NUS, EE2024(E)
PROGRAMMING FOR COMPUTER INTERFACES
EE2024(E) ASSIGNMENT 2 GUIDE

OVERVIEW OF THE BASE BOARD BRIEFING TAB:

- ⇨ To introduce the LPCXpresso Kit used in EE2024(E)
- ⇨ To summarise the processes involved in programming for computer interfaces
- ⇨ To support students in getting started with computer hardware interfacing

This Base Board Briefing tab builds upon content taught by:
ASSOC PROF Tham Chen Khong

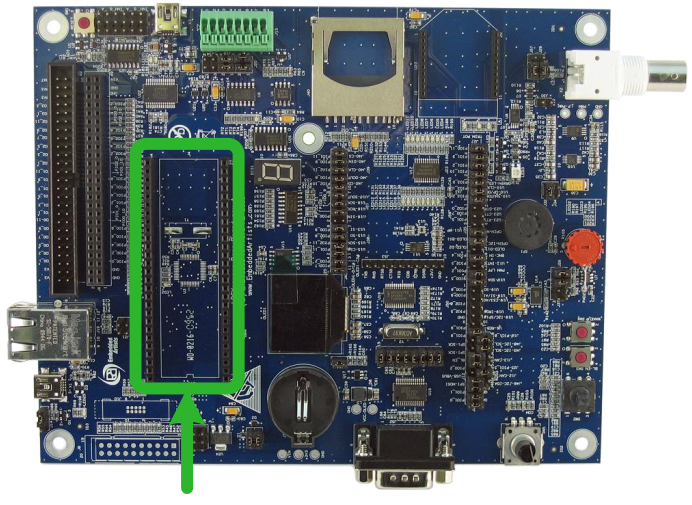
This EE2024(E) assignment 2 guide is authored by:
Christopher Moy Shin LLC

This EE2024(E) assignment 2 guide should not be used in other semesters.
The contents presented here have been customised only for the following:
NUS, EE2024(E), AY2016-2017, Semester 1

Document Version:
AY1617S1 Edition, Release B

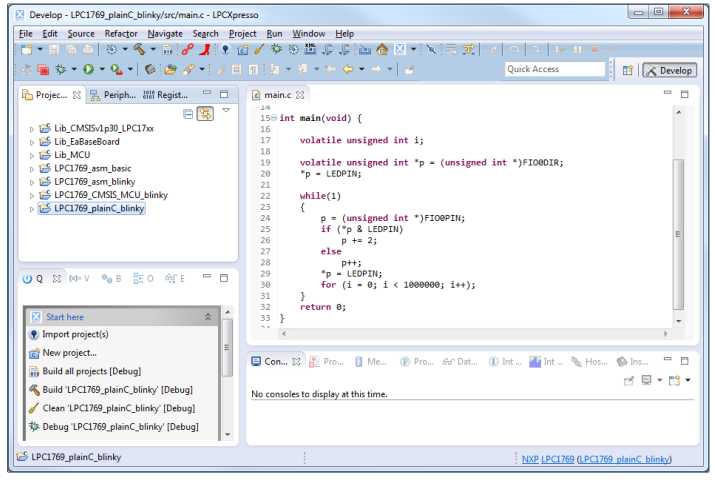
More About this Document

LPCXpresso Base Board



LPCXpresso Connector: 54 pins

LPCXpresso IDE



LPC1769 LPCXpresso Development Board

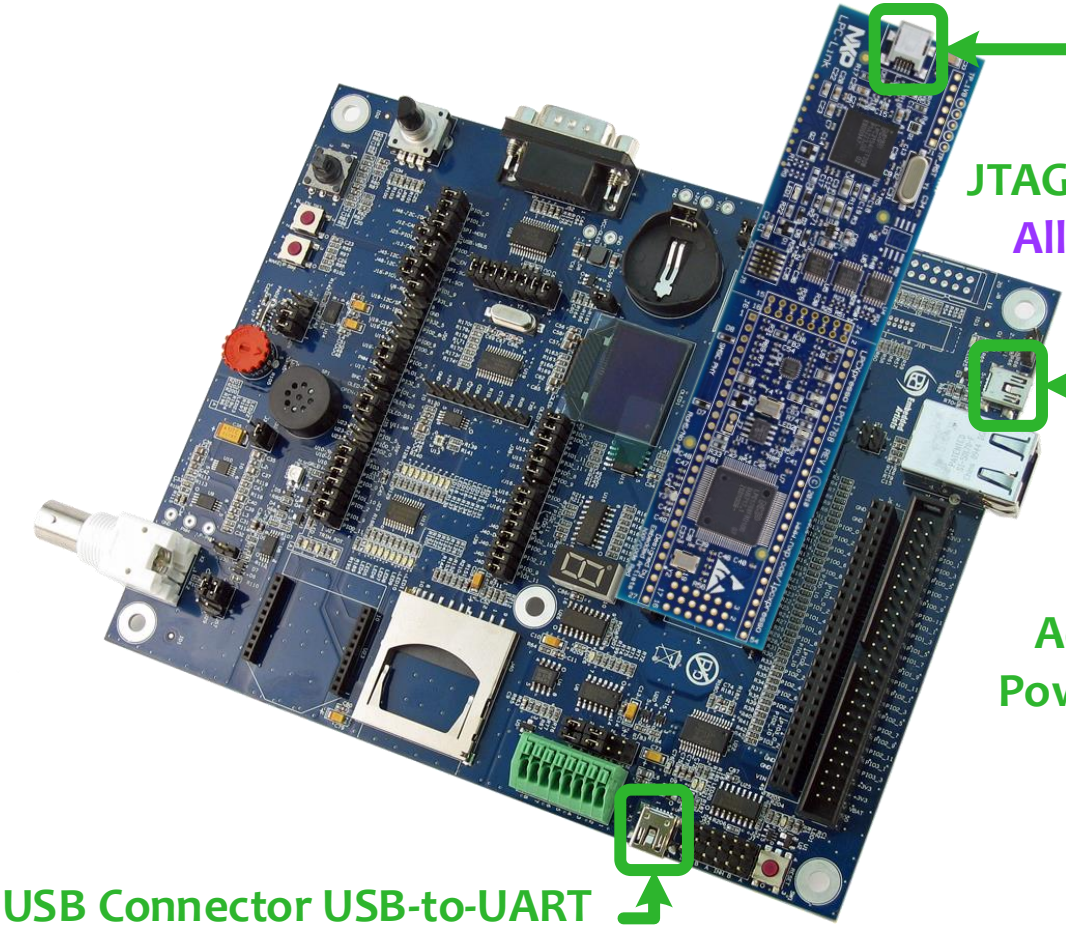
LPC1769 Microcontroller

Mini B to A USB Cable (2 Items)

Digi XBee ® RF-Module (3 Items)

Pictures are for illustration purposes only. Actual devices may vary.

