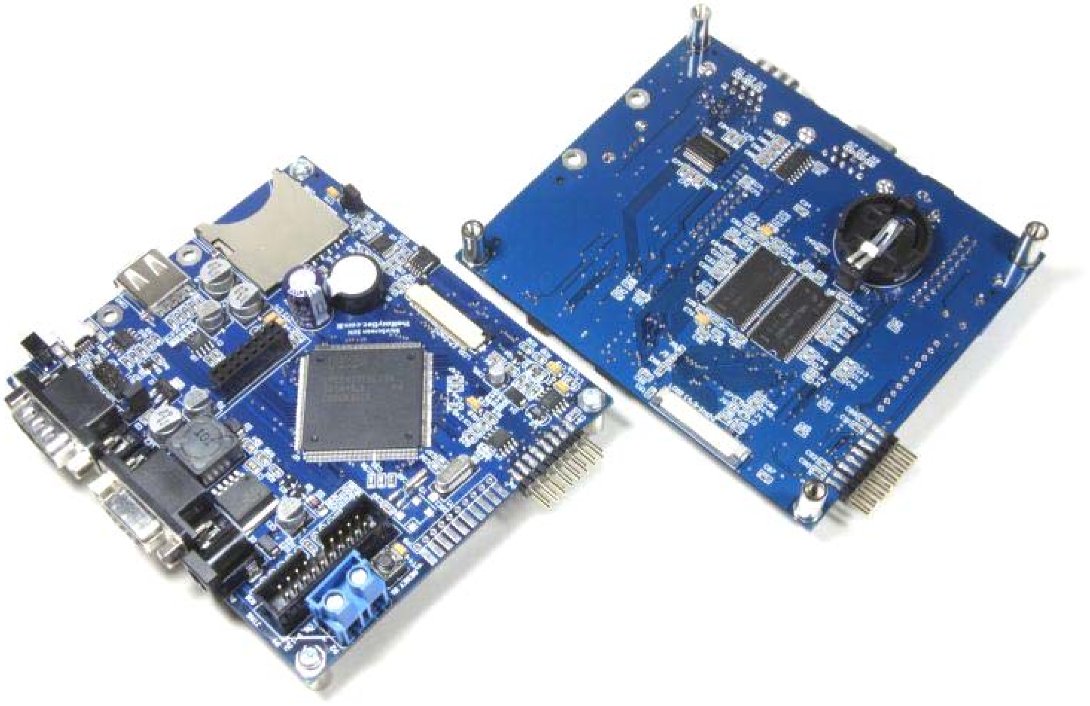
BlueScreen SUN

User Manual v1.00



Enable Your Design

ThaiEasyElec.com

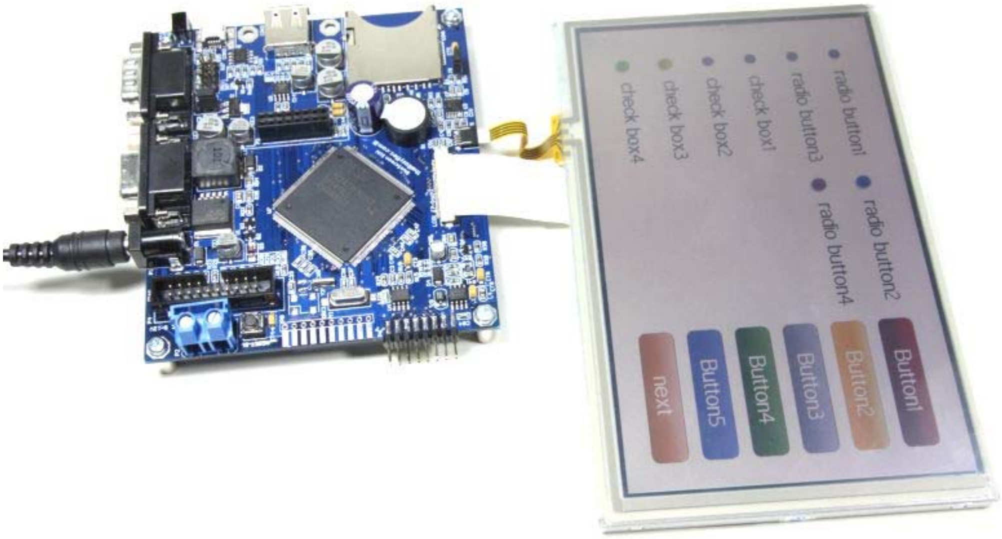
On-line Electronics Shop for Embedded System

Disclaimer:
This document is intended only to assist the reader in the use of the product. ThaiEasyElec.com shall not be liable for any loss or damage arising from the use of any information in this document or any error or omission in such information or any incorrect use of the product.

Index

Appendix

1. Introduction



Blue Screen SUN is a touch screen development board comes with NXP’s powerful ARM7 LPC2478, the LCD controller integrated MCU. Addition with 64MB SDRAM, optional Ethernet module, SD card socket, 8kB EEPROM and connectors for resistive-type-4.3-inch and 7-inch touch screen TFT LCD, Blue Screen SUN is the new generation of famous 2.8-inch touch screen development board ‘Blue Screen’ with bigger size LCD and much more features.

Two source code bundles support non-OS based and OS-based platform. Both platforms share the same touch screen middleware with Blue Screen. Developers who have got familiar with Blue Screen can upgrade there products to bigger LCD easily.

2. Features

Hardware

- NXP’s ARM7 LPC2478
- 2 of 16-bit Micron’s MT48LC16M16A2P-75 SDRAM providing 32-bit, 64MB memory
- Connector for 480x272 pixels TFT LCD with touch screen
- Connectors for 800x480 pixels TFT LCD and 4-wire resistive touch screen panel
- SD card socket (connected via SPI interface) support up to 2GB capacity (High capacity: ‘HC’ type not supported)
- On board 8kB EEPROM (the last 128 bytes are reserved for screen calibrated parameters)
- Serial Port 0 with selectable UART connector (TTL 3.3v with 5v tolerant) or female DB9 connector (RS232) for command line interface and in-system programming (select by a jumper)
- Serial Port 1 with full modem signals on male DB9 connector
- 7 GPIO ports (including ADC ports)
- 1 port ThaiEasyElec’s module connector consisting of SPI and UART signals from MCU (can be used as 10 GPIO ports)
- One USB host interface with USB type A connector and power switch IC LM3526M-L
- One USB device interface with mini-B USB connector (the board cannot be powered from USB connector)
- Connector for ThaiEasyElec’s DP83848 Ethernet module
- JTAG connector for programming and debugging
- Connectors for 9-16VDC power supply (DC jack and screw-type)
- Buzzer

Software

Demonstrating application

- Graphic Library demo
- Catalog demo

Command line interface software module

- Show image from SD card
- SD card commands e.g. change directory, list, open read, open write

Screen object software module

- Design your screen with object oriented method, the running background software will manage which object should be operated

Low level drivers

- LCD driver
- Touch screen controller (AD7843) driver
- Serial port
- SPI interface
- I2C interface

3. Requirement

- USB cable (type A to mini-B) (optional)
- 9V-16VDC power supply
- ThaiEasyElec's USB to serial and cable (optional) for command line interface and programming
- Stylus (optional)
- SD card (optional)
- Supports programmer/debugger (optional) ULink, ULink2 and more
- DP83848 Ethernet module (optional)

