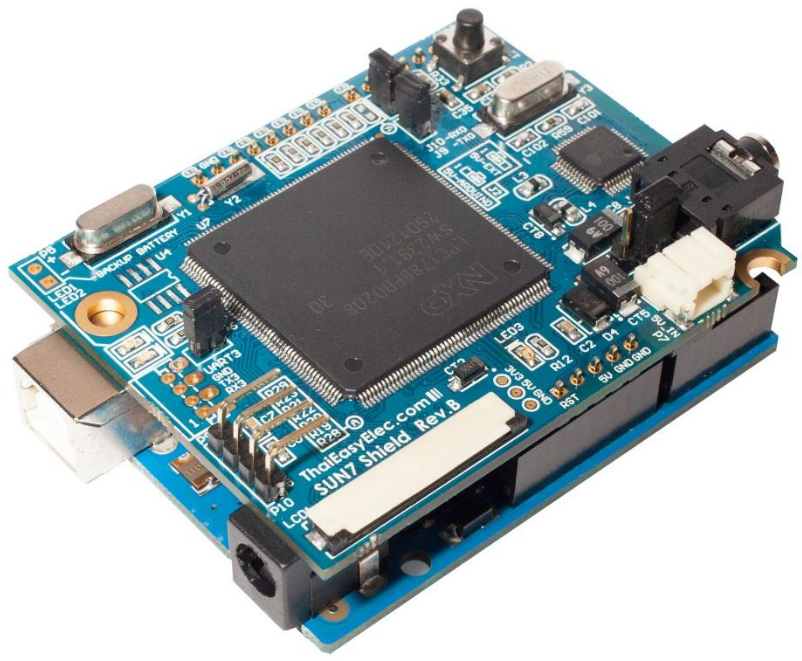
SUN7 Shield

User Manual

v1.0

Enable Your Design
ThaiEasyElec.com
On-line Electronics Shop for Embedded System



Revision History

Version	Date	Changes
1.0	11 Jan 2013	Original version

Contents

1. Introduction

SUN7 Shield was designed to allow Arduino boards to run graphic-based application using 4.3” or 7” touch screen LCD. With GUI Engine running on the shield, the user can create and configure GUI screen by screen on SUN7 Studio which is software created by ThaiEasyElec.com. The user get script folder contains script, images and sounds from SUN7 Studio and put it in the SD card to run the shield. Then all objects on the screen can be controlled by Arduino through a serial port. Moreover, the SUN7 shield contains MP3 decoder, RTC and support firmware upgrade through SD card.

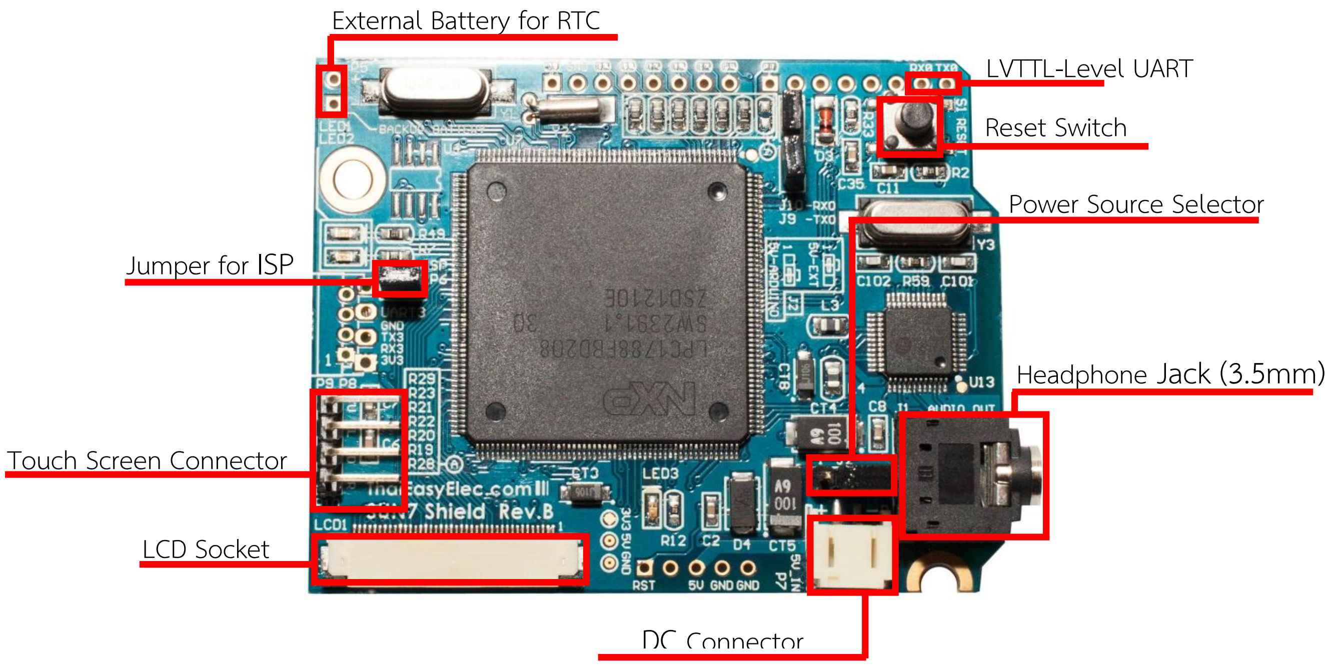
When the SUN7 shield starts up, it reads GUI script from installed SD card. It loads images from SD card to SDRAM according to the script. Prior to start GUI, some messages are sent to the Arduino so the Arduino can initialize some objects using commands on purpose (e.g. change some images, disable some buttons). Then GUI is started, and while GUI is running, responsive messages are sent from the SUN7 shield to the Arduino when any events occur. All commands and messages are text-based.