Any of the three USB interfaces can be used to power the system

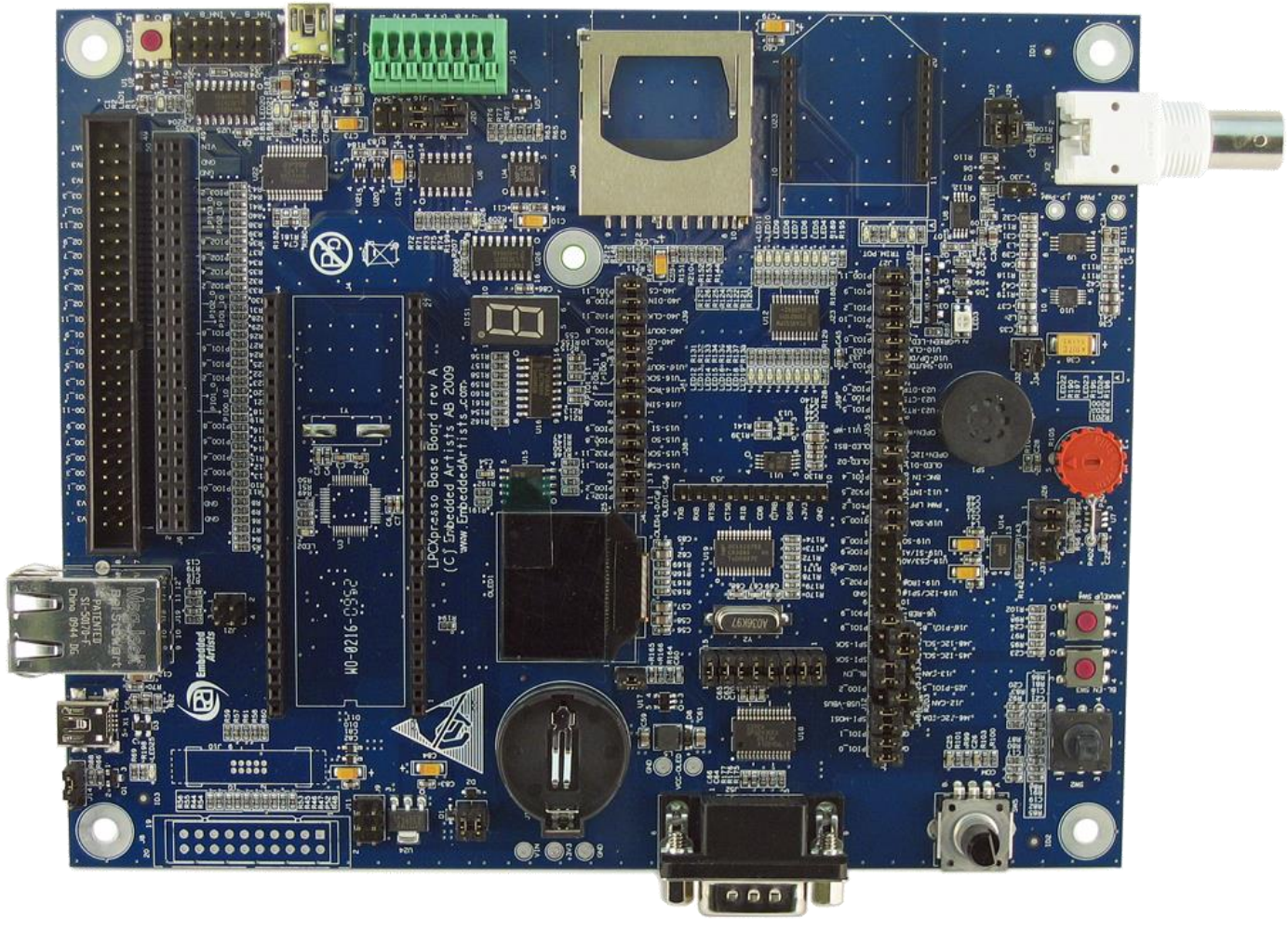


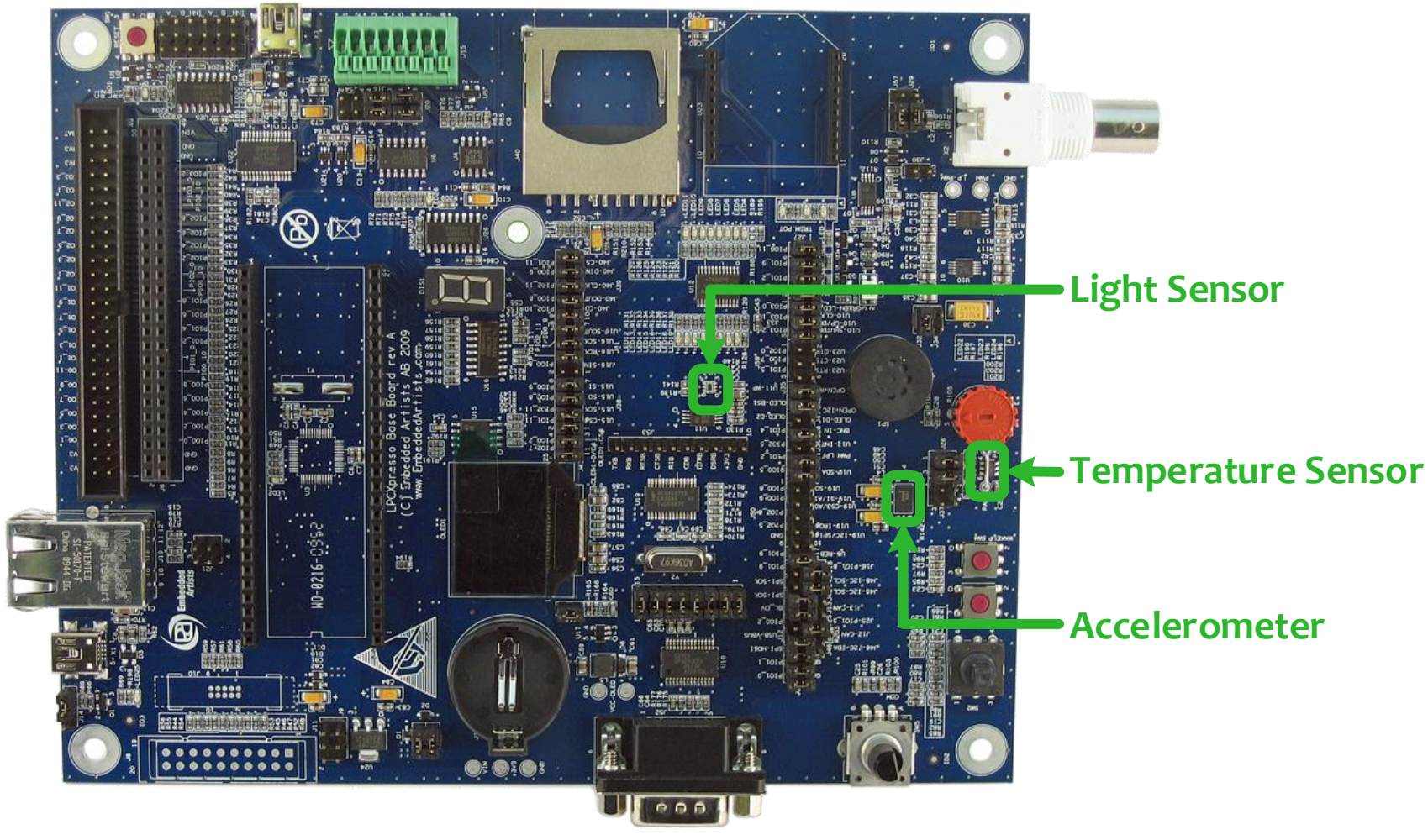
JTAG Probe (LPC-LINK)
Allows Debugging

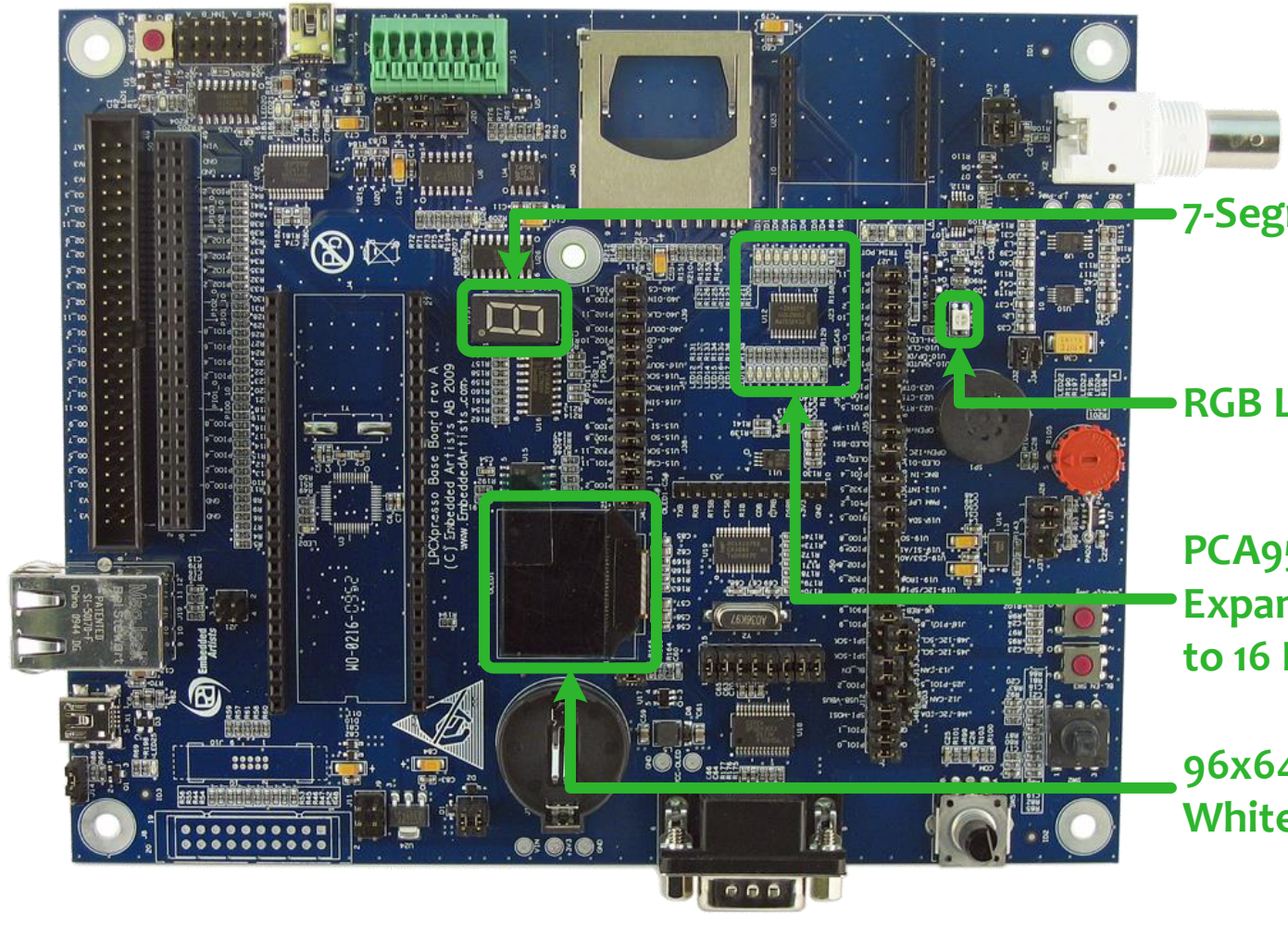
Additional
Power Source

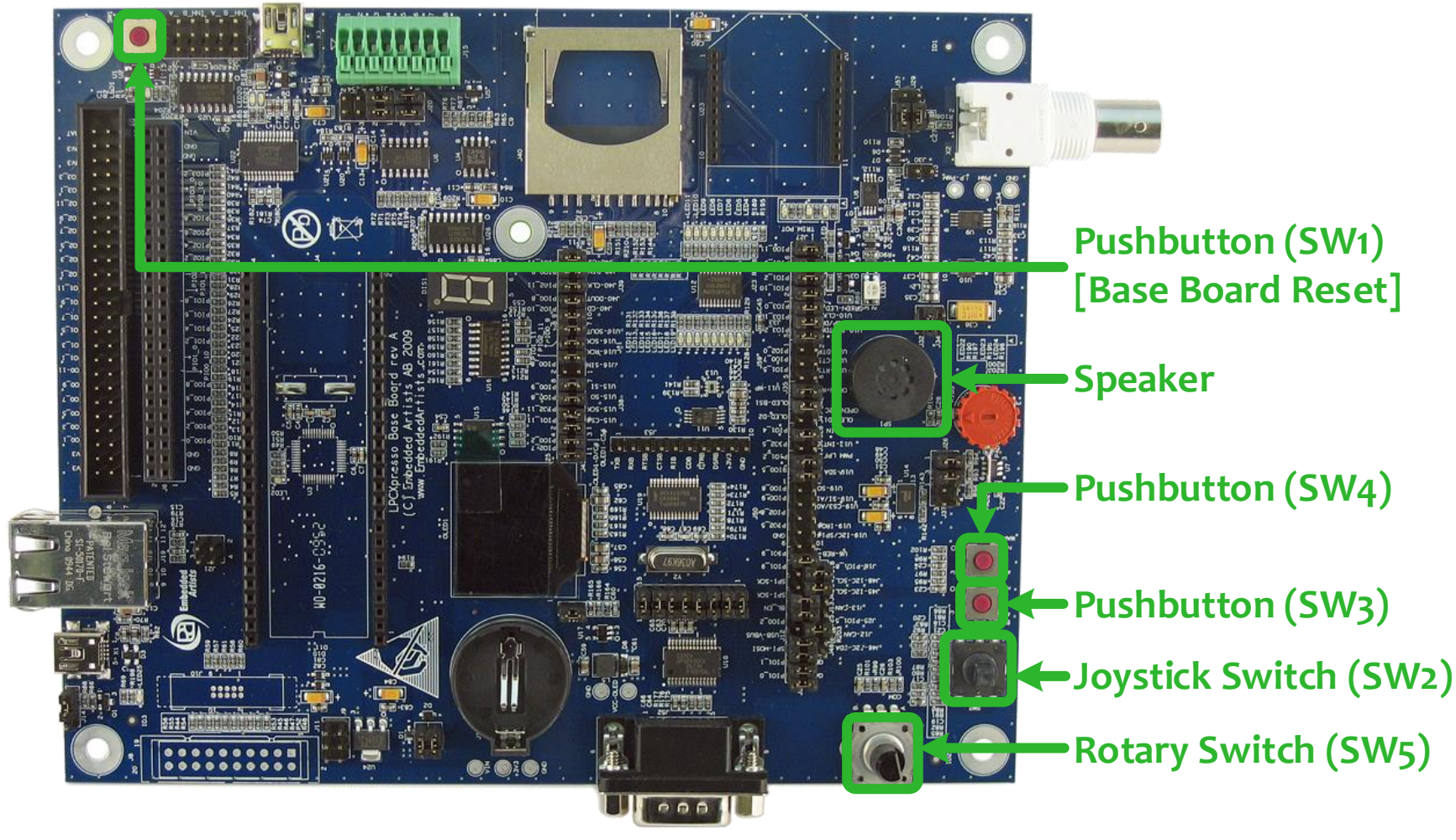
Mini B USB Connector USB-to-UART
Allows Usage of Wired UART

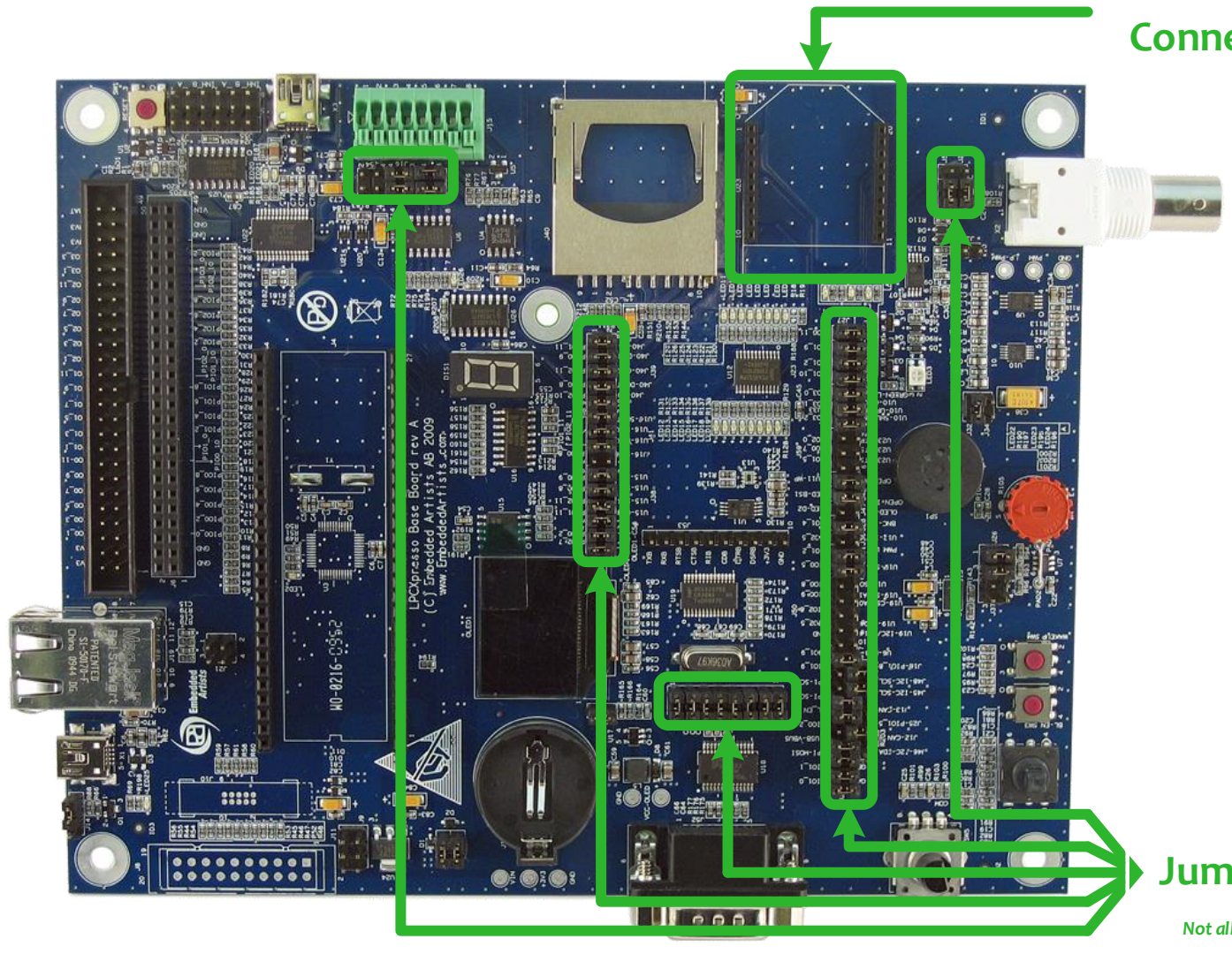
Multiple input devices, output devices and jumpers