VS1011E Module

USB mini B to Serial

DP83848 Ethernet Module

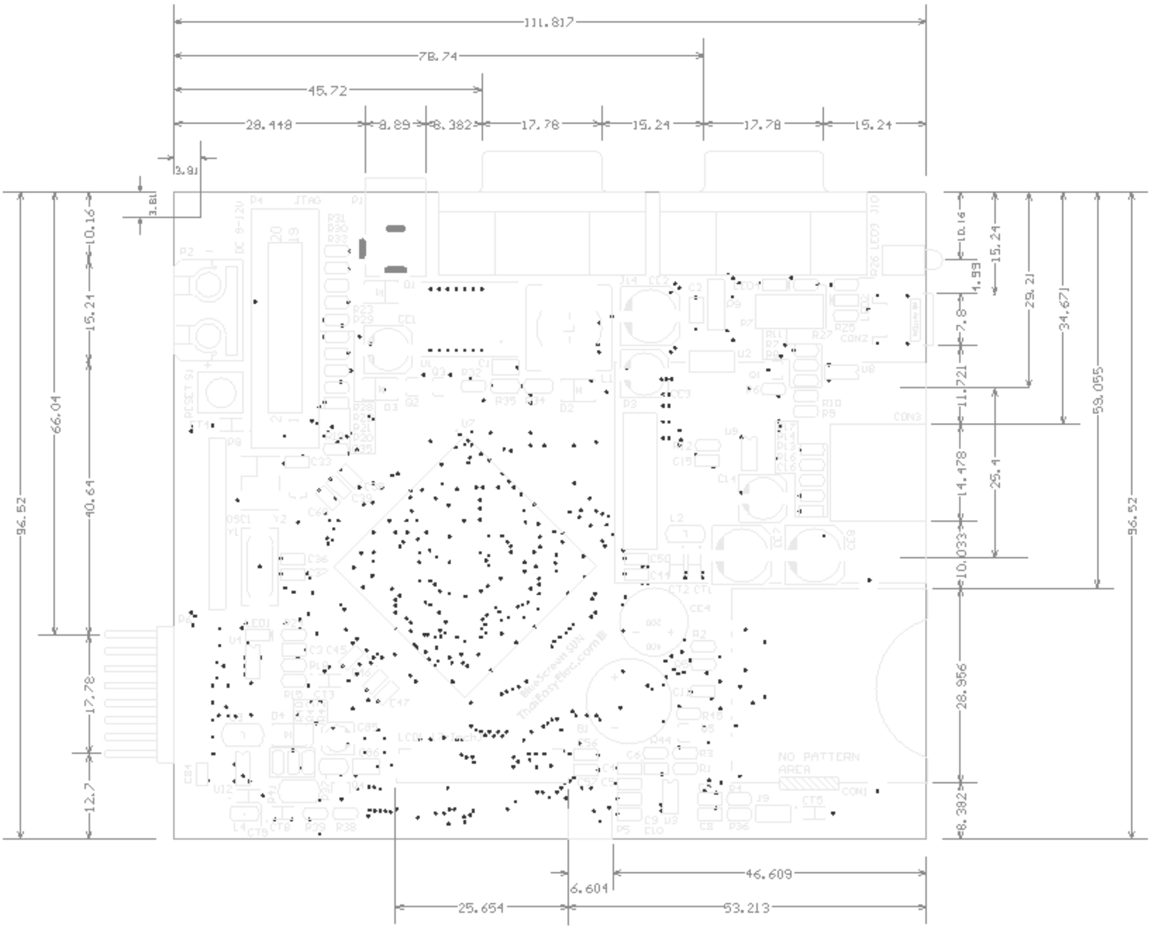
Stylus

mini N-link ARM USB JTAG **

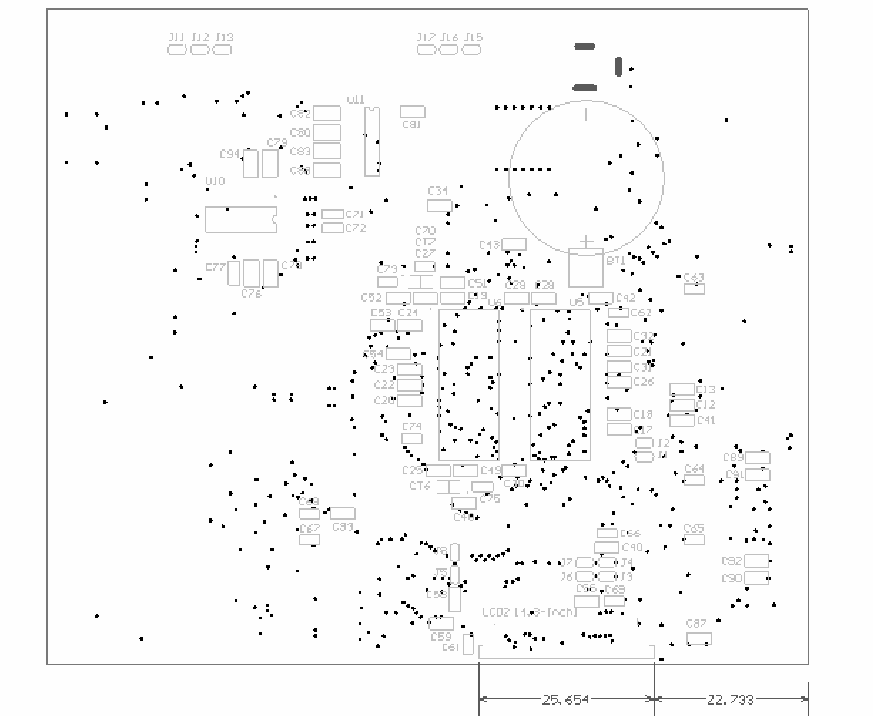
** **mini N-link ARM USB JTAG is now obsoleted**
Please refer the ARM USB JTAG as link below
http://www.thaieasyelec.com/index.php?lay=show&ac=cat_showcat&l=3&cid=1733

4. Drawing

(in mm.)



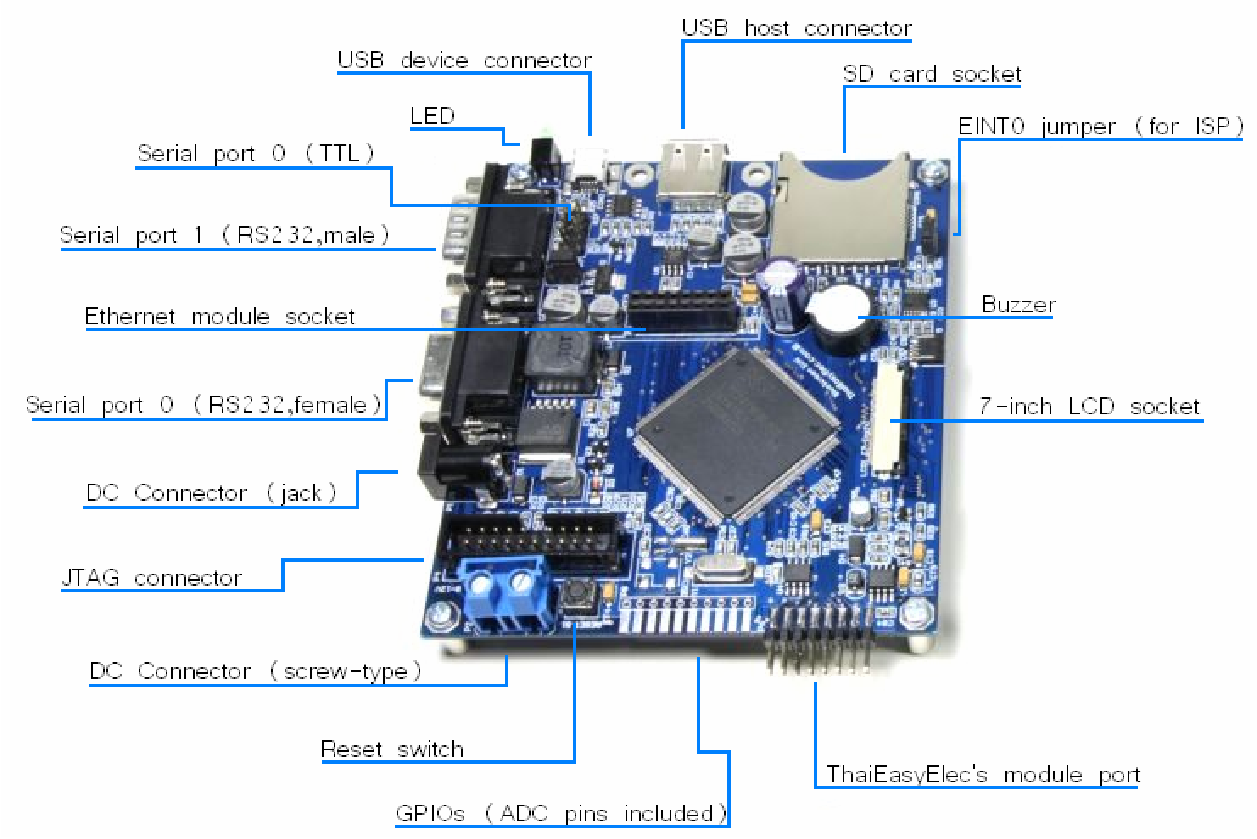
Top side



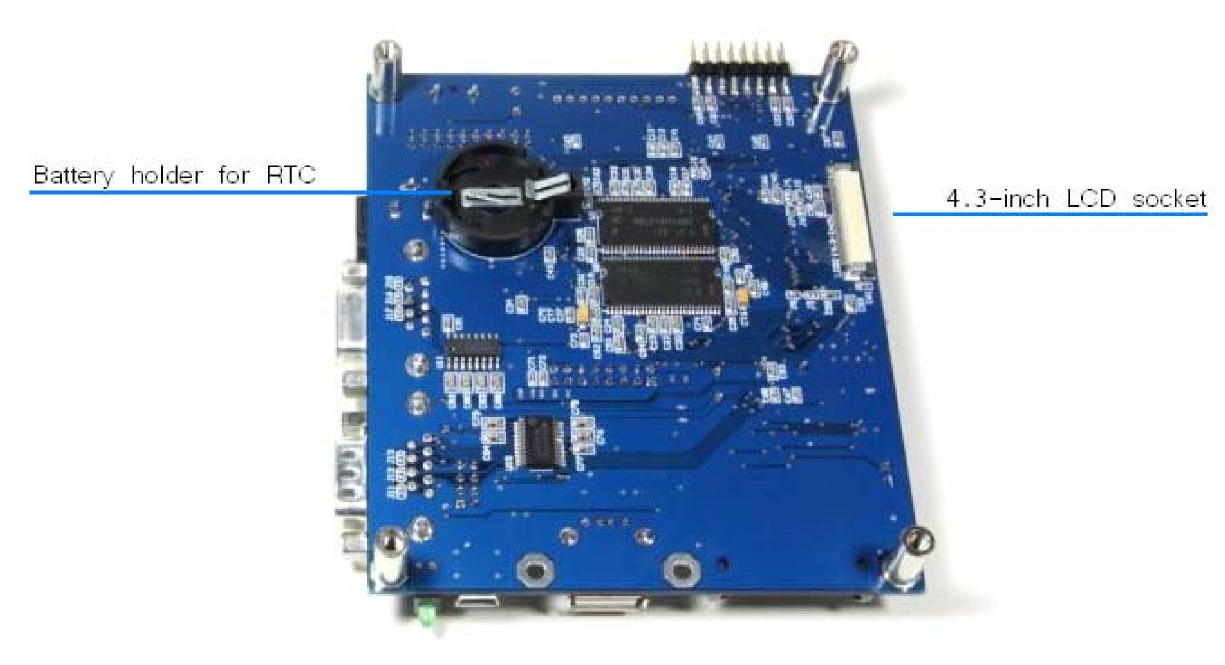
Bottom side

5. Getting start

5.1 Peripherals



Top side



Bottom side

5.2 Power it on

The board can be powered via 2 connectors with 9-16VDC. Both screw-type and jack-type (positive-voltage inside) are shorted together on board, and is protected with half-bridge diode for incorrect polar power.