2. Features

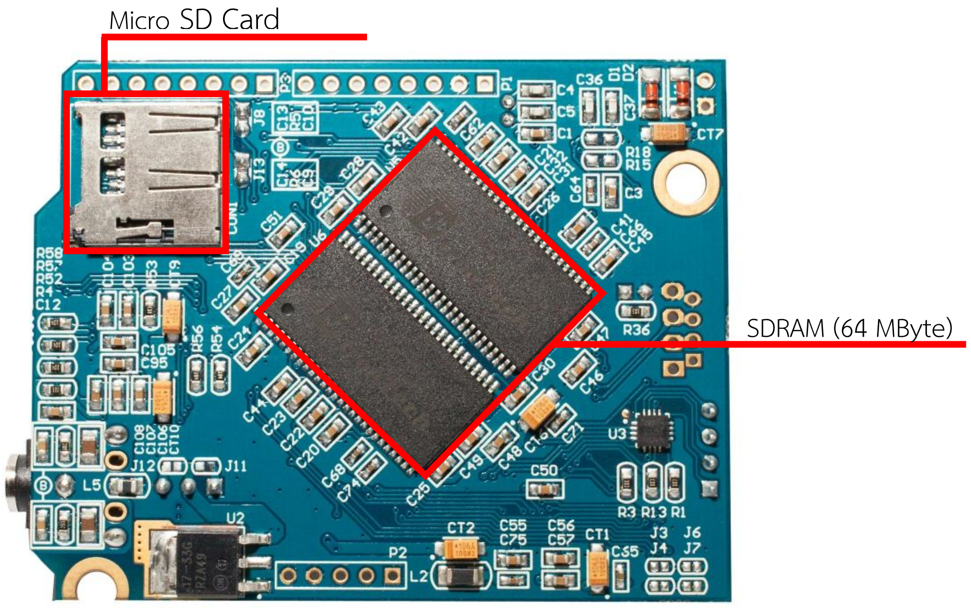
- NXP’s ARM Cortex-M3 LPC1788
- 12 MHz crystal
- Maximum of 64 MB memory with 2 of 16-bit EtronTech’s EM63A165TS-5G SDRAM (32-bit)
- Connector for 800x480 pixels (wide screen 7”) TFT LCD
- Connector for 4-wire resistive touch screen panel
- Touch screen controller IC, STMPE610
- Micro SD card socket (SPI interface) supports SDHC (high capacity type)
- On-chip 4KB EEPROM in LPC1788
- UART with LVTTL-level (3.3V with 5V tolerant)
- Built-in VS1011E MP3 decoder with 3.5 mm headphone jack
- 5VDC power supply terminal