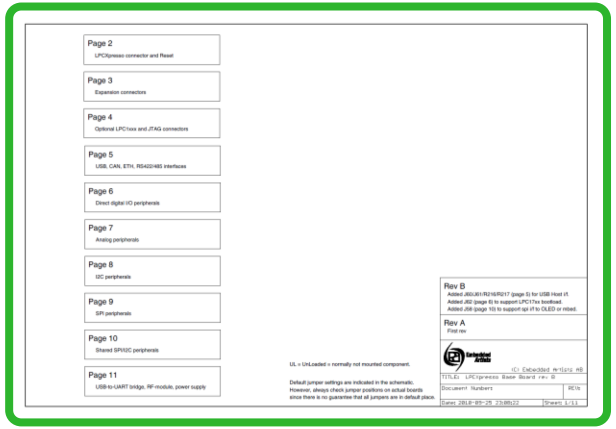




⇨ In order to use a device in a meaningful manner, there are 3 steps to be followed:

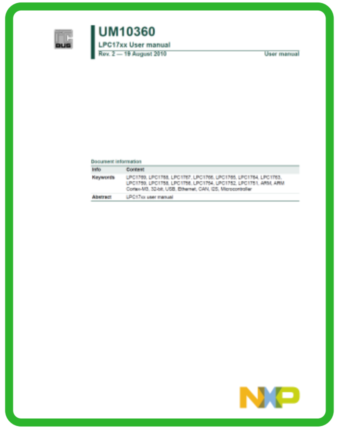
- Hardware setup (Physical changes on the Base Board):
  - Setup connections between device pins and LPC1769 pins with proper device jumper positions
- Software setup (Programming in LPCXpresso Software):
  - Program the PINSEL module to work with specific protocols
- Application programming (Programming in LPCXpresso Software):
  - Create meaningful actions based on data from/to the devices

⇨ To guide us along, 3 essential documents will be used:



Base Board Schematic Document

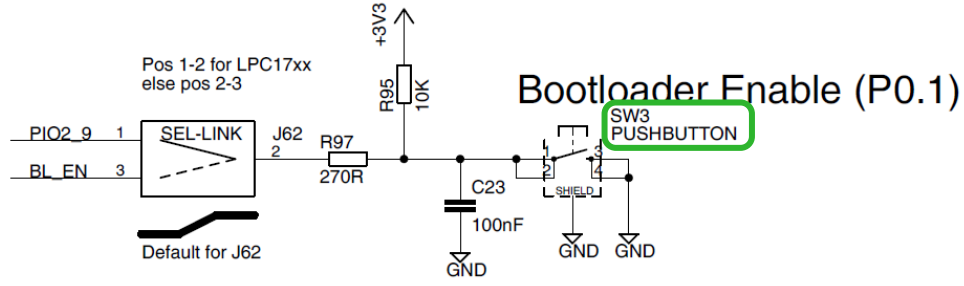
Base Board User Guide



LPC17xx User Manual

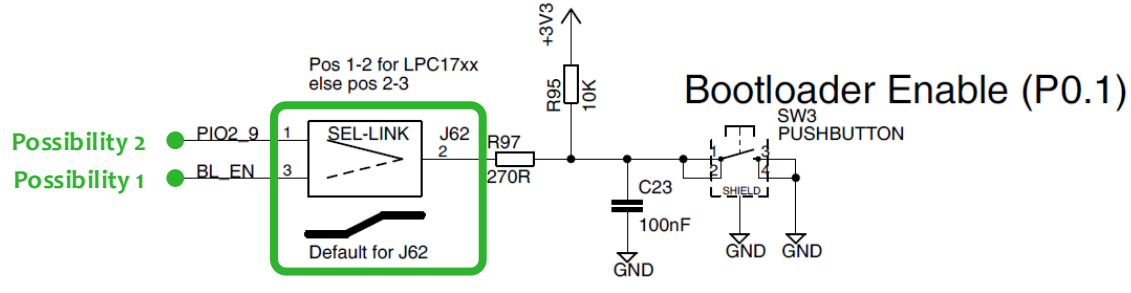
⇨ As an example, suppose the device to be used is a pushbutton, such that when it is pressed, a certain action occurs

- Searching through the Base Board Schematic Document, or the Base Board User Guide, one possibility is the pushbutton switch labelled as SW3



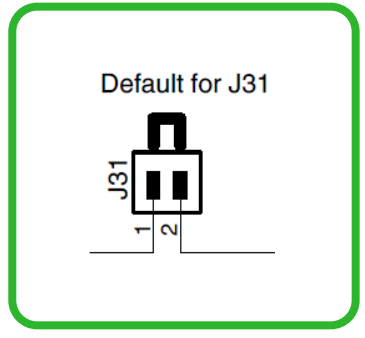
⇨ From the schematic, notice that SW3 can be connected to either one of two possibilities, but not both

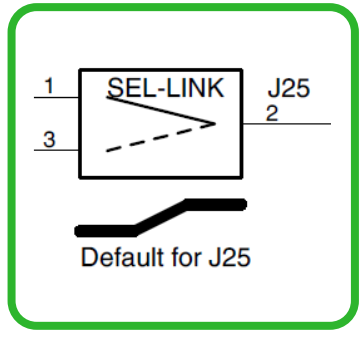
- To complete the connection to either of the possibilities, jumpers will be used

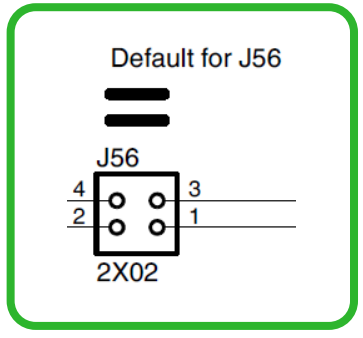


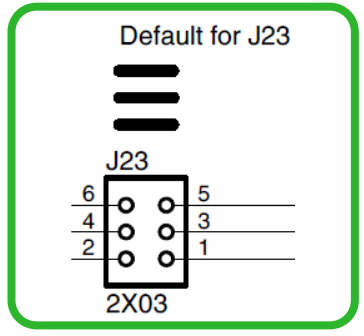
Show Device Location on BB

In the Base Board Schematic Document, device jumper connections are represented as follows:



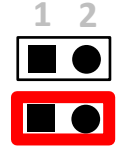




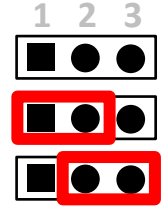


⇨ In the Base Board User Guide, device jumper connections are represented as follows:

- With 2 Base Board pins:
  - Device jumper off (Circuit is disconnected)
  - Device jumper on pin 1-2 (Pin 1 and 2 is connected)



- With 3 Base Board pins:
  - Device jumper off (Circuit is disconnected)
  - Device jumper on pin 1-2 (Pin 1 and 2 is connected)
  - Device jumper on pin 2-3 (Pin 2 and 3 is connected)



⇨ For SW3, the following information have been obtained:

- Pushbutton SW3 can be connected to either PIO2_9 or BL_EN
- Device Jumper J62 determines the connection (PIO2_9 or BL_EN)
- The Base Board User Guide indicates that J62 is a 3 pin setup:
  - If we choose pin BL_EN, then J62 should be on the middle pin and the pin labeled BL_EN
  - If we choose pin PIO2_9, then J62 should be on the middle pin and the pin labeled PIO2_9

Base Board User Guide

4.6.1 Push Button (BL) – SW3

See Figure 22 on page 27 for J13 settings (schematic page 5). A jumper must be inserted in J13, pin 1-2 in order to connect the signal from SW3 to PIO0_1 (which is the bootloader enable pin for LPC1114/LPC1343).

The LPC176x use another bootloader enable pin (P2.10, which is the PIO2_9 signal on the *LPCXpresso Base Board*). Use J62 to select where to connect the BL-button, SW3. See Figure 43 for how to set jumper.

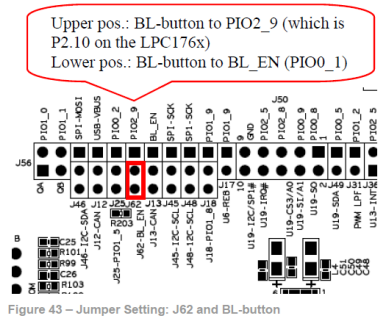
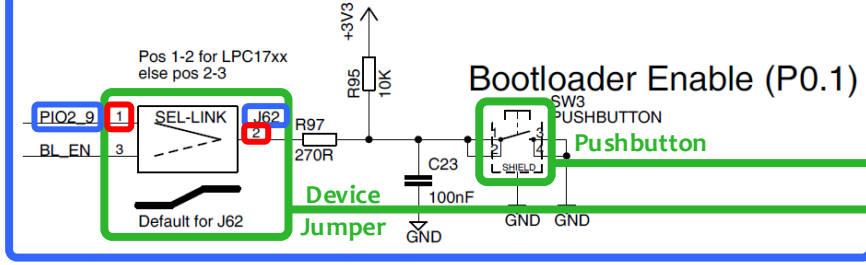
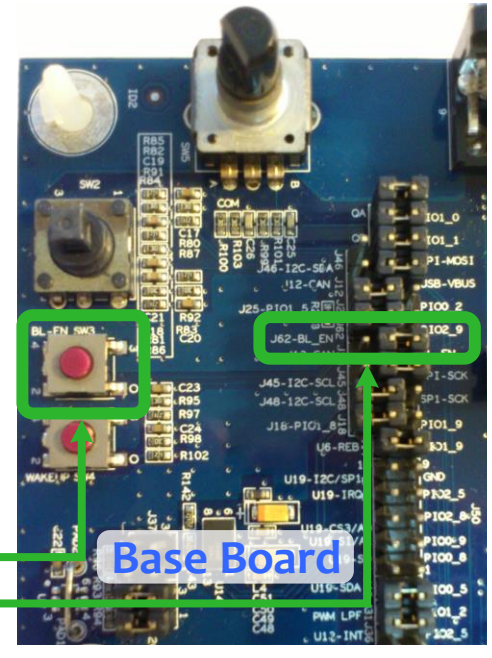