The board shows “Press the screen to recalibrate within 1 second” on a blue screen. In this state, if you want to recalibrate the screen, press on it. Anyway, the board is calibrated from our factory so users don’t need to do it again.

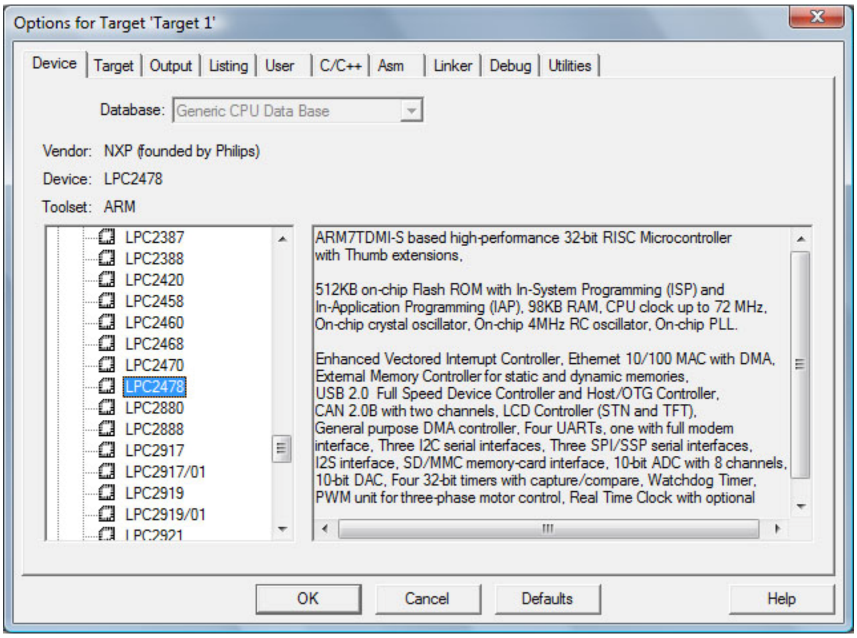
When the board starts up, it loads calibrated parameters from last 128 bytes of EEPROM (see function AppCalibrateScreen2() in app_bs_sun_demo.c). These parameters are used to calculate which point the screen is pressed. They are variable on each board. In case that user overwrites some of these data in EEPROM. At the starting up, the calibration state (picture below) will show automatically without waiting.

In order to recalibrate the screen, there would be 3 points on the screen to be pressed. For accuracy, please touch it slowly and a stylus should be used. When finishing, the demo example will be shown.

5.3 Creating project with Keil

Contents below are referenced to Graphic Library example; there may be some differences in other examples.

The board is supplied with codes. Users have to make project yourself. This guide is an example for Keil. Firstly, you have to create a project. And select the device as "LPC2478"



Database: Generic CPU Data Base

Vendor: NXP (founded by Philips)

Device: LPC2478

Toolset: ARM

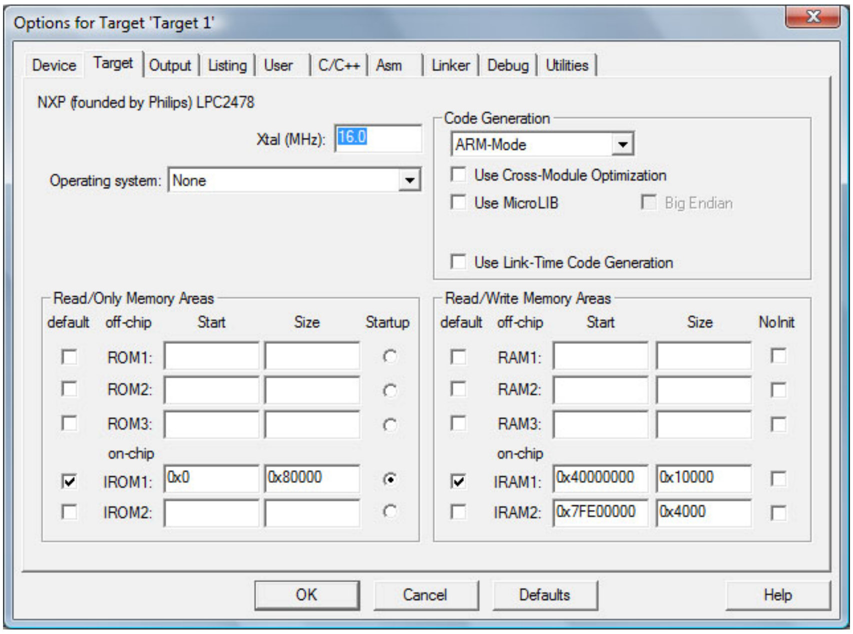
LPC2387
LPC2388
LPC2420
LPC2458
LPC2460
LPC2468
LPC2470
LPC2478
LPC2880
LPC2888
LPC2917
LPC2917/01
LPC2919
LPC2919/01
LPC2921

ARM7TDMI-S based high-performance 32-bit RISC Microcontroller with Thumb extensions.

512KB on-chip Flash ROM with In-System Programming (ISP) and In-Application Programming (IAP), 98KB RAM, CPU clock up to 72 MHz, On-chip crystal oscillator, On-chip 4MHz RC oscillator, On-chip PLL.

Enhanced Vectored Interrupt Controller, Ethernet 10/100 MAC with DMA, External Memory Controller for static and dynamic memories, USB 2.0 Full Speed Device Controller and Host/OTG Controller, CAN 2.0B with two channels, LCD Controller (STN and TFT), General purpose DMA controller, Four UARTs, one with full modem interface, Three I2C serial interfaces, Three SPI/SSP serial interfaces, I2S interface, SD/MMC memory-card interface, 10-bit ADC with 8 channels, 10-bit DAC, Four 32-bit timers with capture/compare, Watchdog Timer, PWM unit for three-phase motor control, Real Time Clock with optional

Secondly, set the XTAL frequency as 16 MHz. This is very important; one downloading the firmware via JTAG with wrong setting will cause the board not to be programmable anymore. If this situation occurs, one way to recover is to program it with Flash Magic.


Options for Target 'Target 1'
Device | Target | Output | Listing | User | C/C++ | Asm | Linker | Debug | Utilities
OK
Cancel
Defaults
Help

NXP (founded by Philips) LPC2478

Xtal (MHz): 16.0

Operating system: None

Code Generation

ARM-Mode

☐ Use Cross-Module Optimization

☐ Use MicroLIB ☐ Big Endian

☐ Use Link-Time Code Generation

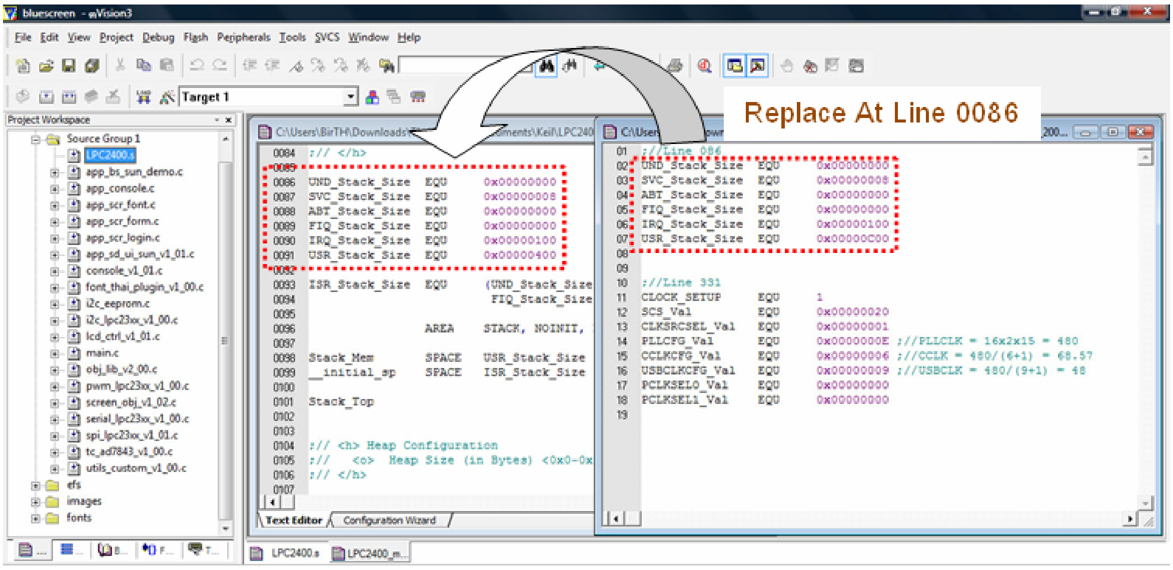
Read/Only Memory Areas

default	off-chip	Start	Size	Startup
☐	ROM1:			○
☐	ROM2:			○
☐	ROM3:			○
	on-chip			
☑	IROM1:	0x0	0x80000	●
☐	IROM2:			○

Read/Write Memory Areas

default	off-chip	Start	Size	NoInit
☐	RAM1:			☐
☐	RAM2:			☐
☐	RAM3:			☐
	on-chip			
☑	IRAM1:	0x40000000	0x10000	☐
☐	IRAM2:	0x7FE00000	0x4000	☐

