/colours
Images.h	Various images for ScreenKey display
USB5000.h	Configuration setup and #defines
At89C5131.h	Includes SDCC compatible SFR definitions
Ext_5131.h	Defined masks for accessing SFR registers
Compiler.h	Generic definitions for various variable types
Build.bat	Command-line utility to build application
USB5Kv0_8.hex	Original hex file as pre-programmed in USB5000

A subfolder called "temp" exists off the source folder. During compilation, SDCC generates various files (e.g. map, mem, asm, lst, etc) and these are stored in the temp folder.

A successful compilation results in a file called *USB5000.hex* created in the source folder. This can be directly programmed into the USB5000 using the *ScreenKey USB Programmer* utility by running the supplied batch file from the command line (remember to navigate to the folder where *ScreenKey USB Programmer* resides and also to copy the USB5000.hex file into this location):

<drive>:\<local folder>\USBProg

or, running the utility directly:

<drive>:\<local folder>\"ScreenKey USB Programmer" USB5000.hex

Remember to include the quotation marks as otherwise the command line will fail.

Source Code Explanation

The source code for the USB5000 Dev Kit demo is quite simple. Main() simpy initialises the different software modules and then polls each module in turn to allow it to process its own tasks.

LCDKeyMgr is the main software module of most interest. This modules includes functions that initialise the ScreenKey as per the ScreenKey datasheet (setting the MUX and FREQ registers), setup ScreenKey backlight colours through the COLOUR register, and also demonstrates how to send graphic images to a ScreenKey. The module also includes routines for dynamically converting text to graphics. There are two fonts included in the module for text to graphics support:

> **Normal font:**   **5x7 pixel space (fixed width)**
>
> **Small font:**   **3x5, 4x5 or 5x5 pixel space (variable width)**

The routines in this module support RGB24/LC24 ScreenKeys with 36*24 pixel resolution and LC16 ScreenKeys with 32*16 pixel resolution. The actual key types used with the USB5000 must be defined in the USB5000.h file.

KbdMgr module handles key scanning and recording of key presses. Key scanning is handled by a timer interrupt. The key scan routine skips a scan sequence if the ScreenKey is being accessed from the LCDKeyMgr module to prevent corruption due to multi-tasking of the key select lines as key scanning lines.

DemoMgr implements a sequence of images and text displays on the ScreenKeys that illustrate what can be shown on the keys and what backlight colours are possible.

GRAPHIC IMAGING

Image Preparation

Refer to the ScreenKey datasheet for a description of how the internal graphic memory is mapped to the pixel displays on the RGB24/LC24 and LC16 ScreenKeys.

The included file Image.h demonstrates many different image files for both ScreenKey resolutions.

A utilty called "SAC Editor" provided by ScreenKeys for other purposes may be used to create your own images in a format suitable for inclusion in the USB5000 demonstratation source code. The SAC Editor utility is included on the USB5000 Dev Kit CD along with the SAC Editor Users Guide.

SAC files should be considered as repositories of ScreenKey images. Although this utility is for use with ScreenKey POS consoles, it can be used to manipulate and store images in unique files. Install the application and then create a new SAC file in which you will sotre your ScreenKey images. You must identify the type of unit and ScreenKeys used (specify OEM5400 with the ScreenKeys you are working with, i.e. RGB or LC, 36*24 or 32*16). Also specify a main menu which is required although it will not be used for USB5000 purposes.

Open the graphic editor by clicking on the graphics icon, or click on the Options menu and the Graphics option.

This will open the graphic editing window:
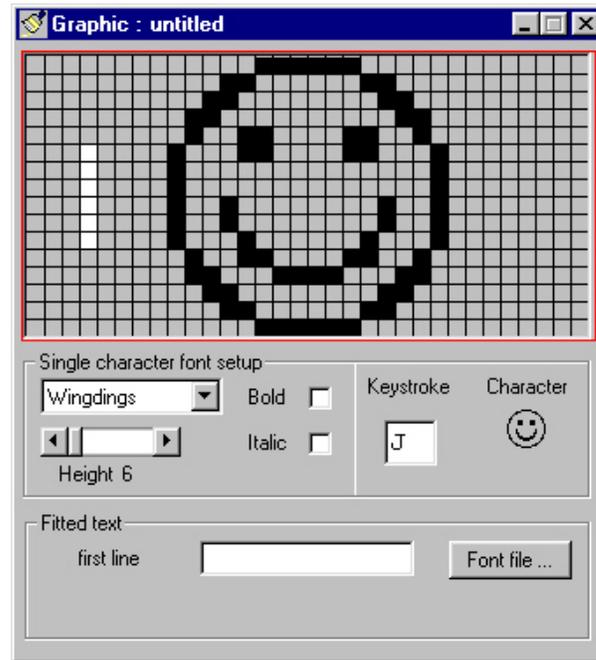
The Graphics window

NOTE If LC24 resolution ScreenKeys were selected as the target the Graphics window will show 36 pixels across and 24 pixels down.

You can create image by clicking on pixels to turn them on or off, import images created with another package, or specify text and use fonts installed on your PC.

Once you are satisfied with your image, save the image and give a suitable name. This will save the image in the SAC file repository.

Next, you can export this image either as a binary file or as a C-hex array suitable for inclusion in the USB5000 as a C-header file. Simply click on export menu option and select "compiled" format. Enter a suitable filename and note the location in which the image file is stored. You will likely want to change the name of the C-hex array and include it in the Image.h file in the USB5000 project.

A P P E N D I X   A

Documentation Control

A.1 Change Control

This document is the responsibility of the author and is subject to formal change control after the initial approved release (i.e. issue 1.0).

A.2 Abbreviations Used/Terms of Reference

LCD	Liquid Crystal Display
SPI	Serial Protocol Interface – industry standard high-speed synchronous serial interface with clock and data lines
MCU	Microprocessor Control Unit
LED	Light Emitting Diode
USB	Universal Serial Bus – common interface conector for PC

A.3 Historical Change Reference

Issue	Date	Author	Changes Made
0.8	Nov 15, 2010	MMcD	First Release

A.4 Change Summary