3. Peripherals

3.1 Layout



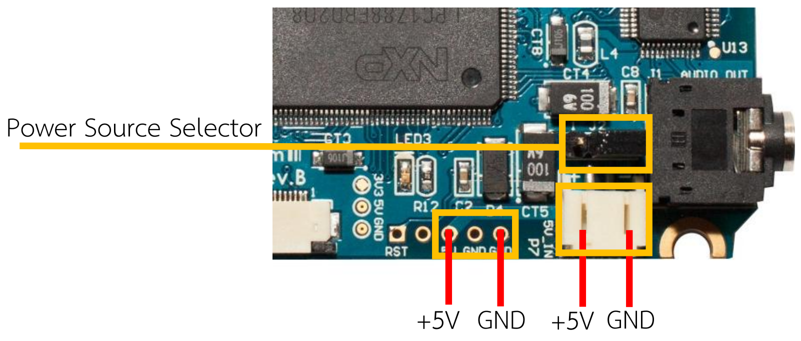
Top view



Bottom view

3.2 Connector Descriptions

3.2.1 Power Supply

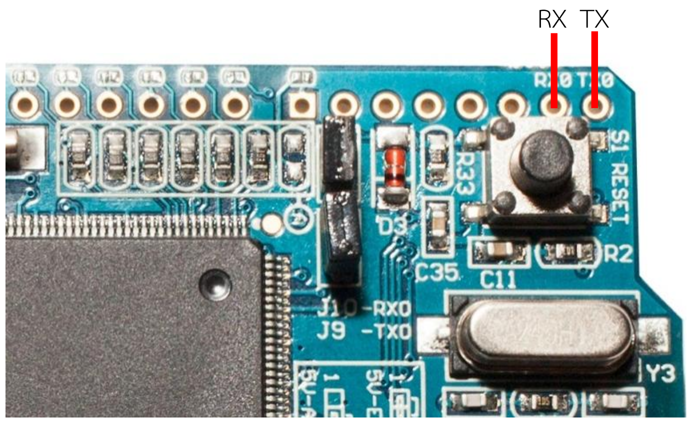


The SUN7 Shield can be powered from Arduino boards or 5V external power supplies. The selection can be made by installing a jumper on J2 as described below.

Selector	Description
[■■]■	Use external power supply
■[■■]	Use power supply on Arduino

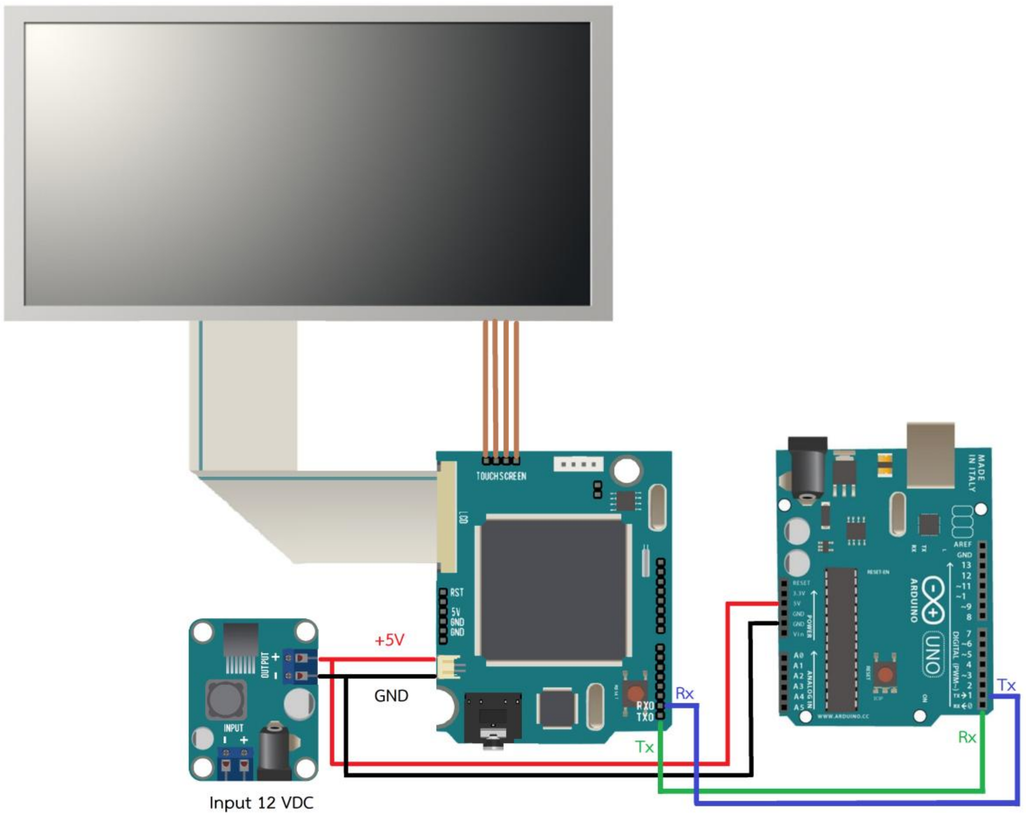
Note that LCDs consume much current so using an external power supply is recommended since the supply circuit on Arduino.