Figure 43 – Jumper Setting: J62 and BL-button

Base Board Schematic Diagram



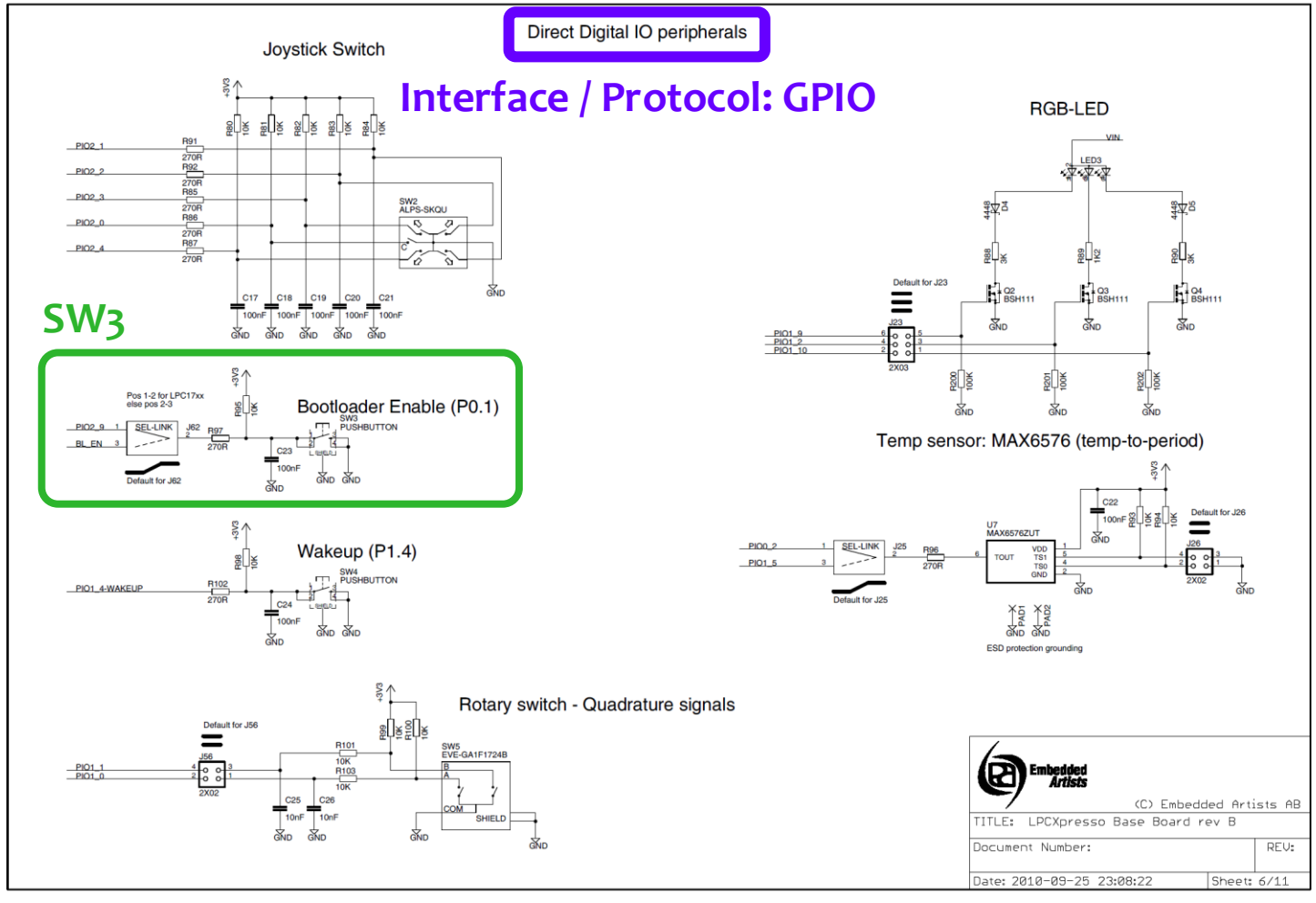


⇨ After the hardware setup, proceed with the software setup

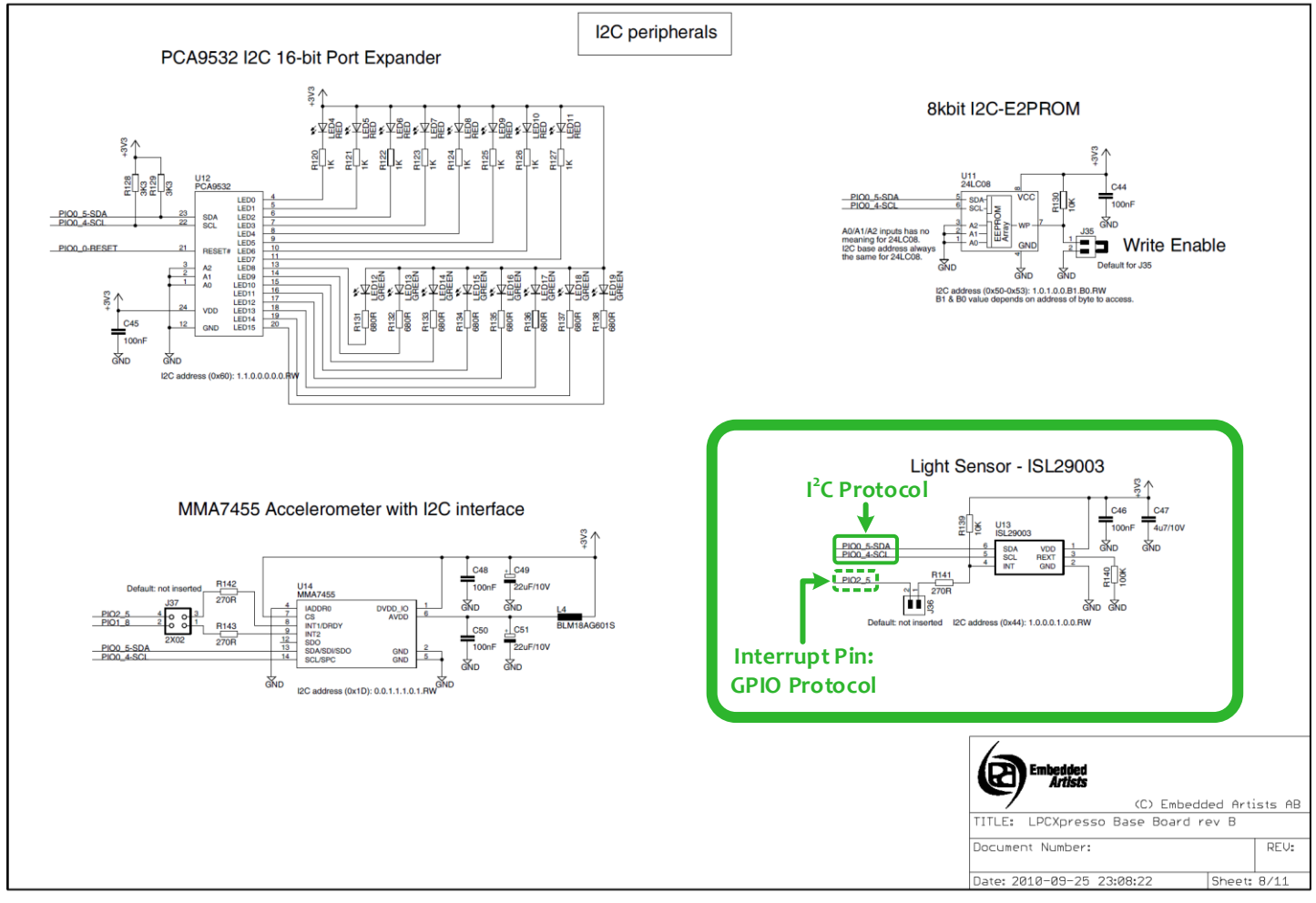
⇨ A device can communicate with LPC1769 using a specific protocol such as:
- I^2C, SPI, SSP, UART, GPIO, and so on

⇨ To be able to set the PINSEL module of the LPC1769 for specific devices, the protocol used for such devices has to be determined through their respective datasheets
- For a pre-designed system, such as our Base Board, the Base Board Schematic Document provides a quick reference
  - The title of the pages in the Base Board Schematic Document indicates the protocol used for the device
  - Detailed information are available in the devices' datasheets
- Protocols used for the different devices, such as SW3, SW4, Light Sensor, Accelerometer, RGB LED, OLED, and so on, will be indicated. Examples of such schematic page titles and respective protocols include:
  - Direct Digital IO peripherals → GPIO
  - I2C peripherals → I^2C
  - SPI peripherals → SPI
  - USB-to-UART bridge interface → UART

⇨ SW3 uses GPIO protocol, as it can be located in the following schematic page:

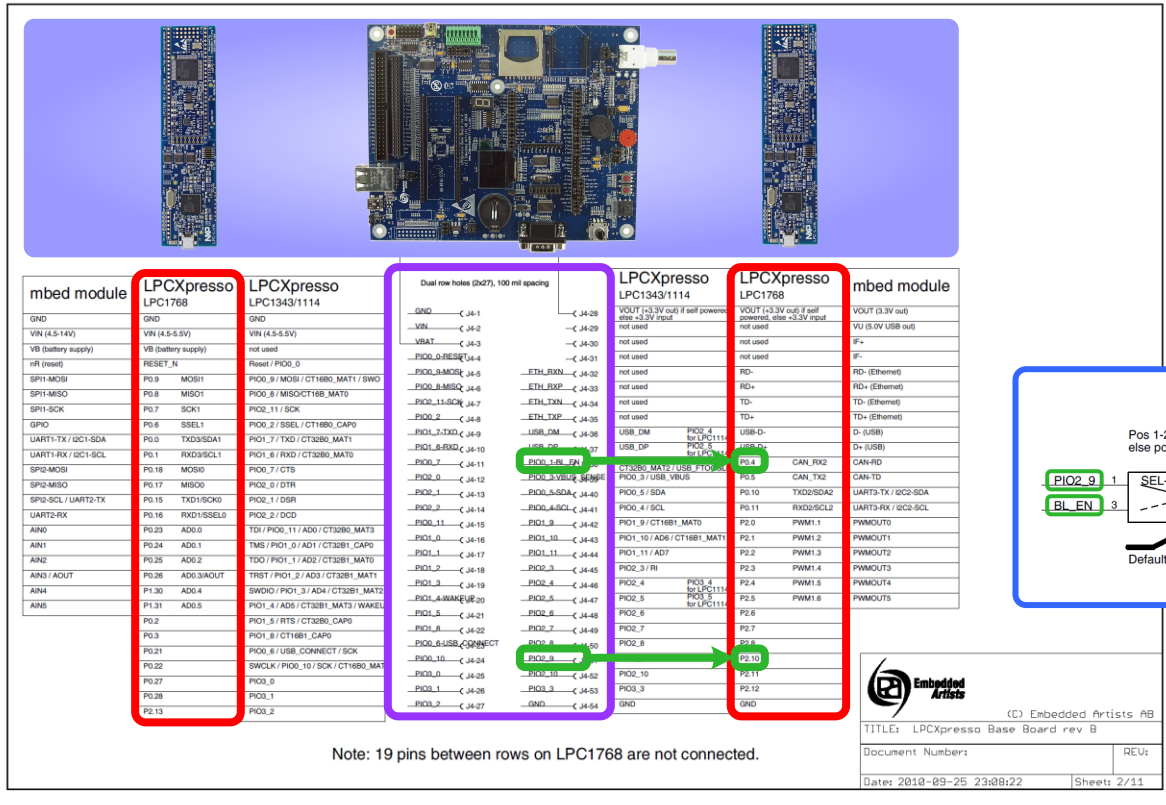


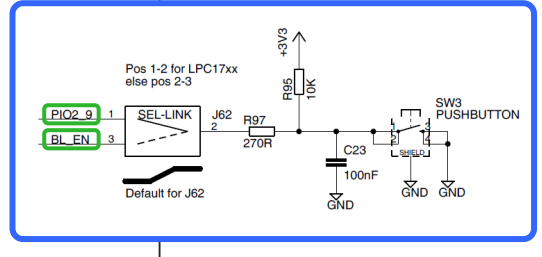
⇨ Certain devices may require multiple protocols for varying functionalities



⇨ In addition to the device protocol, the LPC1769 pins to which the device connects to are required before configuring the PINSEL module of the LPC1769

- All the pins indicated on the Base Board schematics refer to the Base Board pins. A pin mapping table, from the Base Board Schematic Document, indicates which Base Board pins connects to which LPC1769 pins





⇨ Each LPC1769 pin has up to 4 specific functions

⇨ To determine the correct function number for the required LPC1769 Pins:
- Look for the required LPC1769 pins in the Pin Function Select Register tables, in Chapter 8.5 of the LPC17xx User Manual
  - For SW3 connected to BL_EN: Look for P0.4 (Port 0, Pin 4)
  - For SW3 connected to PIO2_9: Look for P2.10 (Port 2, Pin 10)
- Take note of the function number corresponding to the device protocol obtained from the device schematic (Sheet 2)
  - For LPC1769 P0.4 to work in Direct Digital IO mode: Select Function 0
  - For LPC1769 P2.10 to work in Direct Digital IO mode: Select Function 0

Table 79. Pin function select register 0 (PINSEL0 - address 0x4002 C000) bit description

PINSEL0	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P0.0	GPIO Port 0.0	RD1	TXD3	SDA1	00
3:2	P0.1	GPIO Port 0.1	TD1	RXD3	SCL1	00
5:4	P0.2	GPIO Port 0.2	TXD0	AD0.7	Reserved	00
7:6	P0.3	GPIO Port 0.3	RXD0	AD0.6	Reserved	00
9:8	P0.4[1]	GPIO Port 0.4	I2SRX_CLK	RD2	CAP2.0	00
11:10	P0.5[1]	GPIO Port 0.5	I2SRX_WS	TD2	CAP2.1	00
13:12	P0.6	GPIO Port 0.6	I2SRX_SDA	SSEL1	MAT2.0	00
15:14	P0.7	GPIO Port 0.7	I2STX_CLK	SCK1	MAT2.1	00
17:16	P0.8	GPIO Port 0.8	I2STX_WS	MISO1	MAT2.2	00
19:18	P0.9	GPIO Port 0.9	I2STX_SDA	MOSI1	MAT2.3	00
21:20	P0.10	GPIO Port 0.10	TXD2	SDA2	MAT3.0	00
23:22	P0.11	GPIO Port 0.11	RXD2	SCL2	MAT3.1	00
29:24	-	Reserved	Reserved	Reserved	Reserved	0
31:30	P0.15	GPIO Port 0.15	TXD1	SCK0	SCK	00

Table 83. Pin function select register 4 (PINSEL4 - address 0x4002 C010) bit description

PINSEL4	Pin name	Function when 00	Function when 01	Function when 10	Function when 11	Reset value
1:0	P2.0	GPIO Port 2.0	PWM1.1	TXD1	Reserved	00
3:2	P2.1	GPIO Port 2.1	PWM1.2	RXD1	Reserved	00
5:4	P2.2	GPIO Port 2.2	PWM1.3	CTS1	Reserved [2]	00
7:6	P2.3	GPIO Port 2.3	PWM1.4	DCD1	Reserved [2]	00
9:8	P2.4	GPIO Port 2.4	PWM1.5	DSR1	Reserved [2]	00
11:10	P2.5	GPIO Port 2.5	PWM1.6	DTR1	Reserved [2]	00
13:12	P2.6	GPIO Port 2.6	PCAP1.0	RI1	Reserved [2]	00
15:14	P2.7	GPIO Port 2.7	RD2	RTS1	Reserved	00
17:16	P2.8	GPIO Port 2.8	TD2	TXD2	ENET_MDC	00
19:18	P2.9	GPIO Port 2.9	USB_CONNECT	RXD2	ENET_MDIO	00
21:20	P2.10	GPIO Port 2.10	$\overline{\text{EINT0}}$	NMI	Reserved	00
23:22	P2.11[1]	GPIO Port 2.11	$\overline{\text{EINT1}}$	Reserved	I2STX_CLK	00
25:24	P2.12[1]	GPIO Port 2.12	$\overline{\text{EINT2}}$	Reserved	I2STX_WS	00
27:26	P2.13[1]	GPIO Port 2.13	$\overline{\text{EINT3}}$	Reserved	I2STX_SDA	00
31:28	-	Reserved	Reserved	Reserved	Reserved	0