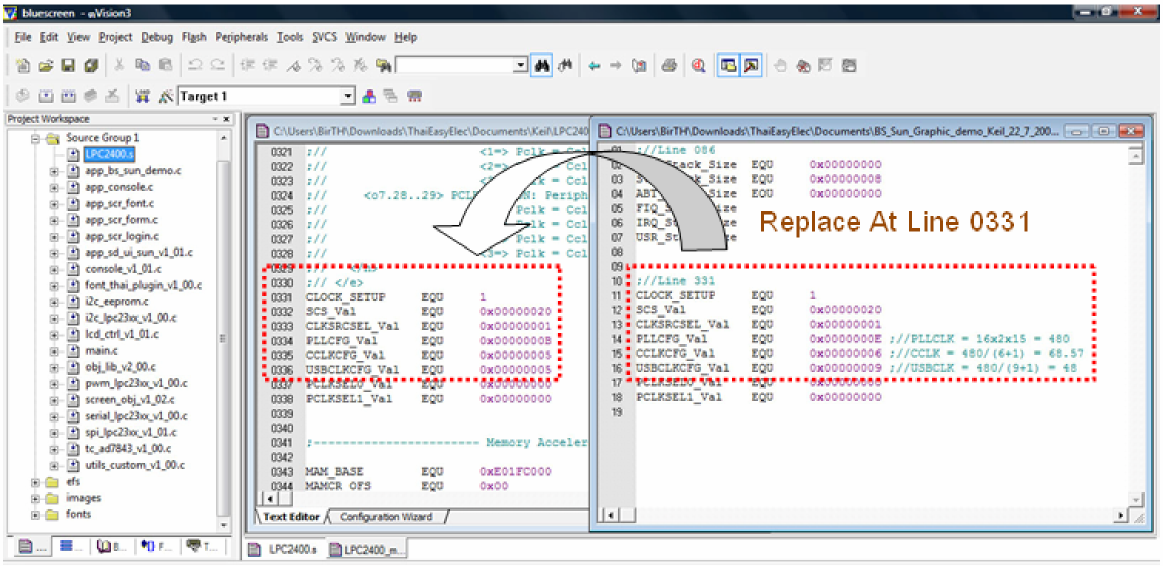
Thirdly, modify the LPC2400.s file along with the LPC2400_modification_note.s file you got from our website.



```
0084 ;// </h>
0085
0086 UND_Stack_Size  EQU     0x00000000
0087 SVC_Stack_Size  EQU     0x00000008
0088 ABT_Stack_Size  EQU     0x00000000
0089 FIQ_Stack_Size  EQU     0x00000000
0090 IRQ_Stack_Size  EQU     0x00000100
0091 USR_Stack_Size  EQU     0x00000400
0092
0093 ISR_Stack_Size  EQU     (UND_Stack_Size
0094                          FIQ_Stack_Size
0095
0096                 AREA    STACK, NOINIT,
0097
0098 Stack_Mem       SPACE   USR_Stack_Size
0099 __initial_sp    SPACE   ISR_Stack_Size
0100
0101 Stack_Top
0102
0103
0104 ;// <h> Heap Configuration
0105 ;//   <o>  Heap Size (in Bytes) <0x0-0x
0106 ;// </h>
0107
```

Replace At Line 0086

```
;//Line 086
UND_Stack_Size  EQU     0x00000000
SVC_Stack_Size  EQU     0x00000008
ABT_Stack_Size  EQU     0x00000000
FIQ_Stack_Size  EQU     0x00000000
IRQ_Stack_Size  EQU     0x00000100
USR_Stack_Size  EQU     0x00000C00


;//Line 331
CLOCK_SETUP     EQU     1
SCS_Val         EQU     0x00000020
CLKSRCSEL_Val   EQU     0x00000001
PLLCFG_Val      EQU     0x0000000E ;//PLLCLK = 16x2x15 = 480
CCLKCFG_Val     EQU     0x00000006 ;//CCLK = 480/(6+1) = 68.57
USBCLKCFG_Val   EQU     0x00000009 ;//USBCLK = 480/(9+1) = 48
PCLKSEL0_Val    EQU     0x00000000
PCLKSEL1_Val    EQU     0x00000000
```

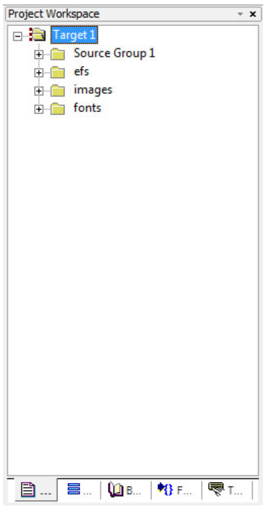


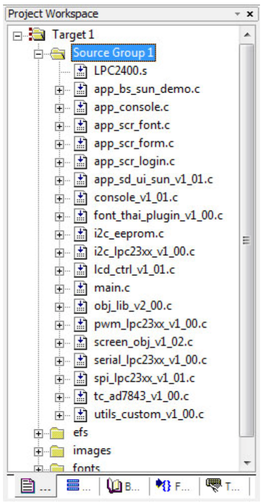
```
0330 ;// </e>
0331 CLOCK_SETUP       EQU     1
0332 SCS_Val           EQU     0x00000020
0333 CLKSRCSEL_Val     EQU     0x00000001
0334 PLLCFG_Val        EQU     0x0000000B
0335 CCLKCFG_Val       EQU     0x00000005
0336 USBCLKCFG_Val     EQU     0x00000005
0337 PCLKSEL0_Val      EQU     0x00000000
0338 PCLKSEL1_Val      EQU     0x00000000
0339
0340
0341 ;---------------------- Memory Acceler
0342
0343 MAM_BASE          EQU     0xE01FC000
0344 MAMCR_OFS         EQU     0x00
```

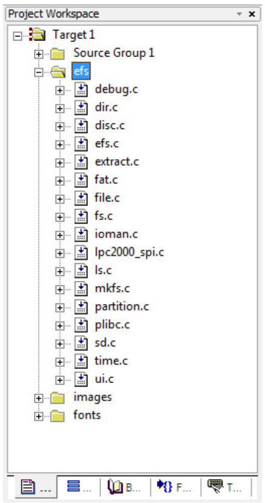
Replace At Line 0331

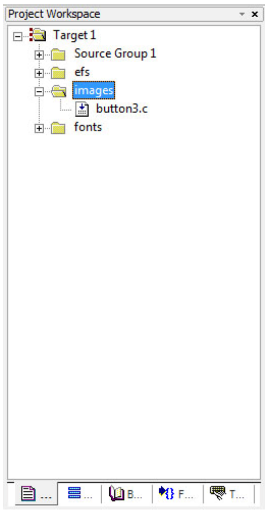
```
;//Line 086
ABT_Stack_Size    EQU     0x00000000
;//Line 331
CLOCK_SETUP       EQU     1
SCS_Val           EQU     0x00000020
CLKSRCSEL_Val     EQU     0x00000001
PLLCFG_Val        EQU     0x0000000E ;//PLLCLK = 16x2x15 = 480
CCLKCFG_Val       EQU     0x00000006 ;//CCLK = 480/(6+1) = 68.57
USBCLKCFG_Val     EQU     0x00000009 ;//USBCLK = 480/(9+1) = 48
PCLKSEL0_Val      EQU     0x00000000
PCLKSEL1_Val      EQU     0x00000000
```

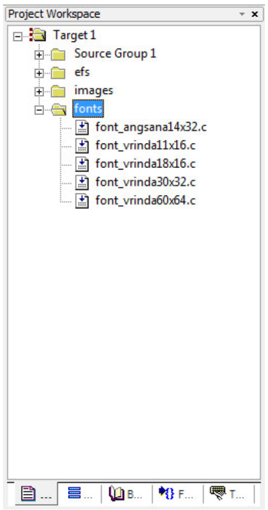
Then add all other source files (.c) in the project. And also add group "efs", "fonts" and "images" then add all source files in those folders to its group. For more detail about "efs" (stand for embedded file system), please see appendix.



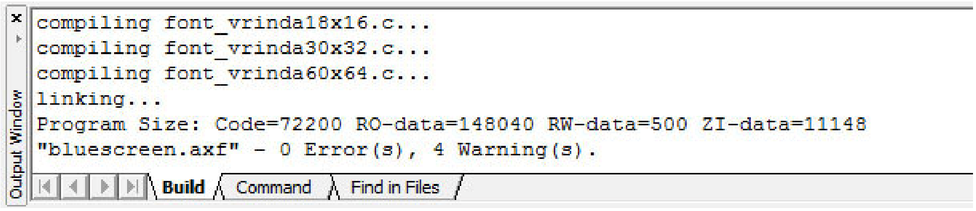








After these settings, the project should be compiled without any errors (some warnings may occur). The project workspace should be seen as below.



```
compiling font_vrinda18x16.c...
compiling font_vrinda30x32.c...
compiling font_vrinda60x64.c...
linking...
Program Size: Code=72200 RO-data=148040 RW-data=500 ZI-data=11148
"bluescreen.axf" - 0 Error(s), 4 Warning(s).
```

5.4 Source files description

Contents below are referenced to Graphic Library example; there may be some differences in other examples.

For more understanding in the content, user may also read Blue Screen board's manual, which uses the same core library.

Header files (only important files are shown)

1. hw_Blue Screen_sun.h : define hardware, for example : TC_CS is at port 2.2

   ```
   #define TC_CS_DPRT          FIO0DIR
   #define TC_CS_PRTS          FIO0SET
   #define TC_CS_PRTC          FIO0CLR
   #define TC_CS_PIN           10
   ```

   DPRT : direction port
   PRTS : port set
   PRTC : port clear
   PIN : pin number in the port

   These may look awkward, but it's very easy to change some pins.

2. app_config.h : this file is used to configure many parameters various on each project. Header files including are in here. And it is included in most of source file. So which declaration shared on more than one source file should be declared in here. Function headers of app_bs_sun_demo.c are also declared here.

3. All other header files are subjected to declare functions only.

Source files (only important files are shown)

1. main.c : background functions are here including timer interrupt service routine (T0_IRQHandler()), i/o initialization (io_init()). In case that user need to use more peripherals, PINSEL may be set in io_init(). Notice that AppInit() and AppRun() are called from main(), user application may be modified (or recreate) using these functions.