3.2.2 UART with LVTTL-level (3.3V with 5V Tolerant)



4. Communication with Arduino

4.1 Interconnection with Arduino



4.2 Serial Port Settings

Baud Rate:	9600
Data Bits:	8
Parity Bits:	none
Stop Bits:	1

4.3 Format

```
Command (Sent from Arduino)
            command parameter1 parameter2 parameter3 … \r\n

Response (Sent from SUN7 Shield after command processed)
            <OK>\r\n (only when response is enabled) or
            <ERROR>\r\n (only when response is enabled) or
            <response> for inquiry commands

Message (Sent from SUN7 shield)
            <parameter1 parameter2 parameter3 …>\r\n
```

Remark:	1. Space width can be any size.
	2. Spaces are used as separators, DO NOT use space in any object names.
	3. Commands are not case-sensitive but parameters are case-sensitive.
	4. Response is disabled by default; see GUIResponse command for more detail.

5. Commands and Messages

5.1 GUI Commands and Messages

5.1.1 Initial Messages

SETUPGUI: Sent out from the SUN7 Shield to inform that reading script is finished but GUI is not yet shown. The user can program Arduino to configure objects at this state. A timer is set when this message is sent to wait for user commands. And there is 1- second timeout after last command received, after timeout, STARTGUI will be sent.

```
            <SETUPGUI>\r\n
```

STARTGUI: Sent out from the SUN7 Shield after SETUPGUI to inform that GUI is started.

```
            <STARTGUI>\r\n
```

5.1.2 Main Settings

GUIResponse: Enable or disable command response.

```
            GUIResponse state
state:      1 = Enable
            0 = Disable (default)
example:    GUIResponse 1
```

GUIInitKeypad: Initialize keypad, use only once after <SETUPGUI> received. Note that the keypad style is fixed by the keypad images from example script folders.

```
            GUIInitKeypad
```

5.1.3 Screen & Popup Window

GUIGotoScr: Change screen to target screen.

```
            GUIGotoScr screen_name
screen_name: target screen name
```

GUIOpenPopup: Open popup window.

```
            GUIOpenPopup popup_screen_name
popup_screen_name: target popup screen name
```

GUIClosePopup: Close popup window.

```
             GUIClosePopup
```

NAMESCS: Message sent from SUN7 Shield to inform that screen is changed to a new one.

```
             <NAMESCS screen_name>\r\n
screen_name: new screen name
```

KEYPAD: Message sent from SUN7 Shield to inform that the keypad is pressed.

```
             <KEYPAD char>\r\n
char:        pressed character
```

5.1.4 Language

GUISetLang: Set language mode.

```
             GUISetLang lang
lang:        language mode, must be set in 1<<n where n: 0-7
```

GUIGetLang: Get language mode.

```
             GUIGetLang lang
lang:        language mode
```

5.1.5 Sound

GUISndClrList: Stop MP3 playback and clear all MP3 added to the list.

```
             GUISndClrList
```

GUISndAddList: Add sound to list. Using this function alone will wait until playing sound finishes. To play new sound instantly, use GUISndClrList beforehand.

```
             GUISndAddList snd_no segment_no
snd_no:      sound ID
segment_no: segment number of sound (use 0 for non-segmented sound)
```

GUISndSetOffsetVol: Before a sound played, the sum of offset volume and individual volume (set from script) will be used to set the chipset.

```
             GUISndSetOffsetVol offset_vol
offset_vol: offset volume
```

GUISndOn: Sound is turned on by default; this function turns sound on if GUISndOff is used.

```
             GUISndOn
```

GUISndOff: Turn off sound. Using audio commands cannot stop playing file from sound plug-in. This function will stop current playing and prohibit list addition on the beginning of screens.

```
            GUISndOff
```

GUISndGetCurrent: Get playing sound ID.

```
            GUISndGetCurrent
return:     <ID of the sound being played>\r\n
```

SNDEND: Message sent from SUN7 Shield to inform that playing sound added to the screen is finished.

```
            <SNDEND>
```

5.1.6 Button

GUIEnableBt: Enable a button specified by name so it becomes responsible for presses.

```
            GUIEnableBt name_bt
name_bt:    button name
```

GUIDisableBt: Disable a button specified by name so it becomes irresponsible for presses.

```
            GUIDisableBt name_bt
name_bt:    button name
```