⇨ Program the LPC1769 PINSEL module for the required function of the LPC1769 pins

- In this case study, we wanted to connect SW3 to BL_EN, and this was done by by setting the correct jumper
- BL_EN pin on the base board is connected to P0.4 pin on the LPC1769
- Therefore, there is a need to program for the function selection for P0.4 of the LPC1769
  - For Port 0: `.Portnum = 0;`
  - For Pin 4: `.Pinnum = 4;`
  - For Function 0: `.Funcnum = 0;`

```
void Pinsel_SW3(void)
{
    PINSEL_CFG_Type PinCfg;
        PinCfg.Portnum = 0;
        PinCfg.Pinnum = 4;
        PinCfg.Funcnum = 0;
        PinCfg.OpenDrain = 0;
        PinCfg.Pinmode = 0;
    PINSEL_ConfigPin(&PinCfg);
}
```

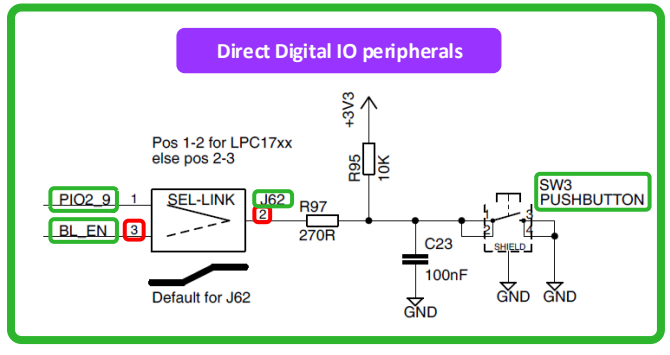
⇨ Overview of the hardware and software setup

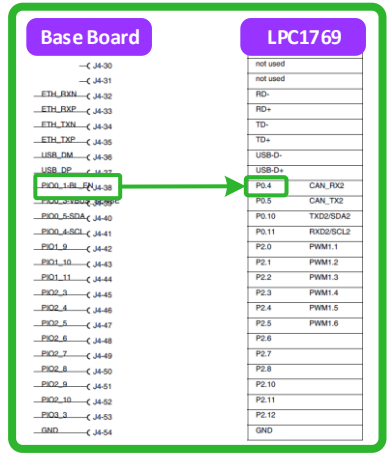
HARDWARE SETUP

SELECT A DEVICE TO USE ON BASE BOARD
Eg. Switch SW3

DETERMINE THE USEABLE BASE BOARD PINS FOR THE SELECTED DEVICE
Eg. SW3 Connects to PIO2_9 or BL_EN

FOR THE CHOSEN BASE BOARD PIN AND DEVICE, SET JUMPER(S) IF REQUIRED
Eg. SW3 to BL_EN by Setting Jumper J62, Pin 2 to Pin 3





APPLICATION PROGRAMMING

SOFTWARE SETUP

DETERMINE THE PERIPHERAL INTERFACE FROM SCHEMATICS, USER GUIDES AND DATASHEETS
Eg. Schematics Sheet 6 Indicates that SW3 Uses GPIO Interfacing

MAP BASE BOARD PINS TO THE LPC1769 PINS THAT THEY ARE CONNECTED TO
Eg. BL_EN on the Base Board Connects to P0.4 on the LPC1769

PROGRAM THE PINSEL MODULE
Eg. The LPC17xx User Manual Indicates that GPIO Interfacing for P0.4 Needs Function 0

Base Board Briefing
Introduction

⇨ The Base Board library **Lib_EaBaseBoard** supports nearly all basic functions needed to initialise and program the Base Board devices, for faster application development:

- acc.c/h ➞ Driver/Header file for the MMA7455 Accelerometer
- eeprom.c/h ➞ Driver/Header file for the 24LC08 EEPROM
- flash.c/h ➞ Driver/Header file for the SPI Flash (AT45DB081D-SU)
- font_macro.h ➞ Header file for the font generation
- font5x7.c/h ➞ Driver/Header file for the font generation
- joystick.c/h ➞ Driver/Header file for the joystick switch
- led7seg.c/h ➞ Driver/Header file for the 7 segment display
- light.c/h ➞ Driver/Header file for the ISL29003 Light-to-Digital Output Sensor
- oled.c/h ➞ Driver/Header file for OLED Display
- pca9532.c/h ➞ Driver/Header file for PCA9532 16-bit I2C LED dimmer
- rgb.c/h ➞ Driver/Header file for the RGB LED
- rotary.c/h ➞ Driver/Header file for the Rotary Switch
- temp.c/h ➞ Driver/Header file for the Temperature Sensor (MAX6576ZUT)
- uart2.c/h ➞ Driver/Header file for the I2C/SPI to UART (SC16IS752) device

⇨ For more advanced device functions, you will need to program by yourself

⇨ The programmer can make use of the required basic functions, with appropriate arguments where applicable. More detailed codes have already been written by other programmers, and are located within other library files

- Example: Programmer decides to use the light sensor, which uses I²C interface

main.c

```
// Light sensor initialization
light_enable();
```

light_enable() is used in main.c

light_enable() has already been defined by another programmer, who has analysed the light sensor's datasheet before writing the enabling codes

Lib_EaBaseBoard light.c

```
void light_enable (void)
{
    uint8_t buf[2];
    buf[0] = ADDR_CMD;
    buf[1] = CMD_ENABLE;
    I2CWrite(LIGHT_I2C_ADDR, buf, 2);
    range = RANGE_K1;
    width = WIDTH_16_VAL;
}
```

Lib_EaBaseBoard light.c (Local Function Within light.c)

```
static int I2CWrite(uint8_t addr, uint8_t* buf, uint32_t len)
{
        ...
        if (I2C_MasterTransferData(I2CDEV, &txsetup, I2C_TRANSFER_POLLING) == SUCCESS){
                return (0);
        } else {
                return (-1);
        }
}
```

Lib_MCU lpc17xx_i2c.c

```
Status I2C_MasterTransferData(LPC_I2C_TypeDef *I2Cx, I2C_M_SETUP_Type *TransferCfg, \
                                                     I2C_TRANSFER_OPT_Type Opt)
{
        ...
}
```

OVERVIEW OF THE LABORATORY GUIDE TAB:

⇨ **Milestones and summary of tasks**
- Milestones that you are expected to reach over several weeks, through self-learning and by taking note of guidelines provided during the lab sessions
- The pace of topics covered during the lab sessions will vary slightly, depending on the overall lecture, tutorial and lab progress

⇨ **List of supporting items**
- Reference documents to be used when working on the assignment
- LPC17xx libraries and demo provided on IVLE

⇨ **Device and software familarisation lab [PCI-4]**
- A demo.zip file containing a demo application is provided on IVLE. Students are required to import and run the demo application using LPCXpresso IDE. Some modifications in the demo application will be demonstrated to familarise students with the base board

EE2024(E) laboratories schedule

	SCHEDULE	ACTIVITY TYPE	DESCRIPTION	GRADE WEIGHT
[08/08/2016 – 12/08/2016]	Week 1			
[15/08/2016 – 19/08/2016]	Week 2			
[22/08/2016 – 26/08/2016]	Week 3	Laboratory [PCI-1]	Introduction to the LPCXpresso IDE environment	
[29/08/2016 – 02/09/2016]	Week 4	Laboratory [PCI-2]	Understanding and working on assignment 1	
[05/09/2016 – 09/09/2016]	Week 5			
[12/09/2016 – 16/09/2016]	Week 6	Assessment [PCI-3]	Assessment of assignment 1	20%
[17/09/2016 – 25/09/2016]	Recess week			
[26/09/2016 – 30/09/2016]	Week 7	Laboratory [PCI-4]	System design and base board familiarisation	
[03/10/2016 – 07/10/2016]	Week 8	Laboratory [PCI-5]	Interfacing to sensors and output devices using GPIO, I2C and SPI	
[10/10/2016 – 14/10/2016]	Week 9			
[17/10/2016 – 21/10/2016]	Week 10	Laboratory [PCI-6]	Interrupt handling and event detection	
[24/10/2016 – 28/10/2016]	Week 11			
[31/10/2016 – 04/11/2016]	Week 12	Laboratory [PCI-7]	Interfacing to external system using UART	
[07/11/2016 – 11/11/2016]	Week 13	Assessment [PCI-8]	Assessment of assignment 2	30%
[12/11/2016 – 18/11/2016]	Reading week			
[19/11/2016 – 03/12/2016]	Examination	Final exams	Please check on the official website for the official exam schedule	50%

System design and base board familiarisation

⇨ Import *demo.zip* project and try to run the *main.c* application without any changes

⇨ Analyse the existing *main.c* code and understand how to work with some devices:
- GPIO Interface: Pushbuttons SW3 and SW4; Joystick
- I2C Interface: PCA9532 to control the 16 LEDs; Accelerometer
- SPI/SSP Interface: OLED display, 7-segment display

⇨ Learn how to use the base board schematic document, base board user guide and LPC17xx user manual

⇨ Modify the *main.c* of demo.zip project to play a music when the pushbutton SW4 is pressed

⇨ Modify the *main.c* of demo.zip project to display different characters on the 7-segment display

⇨ Modify the *main.c* of demo.zip project to display different characters on the 7-segment display when music is playing and not playing

⇨ Understand the use for jumper connections, and attempt to use SW3 with an appropriate jumper position to play the music, instead of SW4

Interfacing to sensors and output devices using GPIO, I2C and SPI

⇨ Learn how to use Systick or Timers to keep track of time, while avoiding the use of non-resource friendly codes, such as for loops, systick delays and timer waits

⇨ Understand the protocols of GPIO, I2C and SPI/SSP interfaces, while developing some essential functions for assignment 2

⇨ Implement the functions that will:
- Turn on/off each LED in the array of 16 LEDs. A port expander allows I2C input signal to be used in order to control these 16 LEDs
- Make the red or blue light of the RGB LED (GPIO) to blink. [Note: The provided RGB library has some correctable errors which you will need to explore independently. Alternatively, you can ignore the use of the RGB library and create your own local function]
- Make the 7-segment display (SSP/SPI) to show different values periodically
- Read the accelerometer (I2C) in polling mode
- Read the light sensor (I2C) in polling mode
- Read the temperature sensor (GPIO) in polling mode
- Display required information on the 96x64 White OLED (SPI/SSP)

⇨ Understand how flowcharts may be used to graphically illustrate the overall processes of assignment 2

Interrupt handling and event detection

- ⇨ Learn how to use SW2, SW3 or SW5 as GPIO interrupt for a start. Explore the differences between GPIO interrupts and external interrupts, and consider using an external interrupt for SW3
- ⇨ Brief introduction to the light sensor interrupt
- ⇨ Understand the complete assignment 2 scenario and use all previously acquired knowledge to complete most of the requirements
- ⇨ Start to implement the communication via UART interface if most requirements have been completed

Interfacing to external system using UART

⇨ Program to work with UART (through the USB-to-UART bridge) and send a string from the LPC1769 to a communication terminal on a computer

⇨ Consider making use of wireless UART communication instead of wired UART communication

⇨ Complete all requirements of assignment 2

⇨ Add extra features and extensions only if the basics requirements of the assignment have been met.

Assignment 2 assessment week

- ⇨ Present and explain the flowcharts for assignment 2 to the assessing G.A.
- ⇨ Fully understand the codes and answer the questions asked by the G.A. during the assessment
- ⇨ Demonstrate the assignment 2 system that you have developed, without any extra features and extensions
- ⇨ Demonstrate the assignment 2 system with extra features and extensions that you may have included
- ⇨ *printf* statements must not be used. All display of information must be done through UART, or other output devices on the baseboard
- ⇨ Submit codes during the assessment, and the report through IVLE
- ⇨ Additional details are given in the last section (assessment section) of the lab manual
- ⇨ Further details will be sent closer to the assessment week

Reference documents

⇨ **Base board user guide:**
- LPCXpresso Base Board Rev B User's Guide: LPCXpresso_BaseBoard_rev_B_Users_Guide_Rev_PA18.pdf

⇨ **Base board schematic diagrams:**
- LPCXpresso Base Board Rev B schematics: LPCXpresso_Base_Board_revB.pdf

⇨ **LPCXpresso IDE installation guide and user manual:**
- LPCXpresso_User_Guide.pdf

⇨ **Base board device datasheets:**
- MAX6576ZUT - Temperature Sensor
- PCA9532 - 16-bit Port Expander (LED dimmer)
- 24LC08 - 8Kbit I2C Serial EEPROM
- MMA7455 - Three Axis Accelerometer
- ISL29003 - Light Sensor
- AT45DB161D - 16Mbit SPI-Flash
- SC16IS752 - Dual UART with I2C/SPi interface
- UG-9664HSWAG01 - OLED Display
- UG-9664HSWAG01 Guide - OLED Display User Guide
- SSD1305 - 132x64 Display Controller

LPC17xx libraries and demo

⇨ **Lib_CMSISv1p30_LPC17xx**

CMSIS library:

- Peripheral Register and Interrupt Definitions: a consistent interface for device registers and interrupts
- Core Peripheral Functions: access functions for specific processor features and core peripherals

⇨ **Lib_EaBaseBoard**

Embedded artists base board library:

- Base board device driver functions, which are to enable device, initialize device, read device, sent data to device, etc.