2. app_bs_sun_demo.c : most of application are here except the graphic user interface which is in app_scr_xxx.c

3. screen_obj.c : this is the code running in the background of screen object management. From the AppScanPen() in app_blue_screen.c, the screen position and the pen status are send to ScrObjDo(). This point is called “global position” because it refers the whole screen. In ScrObjDo(), each object are determined whether the screen position is in its area considering from its origin and its size. Then “.do()” of targeted object will be processed with “local position” got from minus of global position and the object’s origin.

4. app_scr_xxx.c : these files are related to each screen, each object’s parameter are defined here including origin, horizontal and vertical size, do() and draw() function and etc. A part of code from app_scr_login.c is shown below.

```
#define SO_KEYPAD              0 // (1)

ObjKeypad2Init(SO_KEYPAD,0,256);//(2)
so_obj[SO_KEYPAD].draw = keypad_draw;//(3)
so_obj[SO_KEYPAD].do_ = keypad_do;//(4)
so_obj[SO_KEYPAD].leave = user_turn_off_key;//(5)
```

Line (1) is to dedicate that, for this screen (MP3 player screen), keypad is the object number 1.
Line (2), keypad is initialized with ObjKeypad2Init() which is in obj_lib.c, 0 and 256 are position of the object in the screen. With standard object library, some parameters are fixed. Anyway, for customized object, all parameter must be set.
Line (3), (4) and (5) define functions that would be operated on corresponding event.

With screen_obj_v1_01 or later, all functions in so_obj structure are initialized to null function. So they don’t need to be assigned in case that they are not used.

Note that “so_obj” is structure type variable. User may change its parameter when need. So in case that user designs more than one screen in an application. The array “so_obj” can be redefined again and again for each screen.

For more detail about “so_obj” please see in screen_obj.h.

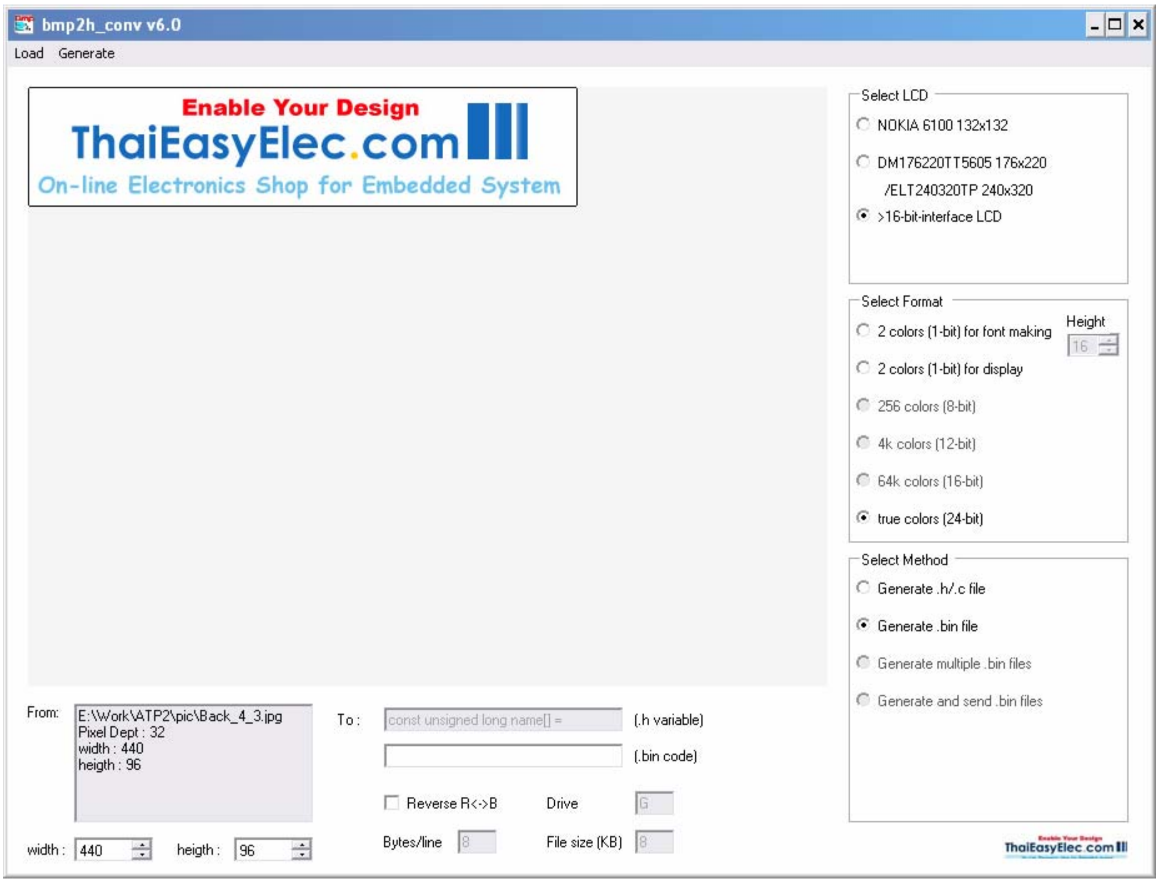
5.5 Image generating

In the demo application, most of images seen on the screen are generated from JPEG files or Bitmap files with the software called “bmp2h_conv”. The last version, which is 6.0, supports 24-bit color format and can be downloaded at our website in product’s page. This part of manual shows step by step how to implement these pictures into your project.

Please also understand about format that the picture would be generated to. With Blue Screen board (2.8-inch LCD); the color format is 16-bit. But with Blue Screen SUN board, it’s 24-bit, just like on your PC. Anyway, in embedded system’s world, it must be store in 32-bit memory. So one pixel picture would be store in 4 bytes of memory.

Firstly, create your image using Paint or whatever you like.

Secondly, run bmp2h_conv and load your image. Set parameters as shown below except of “Select Method”.



Select LCD
○ NOKIA 6100 132x132
○ DM176220TT5605 176x220 /ELT240320TP 240x320
● >16-bit-interface LCD

Select Format
○ 2 colors (1-bit) for font making
Height 16
○ 2 colors (1-bit) for display
○ 256 colors (8-bit)
○ 4k colors (12-bit)
○ 64k colors (16-bit)
● true colors (24-bit)

Select Method
○ Generate .h/.c file
● Generate .bin file
○ Generate multiple .bin files
○ Generate and send .bin files

From: E:\Work\ATP2\pic\Back_4_3.jpg
Pixel Dept : 32
width : 440
heigth : 96

To : const unsigned long name[] = (.h variable)
(.bin code)

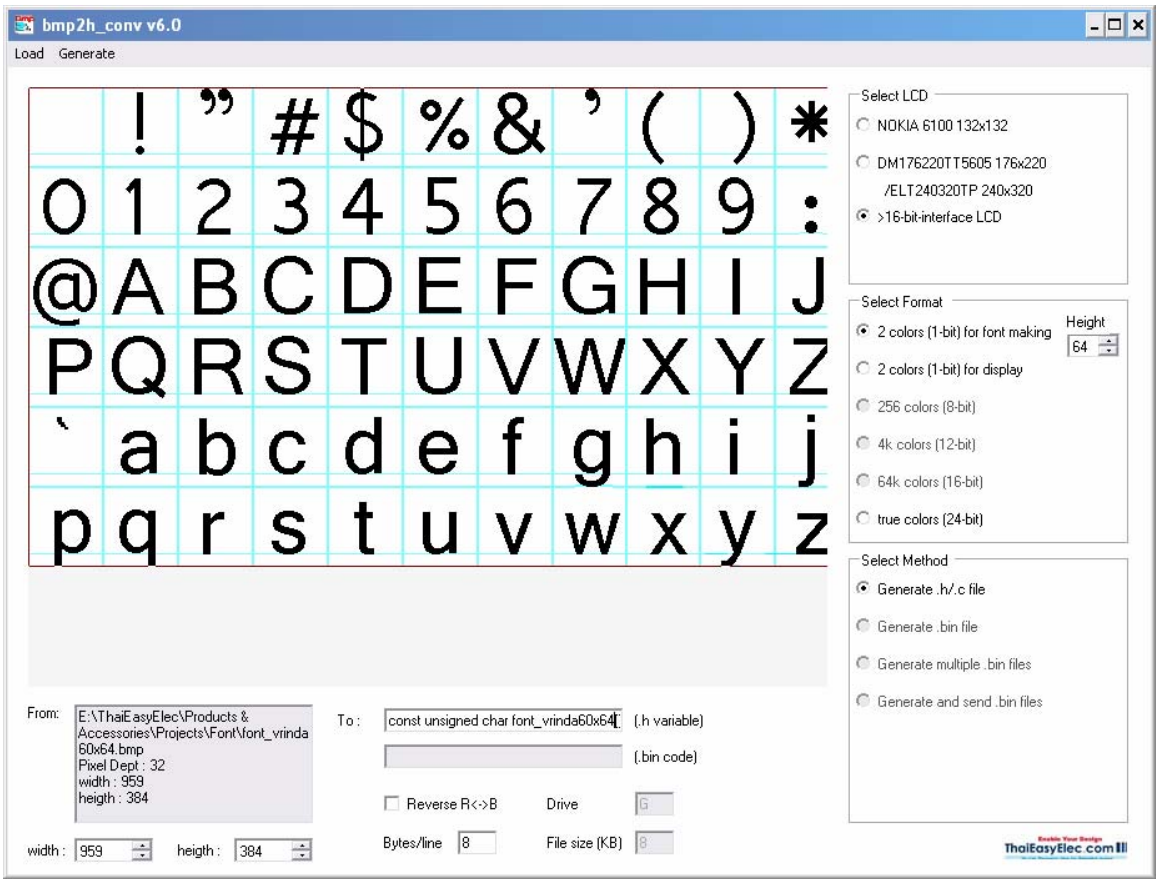
□ Reverse R<->B
Drive G
Bytes/line 8
File size (KB) 8

width : 440
heigth : 96

In “Select Method” option, select what you need; to store image in code memory, use .h or .c file (.h is to be included while .c is to be added to project, the difference in uses is, whenever the image would be used in one source file, .h can used. Unless, use .c and then declare them with “extern” prefix in app_config.h, then you can use it anywhere in the project. For example, fonts are used many times in many files, it must be added like a .c file.)
Generating to .bin files is used for storing image on SD card. This is, in case of large image that would be used as background. Or in case that you are having arrays of images (see Catalog demo example for more details). Note that a full screen image for 7-inch LCD would take 800x480x4 bytes. That’s 1.5MB and is absolutely not be able to store in 512K flash memory.