GUISkipBt: Make a button specified by name to be skipped. It won't be shown whatever its status is. Use GUIUnSkipBt to cancel the effect.

```
            GUISkipBt name_bt
name_bt:    button name
```

GUIUnSkipBt: Unskip a button specified by name.

```
            GUIUnSkipBt name_bt
name_bt:    button name
```

GUIChangeImgBt: Change image for a button state using image ID. The new image must have the same size with the old one.

```
            GUIChangeImgBt name_bt state_bt ID
name_bt:    button name
state_bt:   0 = disable image
            1 = normal image
            2 = press image
ID:         image ID (from script)
```

GUIConfigBt: Configure responsive message for a button. Messages will be sent out only when enabled actions occur. All actions are enabled by default. Message patterns are described afterward.

```
            GUIConfigBt name_screen name_bt action_bt
name_screen:screen name
name_bt:    button name
action_bt:  0 = DO (every 10ms)
            1 = PRESS
            2 = RELEASE
            3 = PRESS and RELEASE
            4 = Enable All (Default)
            5 = Disable All
```

Action Message:

```
            <name_bt action_bt>\r\n
name_bt:    button name
action_bt:  button's action (DO/PRESS/RELEASE)
```

KEYBT: When the button is set as a key button and it's pressed, KEYBT message is also sent with action message.

```
            <KEYBT character>\r\n
character:  character set for the button, sent as hex value
```

SETLANG: When the button is set as a change-language button and it's pressed, SETLANG message is also sent with action message.

```
            <SETLANG new_language>\r\n
new_language:new language mode, sent as hex value
```

5.1.7 Image

GUIEnableImg: Enable an image specified by name so it is displayed on the LCD.

```
            GUIEnableImg name_img
name_img:   image box name
```

GUIDisableImg: Disable an image specified by name so it is not displayed on the LCD.

```
            GUIDisableImg name_img
name_img:   image box name
```

GUIChangeImg: Change image in an image box specified by name with ID of new image. The new image must have the same size with the old one.

```
            GUIChangeImg name_img ID
name_img:   image box name
ID:         image ID (from script)
```

5.1.8 Label

GUISkipLbl: Make a label specified by name to be skipped. Use GUIUnskipLbl to cancel the effect.

```
            GUISkipLbl name_lbl
name_lbl:   label name
```

GUIUnSkipLbl: Unskip a label specified by name.

```
            GUIUnSkipLbl name_lbl
name_lbl:   label name
```

5.1.9 Textbox

GUIAddTxt: Add text to a textbox with maximum length set from script.

```
            GUIAddTxt name_txt text
name_txt:   text box name
text:       text
```

GUIClrTxt: Clear text on a textbox specified by name.

```
            GUIClrTxt name_txt
name_txt:   textbox name
```

GUIGetStrTxt: Get text on a textbox specified by name..

```
            GUIGetStrTxt    name_txt
name_txt:   text box name
return:     <text>\r\n
```

GUISkipTxt: Make a button specified by name to be skipped. It won't be shown whatever its status is. Use GUIUnSkipTxt to cancel the effect.

```
            GUISkipTxt name_txt
name_txt:   textbox name
```

GUIUnSkipTxt: Unskip a button specified by name.

```
            GUIUnSkipTxt name_txt
name_txt:   textbox name
```

GUIConfigTxt: Configure responsive message for a textbox. Messages will be sent out only when enabled actions occur. All actions are enabled by default. Message patterns are described afterward.

```
            GUIConfigTxt name_screen name_txt action_txt
name_screen:screen name
name_txt:   text box name
action_txt: 0 = PRESS (Default)
            1 = Disable All
```

Action Message:

```
            <name_txt action_txt>\r\n
name_txt:   text box name
action_txt: text box’s action (PRESS)
```

5.1.10 Table

GUIWriteTab: Write text in table with options defined by script.

```
            GUIWriteTab name_tab row column text
name_tab:   table name
row:        row number
column:     column number
text:       text
```

GUIWriteTab2: An extended version of GUIWriteTable. With this one, parameters can be specified beyond settings from script.

```
            GUIWriteTab2 name_tab row column text back_color font_color
name_tab:   table name
row:        row number
column:     column number
text:       text
back_color: background color
font_color: font color
```

GUISkipTab: Make a button specified by name to be skipped. It won’t be shown whatever its status is. Use GUIUnSkipTab to cancel the effect.

```
            GUISkipTab name_tab
name_tab:   table name
```

GUIUnSkipTab: Use this function with tables to cancel the effect of GUISkipTable.

```
            GUIUnSkipTab name_tab
name_tab:   table name
```