⇨ **Lib_MCU**

LPC17xx Microcontroller peripheral library:

- LPC17xx peripherals drivers including interfaces, clocks, NVIC, GPIO, etc.

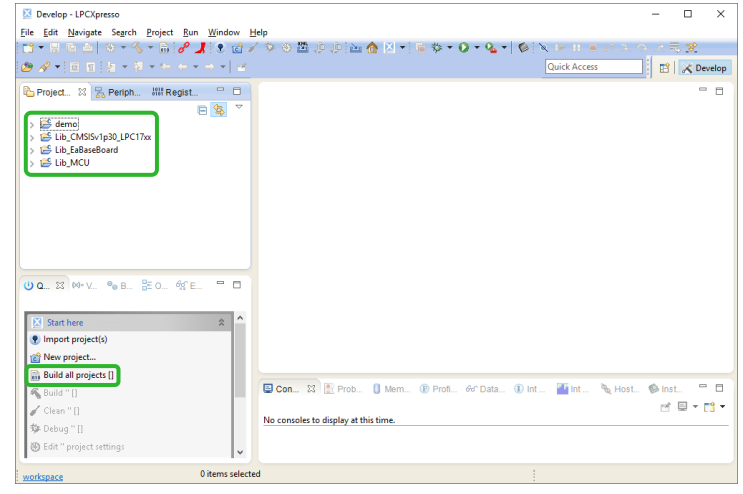
⇨ **demo (Obtained from the archive *demo.zip*)**

The C program demonstration uses the above libraries to operate the base board peripherals. Please start with the demo in order to gain experience with the LPC17xx microcontroller and its libraries.

Device and software familarisation lab

In the introduction lab (Lab 1: PCI-1), the importing of a project was demonstrated. Reference may be made to section 3.7, page 16 from the *LPCXpresso_BaseBoard_rev_B_Users_Guide_Rev_PA18.pdf*, or a graduate assistant can be approached for difficulties. For assignment 2, the following files are required to be imported:

- demo
- Lib_CMSISv1p30_LPC17xx
- Lib_EaBaseBoard
- Lib_MCU



For some system configurations, an exit error might occur during debugging. To avoid this situation, start with a clean workspace by deleting the existing workspace **folder** from your drive. Furthermore, import the libraries **before** importing the demo file.

Select "**Build all projects**" before proceeding

Device and software familarisation lab

Ensure that all baseboard jumpers are at their default positions before plugging in your assembled kit (LPC1769 LPCXpresso development board + base board).

Reference is to made from the laminated sheet provided together with the LPCXpresso kit, and which is labelled as **Figure 10 – LPCXpresso Base Board default Jumper Settings**.

Alternatively, for the default jumper settings, refer to page 15 of *LPCXpresso_BaseBoard_rev_B_Users_Guide_Rev_PA18.pdf*

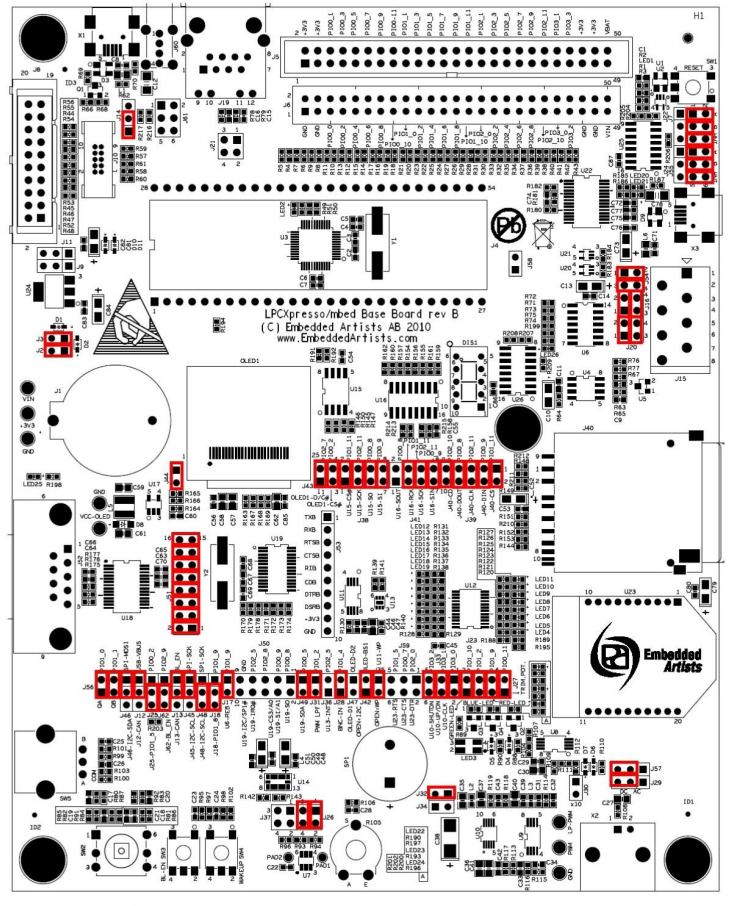


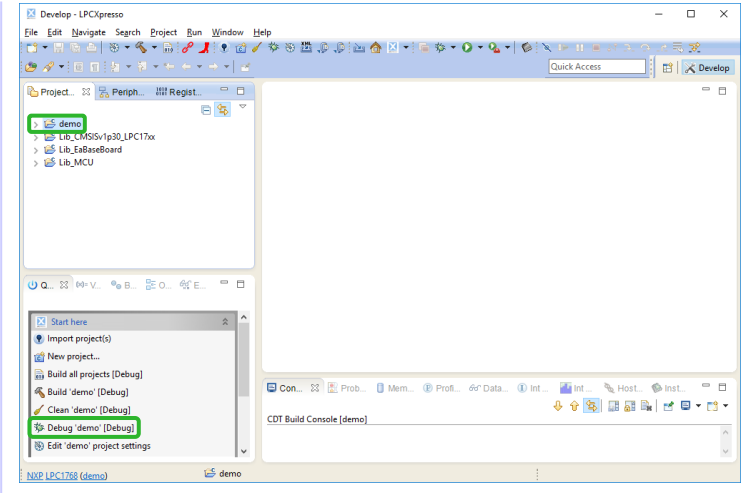
Figure 10 – LPCXpresso Base Board default Jumper Settings

Device and software familarisation lab

Plug in your LPC1769 LPCXpresso development board and then:

- Select **"demo"**
- Followed by **"Debug 'demo' [Debug]"**

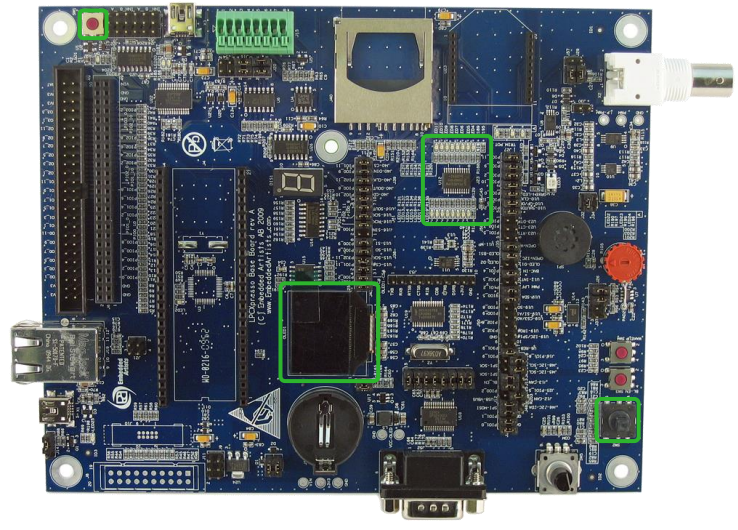
Connect at least two USB power sources, in case the base board and LPC1769 LPCXpresso development board are unable to get the required amount of power with just one USB power source



Device and software familarisation lab

The demo application uses several peripherals on the base board, such as accelerometer, joystick, 16 LEDs and OLED. It can:

- Light up the 16 LEDs connected to the PCA9532 by using the base board inclination data in a certain axis
  - The reset button SW1 allows the base board to reinitialise the default tilt value. Deviations from this default tilt value causes the LED bar to move
  - Three out of the sixteen LEDs will always be lit as a bar
  - By tilting the board, the LED bar light will move
  - The tilt direction and magnitude determines the direction and speed of the moving LED bar
- Draw on the OLED display with the joystick
  - The joystick can be used to draw on the OLED display. This is like moving a pen without lifting it out of the paper
  - By pressing the centre key on the joystick the display is cleared

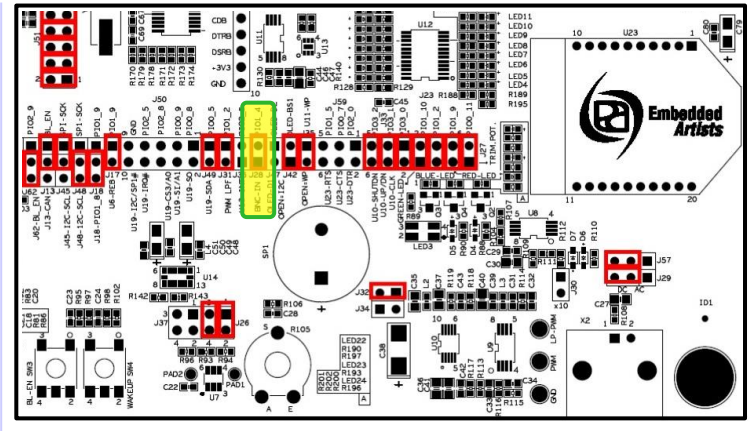


Device and software familarisation lab

Modify the demo application in order to play sounds by pressing the SW4 button. The following code in the *main.c* file, at the suggested *original* line number, can be used for this task:

```
/* Line 261 */      PINSEL_CFG_Type PinCfg;
/* Line 262 */
/* Line 263 */          PinCfg.Funcnum = 0;
/* Line 264 */          PinCfg.OpenDrain = 0;
/* Line 265 */          PinCfg.Pinmode = 0;
/* Line 266 */          PinCfg.Portnum = 1;
/* Line 267 */          PinCfg.Pinnum = 31;
/* Line 268 */
/* Line 269 */      PINSEL_ConfigPin(&PinCfg);
/* Line 270 */
/* Line 271 */      GPIO_SetDir(1, 1<<31, 0);

/* Line 371 */      btn1 = (GPIO_ReadValue(1) >> 31) & 0x01;
```



Note: In order to use SW4, remove the device jumper J28 from the base board, as indicated in section 4.6.2, page 41, in *LPCXpresso_BaseBoard_rev_B_Users_Guide_Rev_PA18.pdf*

Device and software familarisation lab

Modify the demo application in order to display characters on the 7-segment display:

- Show the number 5 on the 7-segment display if the sound is not being played
- Show the alphabet P on the 7-segment display if the sound is being played

The needed headers, functions and instructions are listed below:

- This should be included at the beginning of the *main.c* file, but after the ***#include "lpc17xx_gpio.h"***. Failure to do so will lead to build build/debug errors:

```
#include "led7seg.h"
```

- Include this initialisation function at the beginning of the ***int main(void)*** function, where other initialisation functions are also located (For example: After ***oled_init()***):

```
led7seg_init();
```

- Use the following function to display the number 5 on the 7-segment display:

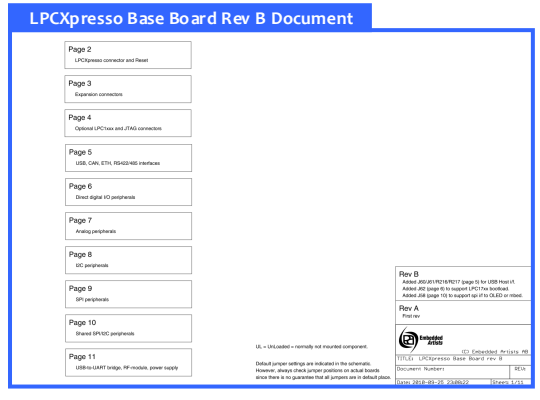
```
led7seg_setChar('5', FALSE);
```

- Use the following function to display the alphabet P on the 7-segment display:

```
led7seg_setChar('P', FALSE);
```

OVERVIEW OF THE SCHEMATICS TAB:

⇨ **References to schematics based on:**
- The interface type
  - Obtained from the LPCXpresso Base Board Rev B document
  - For character searching, please use the original document
- Commonly used devices



⇨ **Further support provided by the LPCXpresso Base Board Rev B User's Guide document**
- The LPCXpresso Base Board User Guide PDF file should be used to properly set up the devices that will be used

UL = UnLoaded = normally not mounted component.