Then, click “Generate” and specify the target file name.

To make a font, see settings below.



Select LCD
○ NOKIA 6100 132x132
○ DM176220TT5605 176x220 /ELT240320TP 240x320
● >16-bit-interface LCD

Select Format
● 2 colors (1-bit) for font making
○ 2 colors (1-bit) for display
○ 256 colors (8-bit)
○ 4k colors (12-bit)
○ 64k colors (16-bit)
○ true colors (24-bit)
Height: 64

Select Method
● Generate .h/.c file
○ Generate .bin file
○ Generate multiple .bin files
○ Generate and send .bin files

From: E:\ThaiEasyElec\Products & Accessories\Projects\Font\font_vrinda60x64.bmp
Pixel Dept : 32
width : 959
heigth : 384

To : const unsigned char font_vrinda60x64[] (.h variable)
(.bin code)
□ Reverse R<->B
Drive: G
Bytes/line: 8
File size (KB): 8
width : 959
heigth : 384

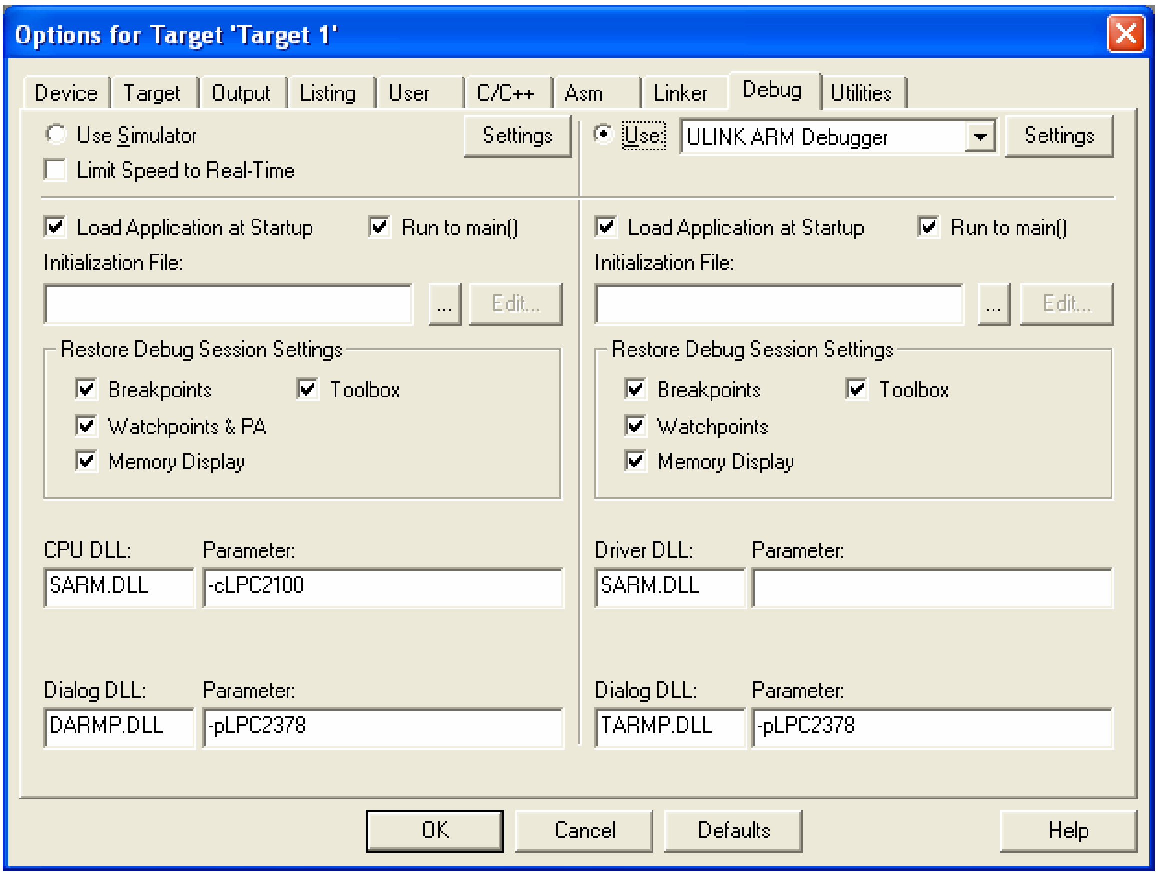
Notice on the “Height” parameter in “Select Format” option. This must be set to match your font images. It could only be in multiples of 8.

5.6 Programming, Debugging and Command line interface

5.6.1 JTAG Interface

Blue Screen SUN board supports JTAG interface for debugging and programming. Many programmers can be used corresponding to your IDE.

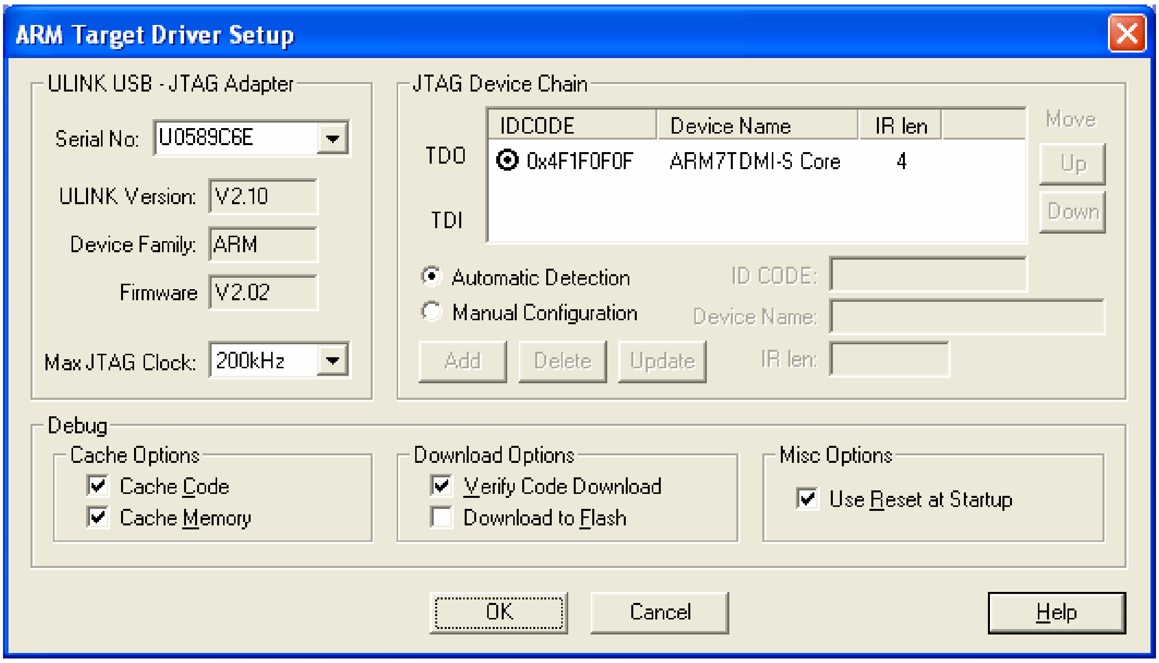
For Keil's software, on the target's option select 'Debug' and configure the flash programmer as follow.



○ Use Simulator
□ Limit Speed to Real-Time
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watchpoints & PA
☑ Memory Display
CPU DLL: SARM.DLL Parameter: -cLPC2100
Dialog DLL: DARMP.DLL Parameter: -pLPC2378

● Use: ULINK ARM Debugger
☑ Load Application at Startup ☑ Run to main()
Initialization File:
Restore Debug Session Settings
☑ Breakpoints ☑ Toolbox
☑ Watchpoints
☑ Memory Display
Driver DLL: SARM.DLL Parameter:
Dialog DLL: TARMP.DLL Parameter: -pLPC2378

Also select 'Settings' and set MAX JTAG Clock not over than 200kHz if you are using mini N-Link. Faster speed can be set with ULink2 or compatible programmer.