GUIConfigTab: Configure responsive message for a table. Messages will be sent out only when enabled actions occur. All actions are enabled by default. Message patterns are described afterward.

```
            GUIConfigTab name_screen name_tab action_tab
name_screen:screen name
name_tab:   table name
action_tab: 0 = DO (every 10ms)
            1 = PRESS
            2 = Enable All (Default)
            3 = Disable All
```

Action Message:

```
            <name_tab action_tab row column>\r\n
name_tab:   table name
action_tab: table's action (DO/PRESS)
row:        pressed row
column:     pressed column
```

5.1.11 Percent Bar

GUIEnableBar: Enable a percent bar specified by name.

```
            GUIEnableBar name_bar
name_bar:   percent bar name
```

GUIDisableBar: Disable a percent bar specified by name.

```
            GUIDisableBar name_bar
name_bar:   percent bar name
```

GUISetValBar: Set a percent bar's value.

```
            GUISetValBar name_bar value
name_bar:   percent bar name
value:      value percent bar (0-100)
```

GUIGetValBar: Get a percent bar's value.

```
            GUIGetValBar name_bar value
name_bar:   percent bar name
value:      value percent bar (0-100)
return:     <value percent bar>\r\n
```

GUIConfigBar: Configure responsive message for a percent bar. Messages will be sent out only when enabled actions occur. All actions are enabled by default. Message patterns are described afterward.

```
            GUIConfigBar name_screen name_bar action_bar
name_screen:screen name
name_bar:   percent bar name
action_bar: 0 = MOVE (every 10ms)
            1 = STOP
            2 = Enable All (Default)
            3 = Disable All
```

Action Message:

```
            <name_bar action_bar value>\r\n
name_bar:   percent bar name
action_bar: percent bar's action (MOVE/STOP)
value:      value percent bar (0-100)
```

5.2 Other Commands and Messages

5.2.1 Echo

Echo: Enable or disable echo on serial communication.

```
            Echo state
state:      1 = Enable
            0 = Disable (default)
example:    Echo 1
```

5.2.2 Real Time Clock (RTC)

SetTime: Set time.

```
            SetTime hour min
hour:       setting hour
min:        setting minute
example:    SetTime 18 30
```

SetDate: Set date.

```
            SetDate day date month year
day:        enter day in 0-6,SUN-SAT or sun-sat
date:       enter date
month:      enter month in 1-12,JAN-DEC or jan-dec
year:       enter year in 0-9999
example:    SetDate SUN 10 JAN 2012
```

GetTime: Get current time.

```
            GetTime
return:     <hour minute second>\r\n
```

GetDate: Get current date.

```
            GetDate
return:     <day date month year>\r\n
```

5.2.3 Sound

Play: Play a single MP3 file.

```
            Play path_folder file_name
path_folder: path folder
file_name:  file name
example:    Play Audio\inter\Jazz music.mp3
```

PlayAll: Play all MP3 files in a folder.

```
            PlayAll path_folder
path_foder: path folder
example:    PlayAll Audio\inter\Jazz
```

Audio: Manage sound playback.

```
            Audio command
command:    S = Stop
            P = Pause
            C = Continue
            N = Next
            M = Mute
            L = Unmute
            J = Jump Audio (0 - 99)
            V = Volume (0 - 99)
            U = Volume Up (every 5 point)
            D = Volume Down (every 5 point)
            B = Set Bass (0 - 99)
            T = Set Treble (0 - 99)
example:    Audio V 80
            Audio P
```

GetPlayName: Get raw file name being played.

```
            GetPlayName
return:     <file name>\r\n
```

GetIsPlay: Check whether a MP3 file is being played.

```
            GetIsPlay
return:     <1>\r\n when a sound is being played. Otherwise, <0>\r\n.
```