Default jumper settings are indicated in the schematic.
However, always check jumper positions on actual boards
since there is no guarantee that all jumpers are in default place.

Rev B
Added J60/J61/R216/R217 (page 5) for USB Host i/f.
Added J62 (page 6) to support LPC17xx bootload.
Added J58 (page 10) to support spi i/f to OLED or mbed.

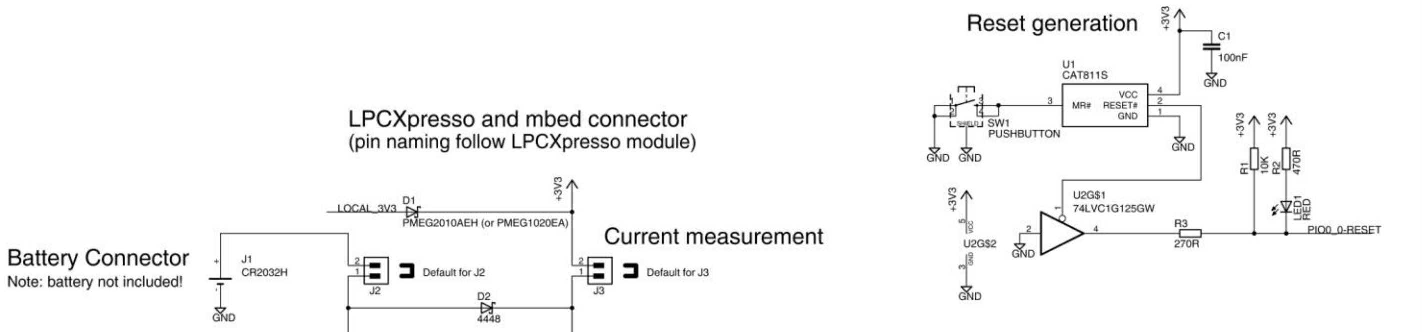
Rev A
First rev

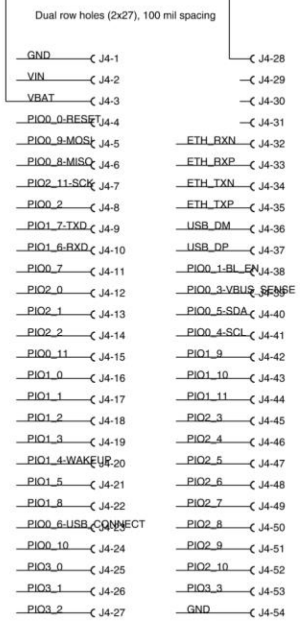
Embedded Artists

TITLE: LPCXpresso Base Board rev B

Document Number: | REV:

Date: 2010-09-25 23:08:22 | Sheet: 1/11



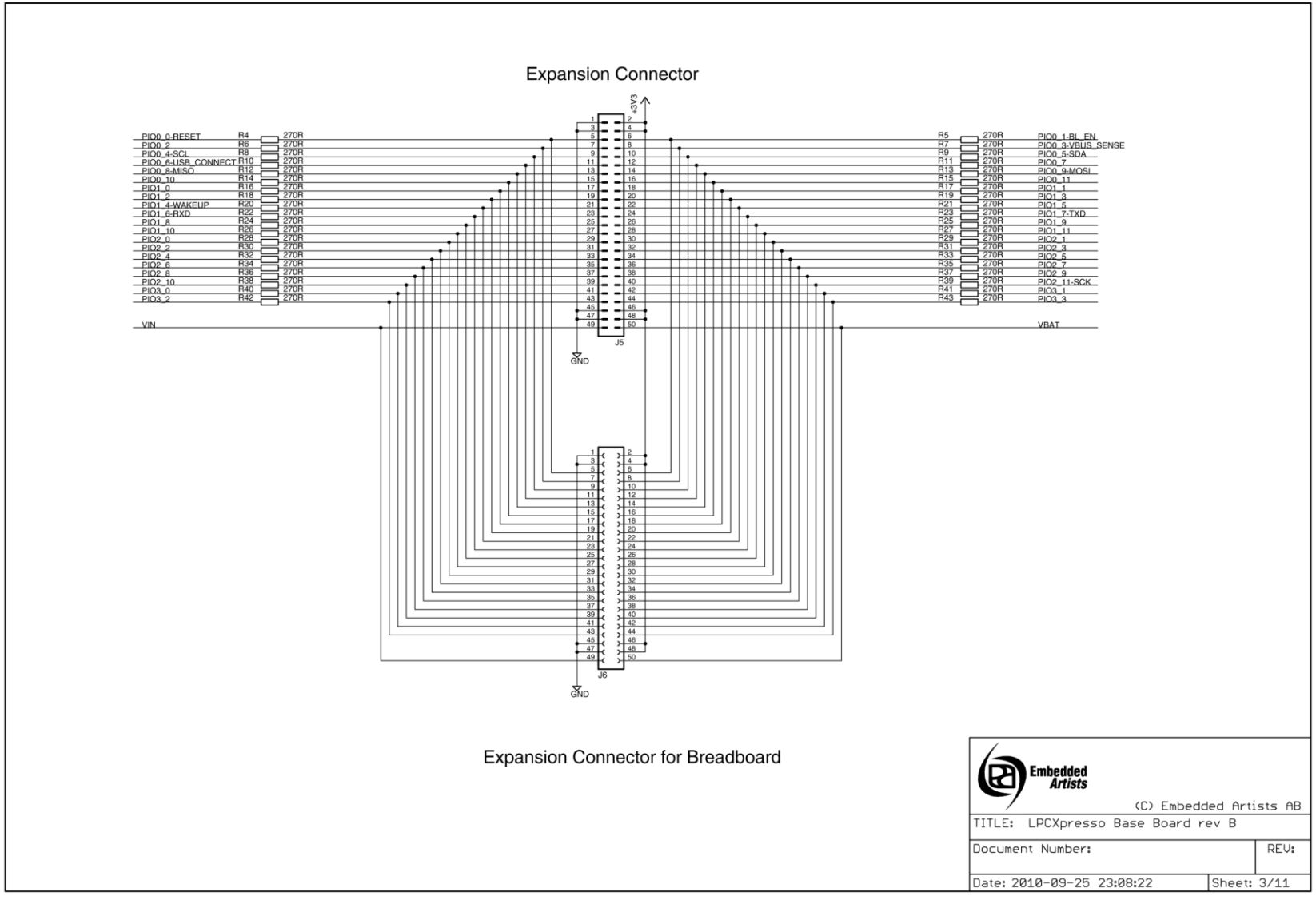


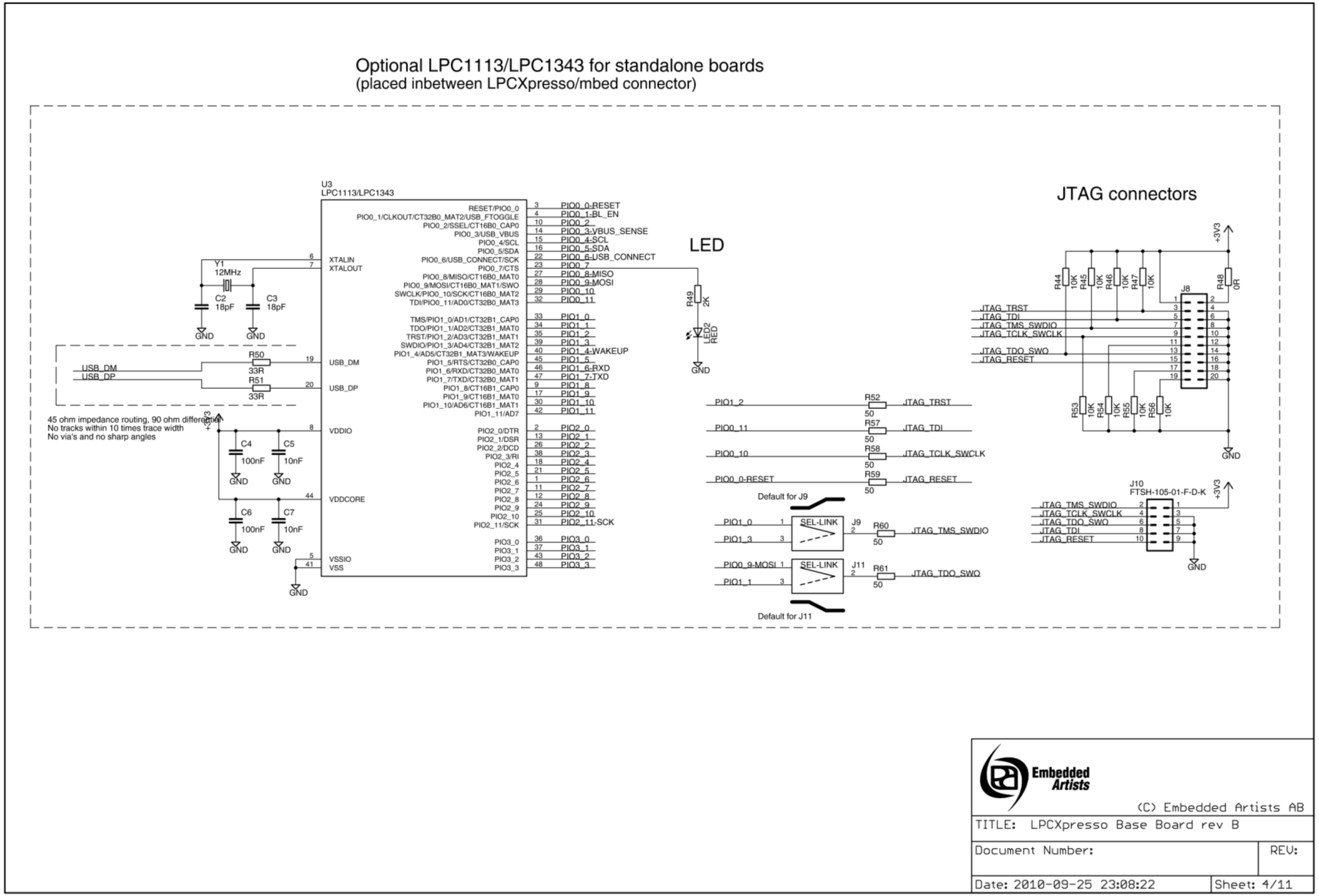
mbed module	LPCXpresso LPC1768		LPCXpresso LPC1343/1114
GND	GND		GND
VIN (4.5-14V)	VIN (4.5-5.5V)		VIN (4.5-5.5V)
VB (battery supply)	VB (battery supply)		not used
nR (reset)	RESET_N		Reset / PIO0_0
SPI1-MOSI	P0.9	MOSI1	PIO0_9 / MOSI / CT16B0_MAT1 / SWO
SPI1-MISO	P0.8	MISO1	PIO0_8 / MISO/CT16B_MAT0
SPI1-SCK	P0.7	SCK1	PIO2_11 / SCK
GPIO	P0.6	SSEL1	PIO0_2 / SSEL / CT16B0_CAP0
UART1-TX / I2C1-SDA	P0.0	TXD3/SDA1	PIO1_7 / TXD / CT32B0_MAT1
UART1-RX / I2C1-SCL	P0.1	RXD3/SCL1	PIO1_6 / RXD / CT32B0_MAT0
SPI2-MOSI	P0.18	MOSI0	PIO0_7 / CTS
SPI2-MISO	P0.17	MISO0	PIO2_0 / DTR
SPI2-SCL / UART2-TX	P0.15	TXD1/SCK0	PIO2_1 / DSR
UART2-RX	P0.16	RXD1/SSEL0	PIO2_2 / DCD
AIN0	P0.23	AD0.0	TDI / PIO0_11 / AD0 / CT32B0_MAT3
AIN1	P0.24	AD0.1	TMS / PIO1_0 / AD1 / CT32B1_CAP0
AIN2	P0.25	AD0.2	TDO / PIO1_1 / AD2 / CT32B1_MAT0
AIN3 / AOUT	P0.26	AD0.3/AOUT	TRST / PIO1_2 / AD3 / CT32B1_MAT1
AIN4	P1.30	AD0.4	SWDIO / PIO1_3 / AD4 / CT32B1_MAT2
AIN5	P1.31	AD0.5	PIO1_4 / AD5 / CT32B1_MAT3 / WAKEUP
	P0.2		PIO1_5 / RTS / CT32B0_CAP0
	P0.3		PIO1_8 / CT16B1_CAP0
	P0.21		PIO0_6 / USB_CONNECT / SCK
	P0.22		SWCLK / PIO0_10 / SCK / CT16B0_MAT2
	P0.27		PIO3_0
	P0.28		PIO3_1
	P2.13		PIO3_2