ARM Target Driver Setup

ULINK USB - JTAG Adapter

Serial No: U0589C6E

ULINK Version: V2.10

Device Family: ARM

Firmware: V2.02

Max JTAG Clock: 200kHz

JTAG Device Chain

	IDCODE	Device Name	IR len
TDO	● 0x4F1F0F0F	ARM7TDMI-S Core	4
TDI			

● Automatic Detection

○ Manual Configuration

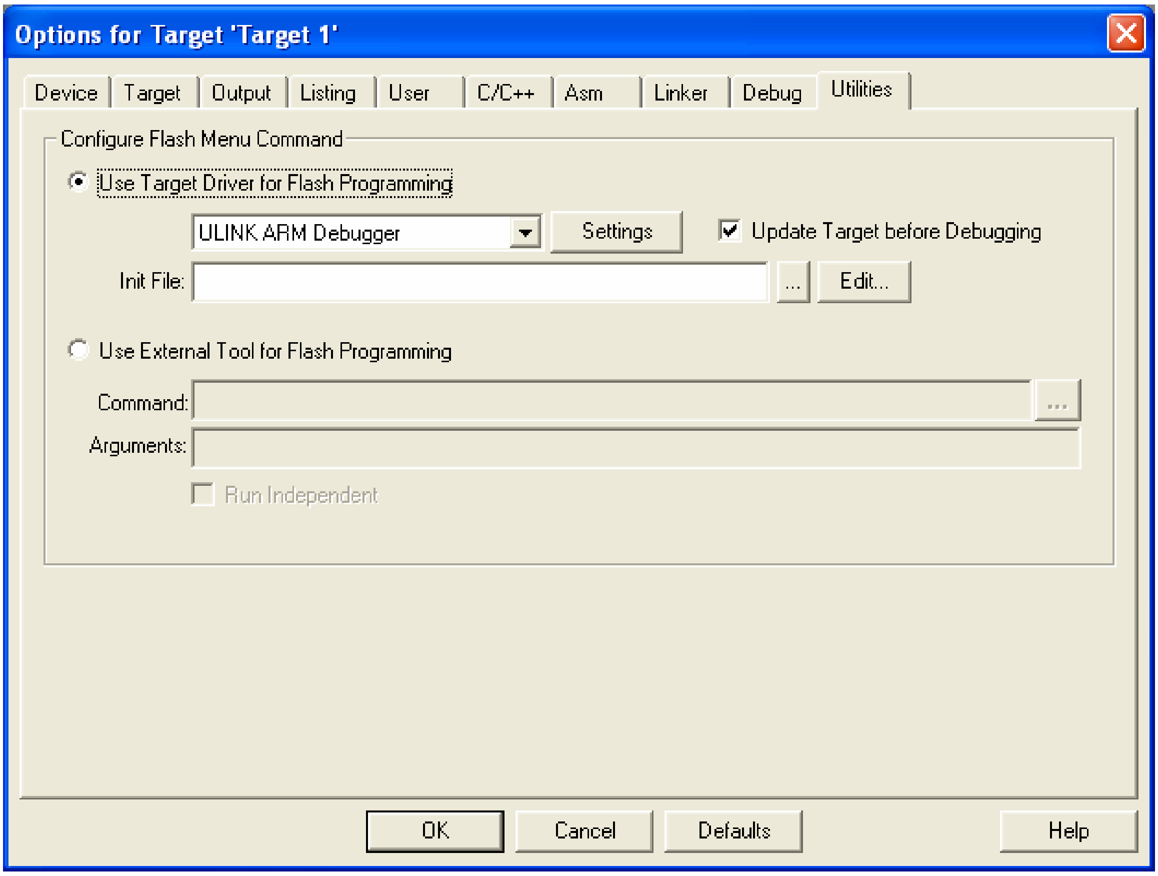
Debug

Cache Options
☑ Cache Code
☑ Cache Memory

Download Options
☑ Verify Code Download
☐ Download to Flash

Misc Options
☑ Use Reset at Startup

Back to target's option select Utilities and configure as follow.



Options for Target 'Target 1'

Configure Flash Menu Command

● Use Target Driver for Flash Programming

ULINK ARM Debugger

☑ Update Target before Debugging

Init File:

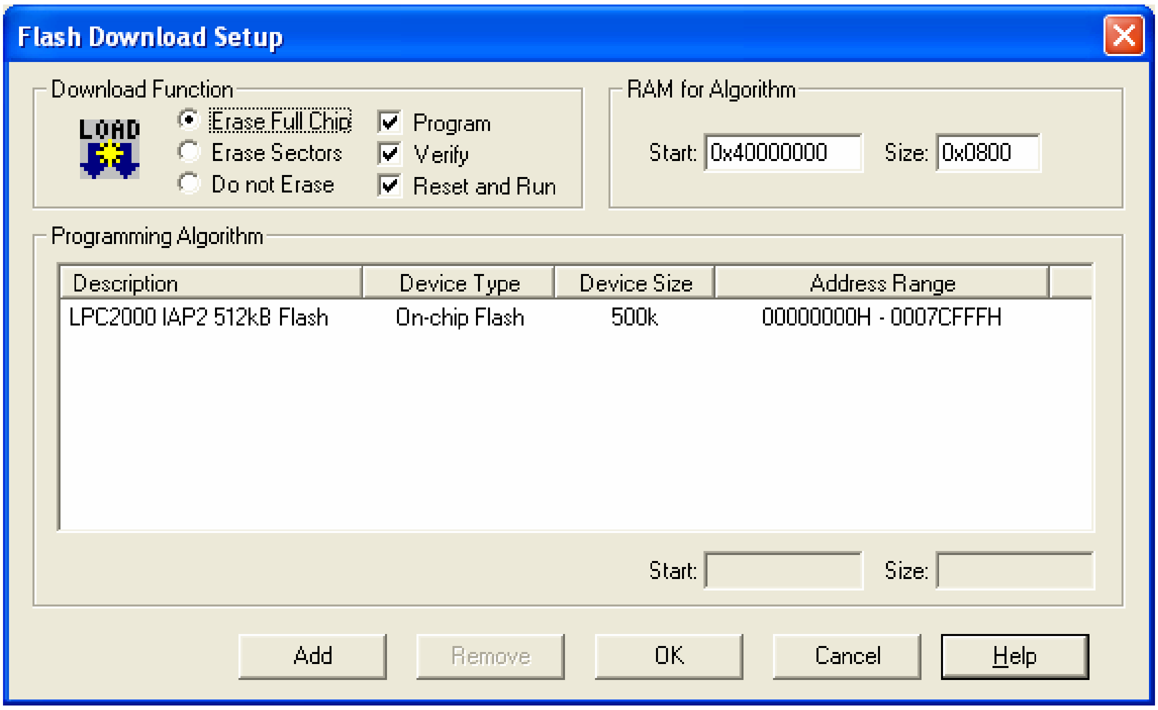
○ Use External Tool for Flash Programming

Command:

Arguments:

☐ Run Independent

Select setting and add programming algorithm as follow.



Flash Download Setup

Download Function

● Erase Full Chip
○ Erase Sectors
○ Do not Erase

☑ Program
☑ Verify
☑ Reset and Run

RAM for Algorithm

Start: 0x40000000 Size: 0x0800

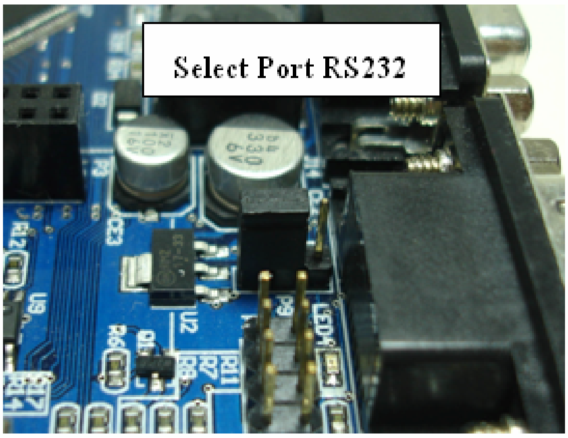
Programming Algorithm

Description	Device Type	Device Size	Address Range
LPC2000 IAP2 512kB Flash	On-chip Flash	500k	00000000H - 0007CFFFH

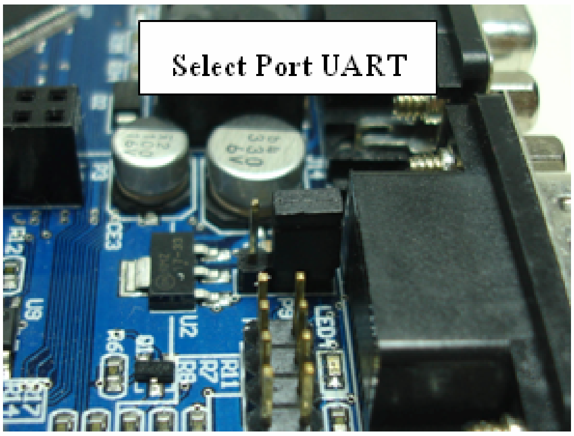
Start: Size:

5.6.2 Serial port

Command line interface can be connected using COM0 (as CONS_SER is defined to '0' in app_config.h). Anyway, the board provides COM0 in 2 types; RS232 and TTL. There is a jumper (P9) to be set, see pictures below.

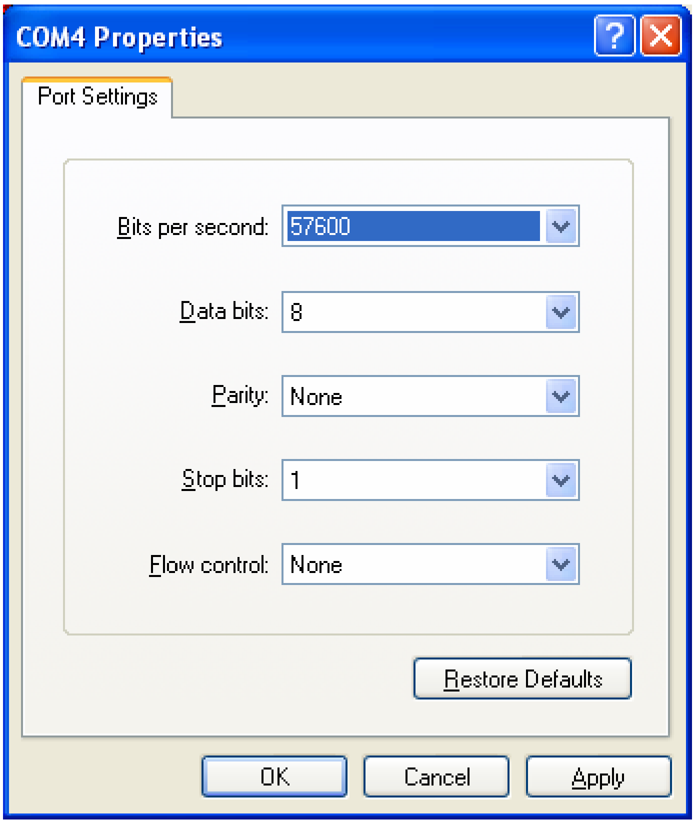


Select Port RS232



Select Port UART

To communicate the board with command line interface, HyperTerminal can be used. Set the COM port as follow.