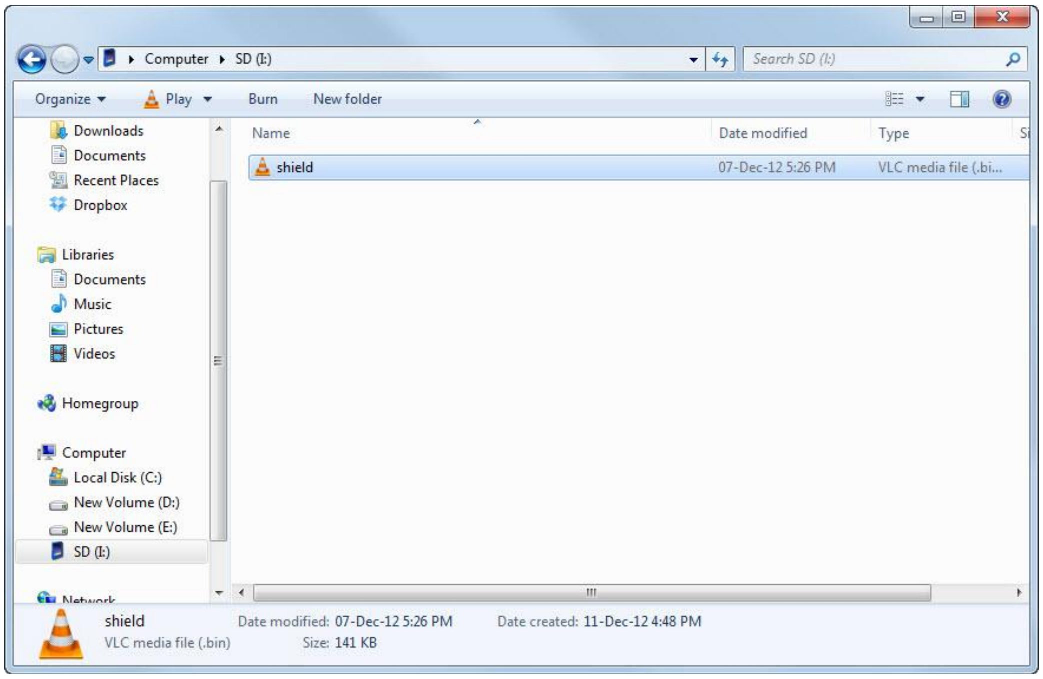
6. Firmware Upgrade

The user has to upgrade firmware of SUN7 Shield using a correlative firmware with LCD size and orientation. There are 3 available choices provided:

- 7” LCD with landscape orientation
- 7” LCD with portrait orientation
- 4.3” LCD with landscape orientation

Instructions

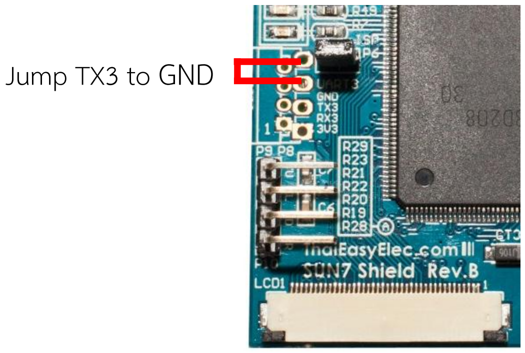
1. Store selected shield.bin in a micro-SD card and plug it into the board



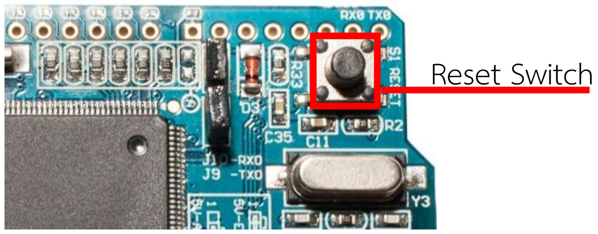
Name	Date modified	Type
shield	07-Dec-12 5:26 PM	VLC media file (.bi...

shield — Date modified: 07-Dec-12 5:26 PM — Date created: 11-Dec-12 4:48 PM
VLC media file (.bin) — Size: 141 KB