COM4 Properties

Port Settings

Bits per second: 57600

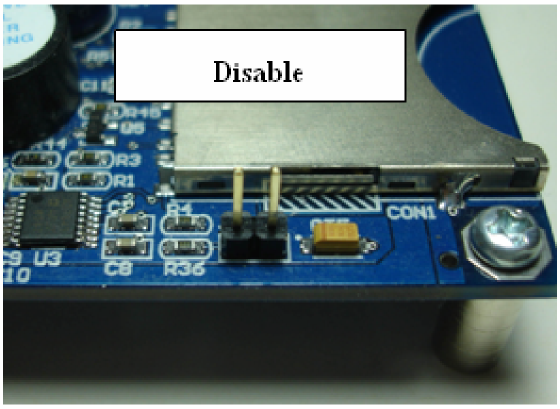
Data bits: 8

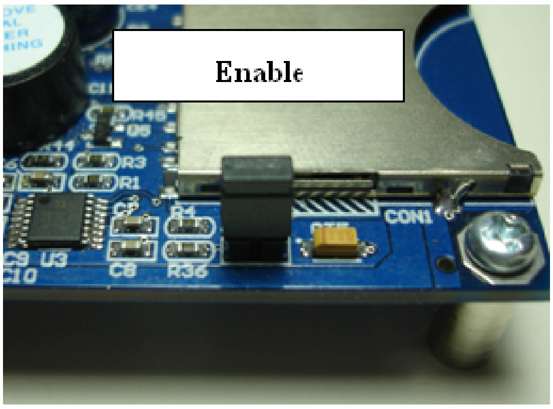
Parity: None

Stop bits: 1

Flow control: None

The serial port 0 also served for ISP (In-system programming), which user can use software like Flash Magic to program the MCU. To do this, place a jumper on J9 (see pictures below) and then press reset. Now the MCU should starts with ISP boot loader.





Please also remove the jumper when programming finished and then reset the board again.

Appendix

1. TS_MODE (refers to lcd_ctrl.h)

TS_MODE is display mode used in functions start with TSLCD. Now there are 3 modes available, TS_MODE_NORMAL, TS_MODE_INVERSE and TS_MODE_FULL.

- TS_MODE_NORMAL : To displaying a circle or some images having white background, user may want this white background to be transparent. Using this mode, all pixel having value 0xFFFF will considered as 'background' and it will be display as read-back color. The example of this mode displaying is the volume bar. There you can see blue background instead of white.

- TS_MODE_INVERSE : As it's name "inverse". The circle or rectangular drawn will have inverse color to the old color. Color parameter sent to the function will be ignored.

- TS_MODE_FULL : In case of showing a text message again and again in the same area. You need the use this mode as the blank space will be filled with background color. Anyway, displaying an image (TSLCDShowPic2()) with this mode all color from original image code will be display including white color.

Note that a rectangular doesn't have blank space; in this case TS_MODE_NORMAL and TS_MODE_FULL have same effect.

2. Pen status (refers to app_config.h)

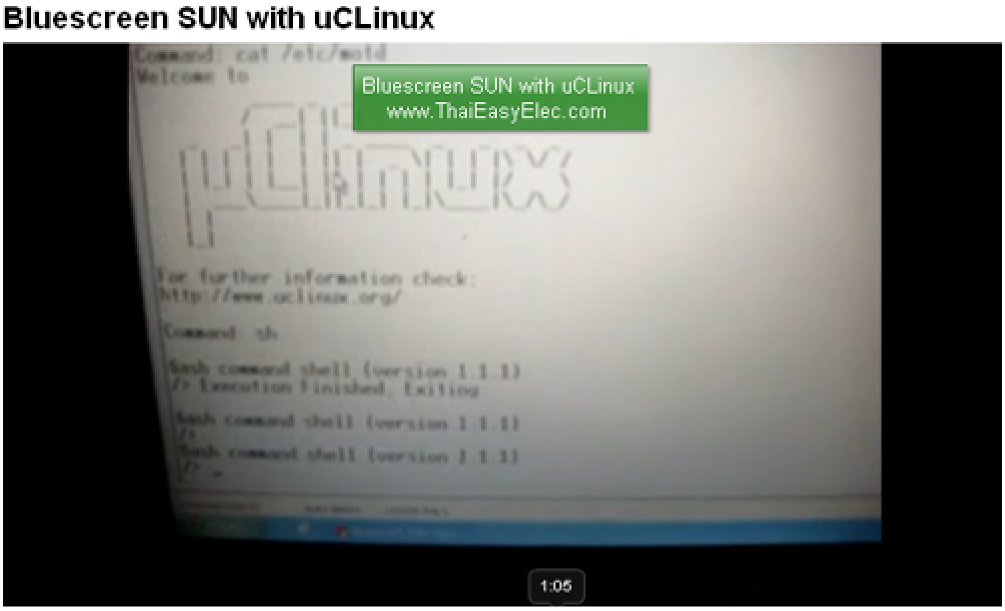
Pen status or in code "pstatus" means the current state of pen. This is useful parameter sent to ScrObjDo(). There are 4 available statuses:

- PST_NOTFOUND : means that the screen is not pressed.
- PST_DOWN : occurs once the screen is pressed.
- PST_HOLD : occurs continuously while the screen is pressed.
- PST_UP : occurs once the screen is released.

3. Embedded file system library (EFSL)

Embedded file system library using in this application is free library downloadable from http://sourceforge.net/projects/efsl/ . It's limited to SPI only (LPC23xx's MCI is not supported). It's very easy to port it to your existing hardware. AVR, NXP's ARM7, and some other platform have already had example.

In this application lpc200_spi.c is modified from original code. It is the bridge between our SPI library and EFSL. Other files is also modified (some lines are commented) to decrease warnings and errors.



Bluescreen SUN with uCLinux

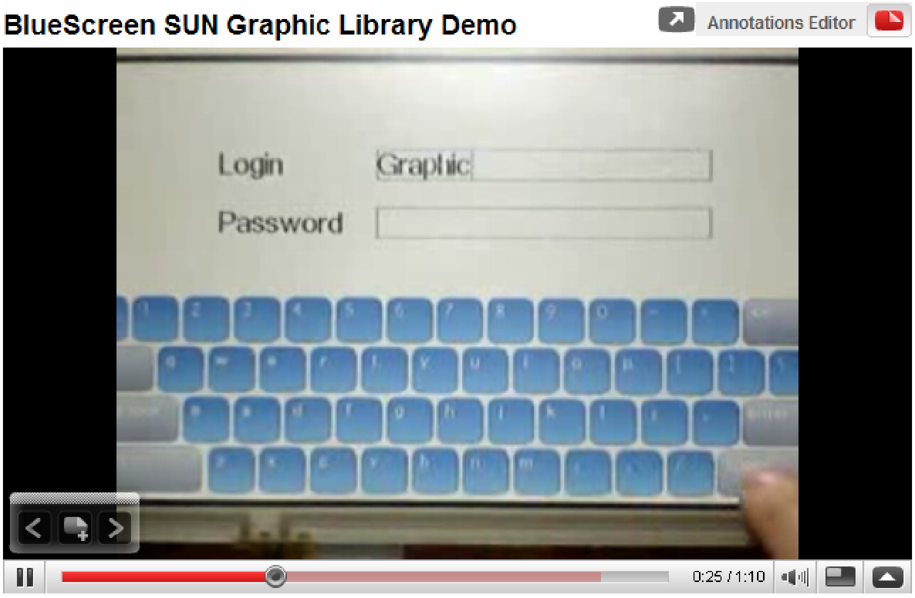
```
Command: cat /etc/motd
Welcome to

For further information check:
http://www.uclinux.org/

Command: sh

Sash command shell (version 1.1.1)
/> Execution Finished, Exiting

Sash command shell (version 1.1.1)
/>
Sash command shell (version 1.1.1)
/> _
```

BLUESCREEN SUN with uCLinux
http://www.youtube.com/watch?v=NMIM8vTSMuo
Booting uClinux from SD Card , Bootloader copy kernel image from SD Card to SDRAM and Boot

BlueScreen SUN coming up!!!

BLUESCREEN SUN Point Of Sale Demo
http://www.youtube.com/watch?v=7DKAqvJlyLs

BLUESCREEN SUN with Ethernet Module



Login Graphic
Password

BlueScreen SUN Graphic Library Demo
http://www.youtube.com/watch?v=B5bwluqbr9w
LPC2478 with 7 inches touch screen LCD with graphic library. Source code available now!!!

MORE Application , please visit

http://www.ThaiEasyElec.com

http://www.ThaiEasyElec.net

BLOG SUPPORT

http://bluescreen-sun-etee012.blogspot.com

Prepared by
ThaiEasyElec.com
Venus Supply Co.,Ltd
196/1, Soi Thedsaban-Nimit-Nau 8,
Thedsaban-Nimit-Nau Road, Ladyao,
Chatuchak, Bangkok 10900
Tel. +(66)2954-2408 , Fax. +(66)2953-8443
Email Sales@thaieasyelec.com
Support@thaieasyelec.com