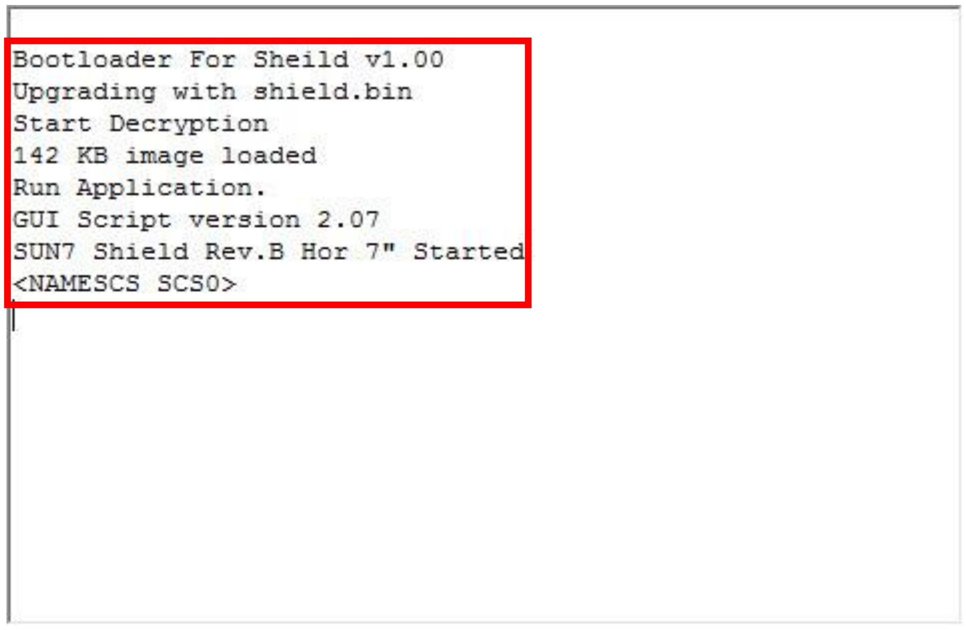
2. Jump TX3 to ground



3. Press reset button once while the board is powered

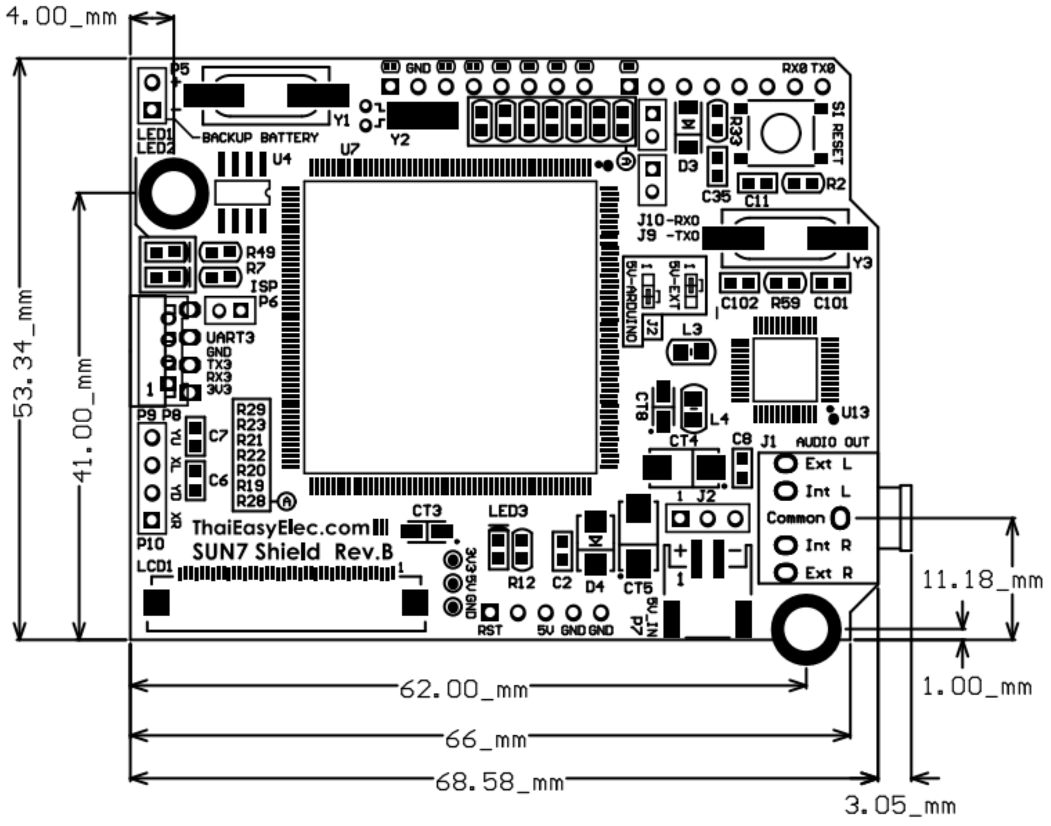


4. Finished!!!



```
Bootloader For Sheild v1.00
Upgrading with shield.bin
Start Decryption
142 KB image loaded
Run Application.
GUI Script version 2.07
SUN7 Shield Rev.B Hor 7" Started
<NAMESCS SCS0>
```

7. Dimension


4.00_mm
53.34_mm
41.00_mm
11.18_mm
1.00_mm
62.00_mm
66_mm
68.58_mm
3.05_mm
ThaiEasyElec.com
SUN7 Shield Rev.B
BACKUP BATTERY
UART3
GND
TX3
RX3
3V3
AUDIO OUT
Ext L
Int L
Common
Int R
Ext R
RESET
RST
5V
GND
5V_IN