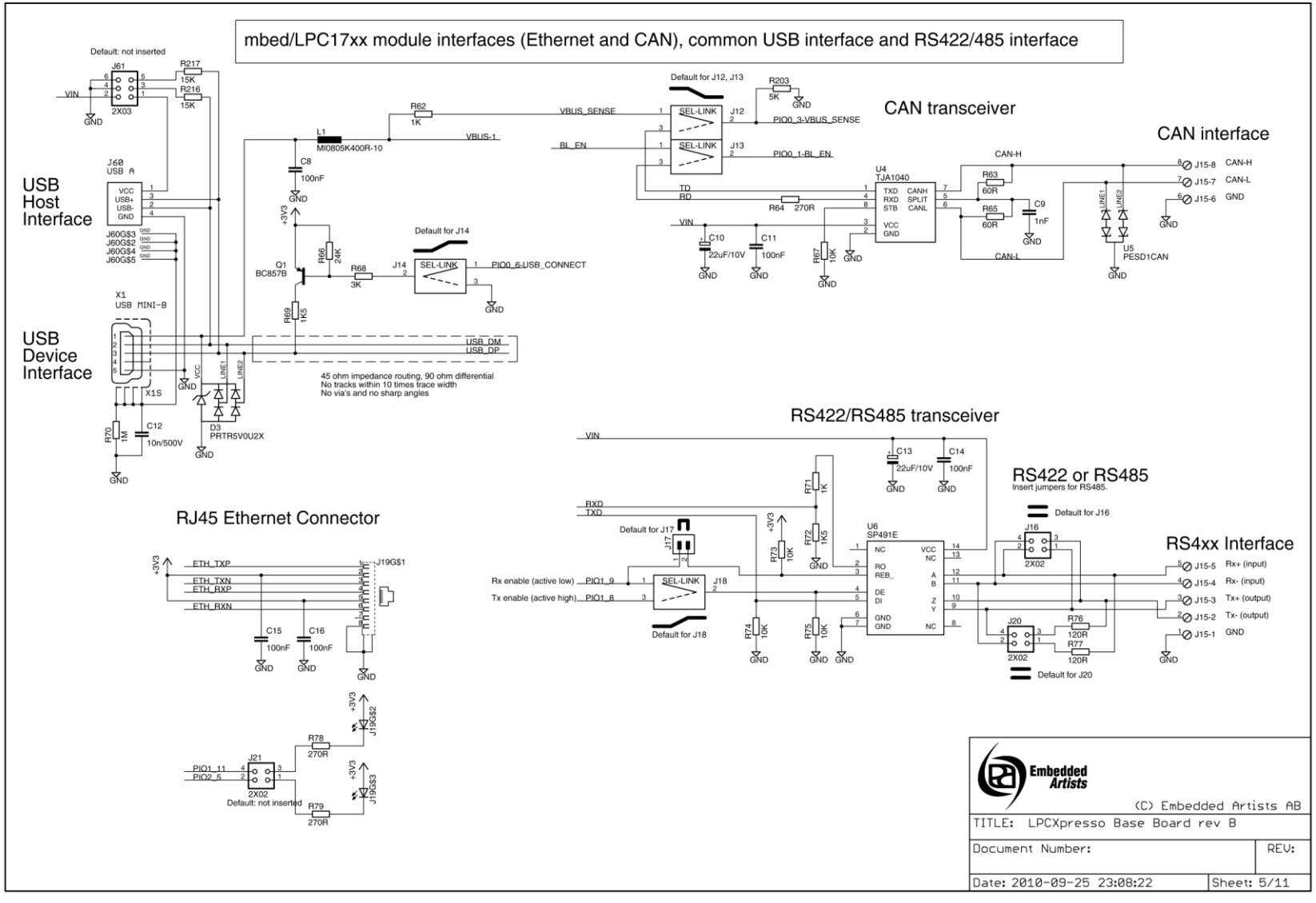
LPCXpresso LPC1343/1114		LPCXpresso LPC1768		mbed module
VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (+3.3V out) if self powered, else +3.3V input		VOUT (3.3V out)
not used		not used		VU (5.0V USB out)
not used		not used		IF+
not used		not used		IF-
not used		RD-		RD- (Ethernet)
not used		RD+		RD+ (Ethernet)
not used		TD-		TD- (Ethernet)
not used		TD+		TD+ (Ethernet)
USB_DM	PIO2_4 for LPC1114	USB-D-		D- (USB)
USB_DP	PIO2_5 for LPC1114	USB-D+		D+ (USB)
PIO0_1 / CLKOUT / CT32B0_MAT2 / USB_FTOGGLE		P0.4	CAN_RX2	CAN-RD
PIO0_3 / USB_VBUS		P0.5	CAN_TX2	CAN-TD
PIO0_5 / SDA		P0.10	TXD2/SDA2	UART3-TX / I2C2-SDA
PIO0_4 / SCL		P0.11	RXD2/SCL2	UART3-RX / I2C2-SCL
PIO1_9 / CT16B1_MAT0		P2.0	PWM1.1	PWMOUT0
PIO1_10 / AD6 / CT16B1_MAT1		P2.1	PWM1.2	PWMOUT1
PIO1_11 / AD7		P2.2	PWM1.3	PWMOUT2
PIO2_3 / RI		P2.3	PWM1.4	PWMOUT3
PIO2_4	PIO3_4 for LPC1114	P2.4	PWM1.5	PWMOUT4
PIO2_5	PIO3_5 for LPC1114	P2.5	PWM1.6	PWMOUT5
PIO2_6		P2.6		
PIO2_7		P2.7		
PIO2_8		P2.8		
PIO2_9		P2.10		
PIO2_10		P2.11		
PIO3_3		P2.12		
GND		GND		

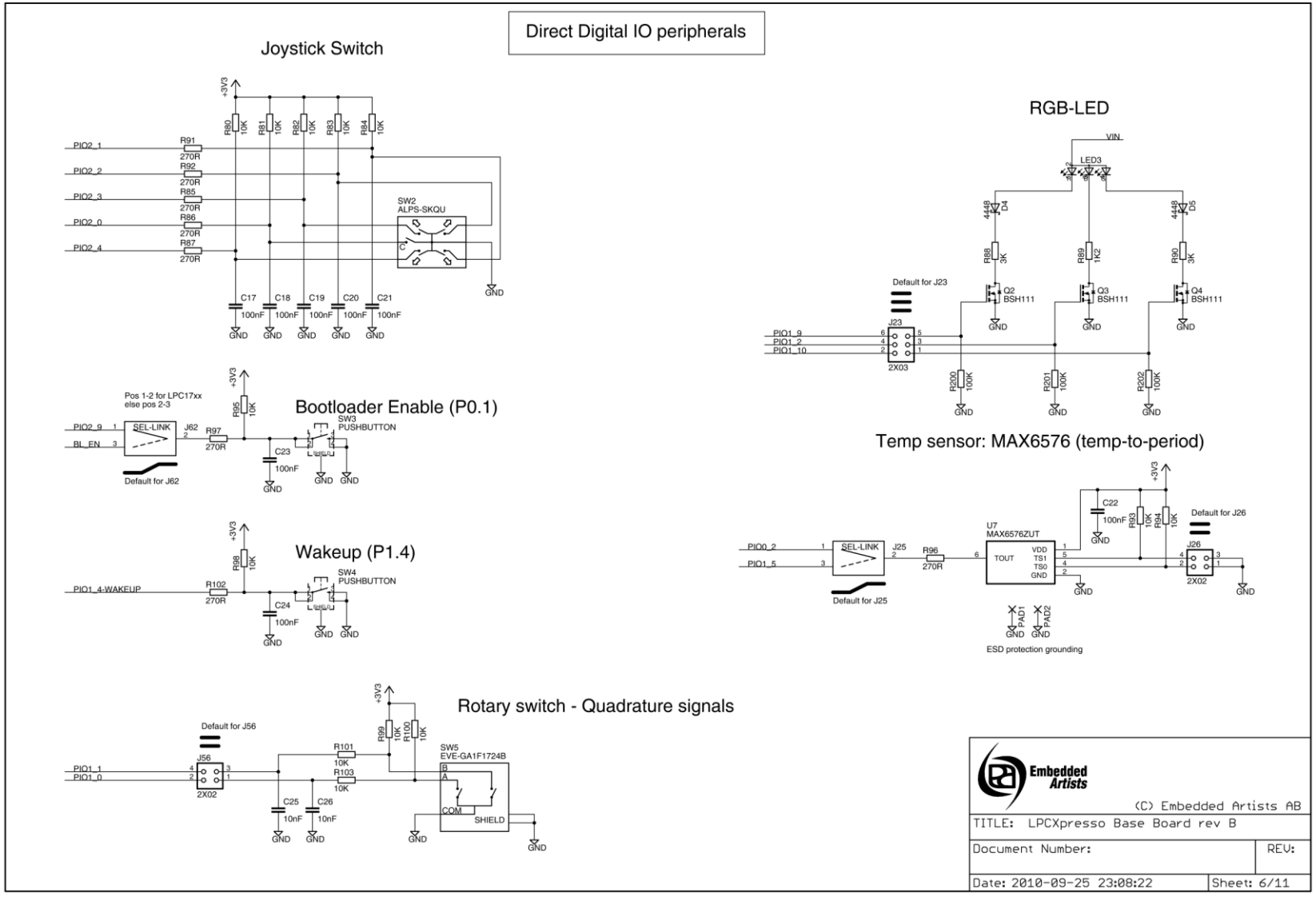
Note: 19 pins between rows on LPC1768 are not connected.

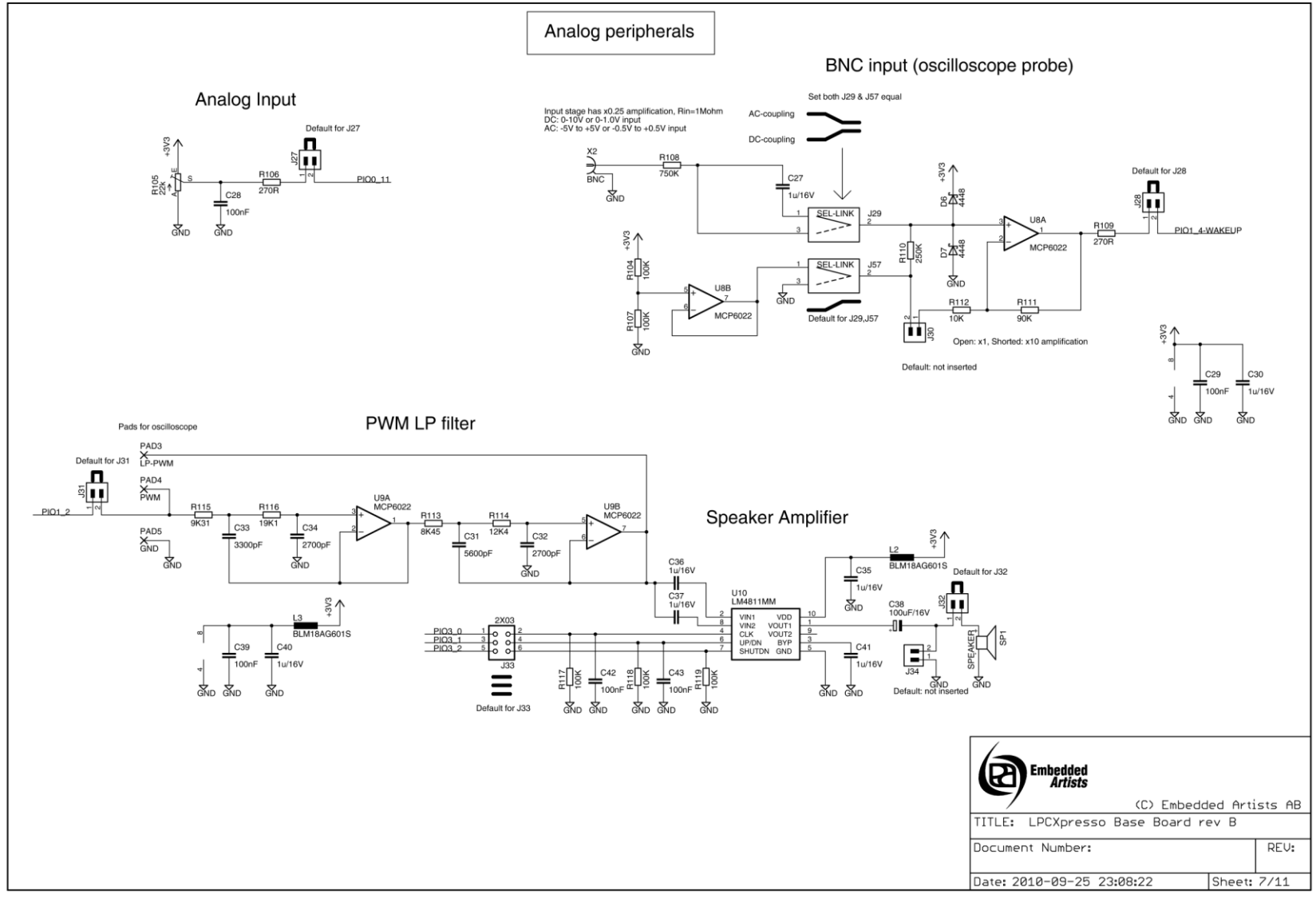
Embedded Artists
(C) Embedded Artists AB
TITLE: LPCXpresso Base Board rev B
Document Number:
REV:
Date: 2010-09-25 23:08:22
Sheet: 2/11

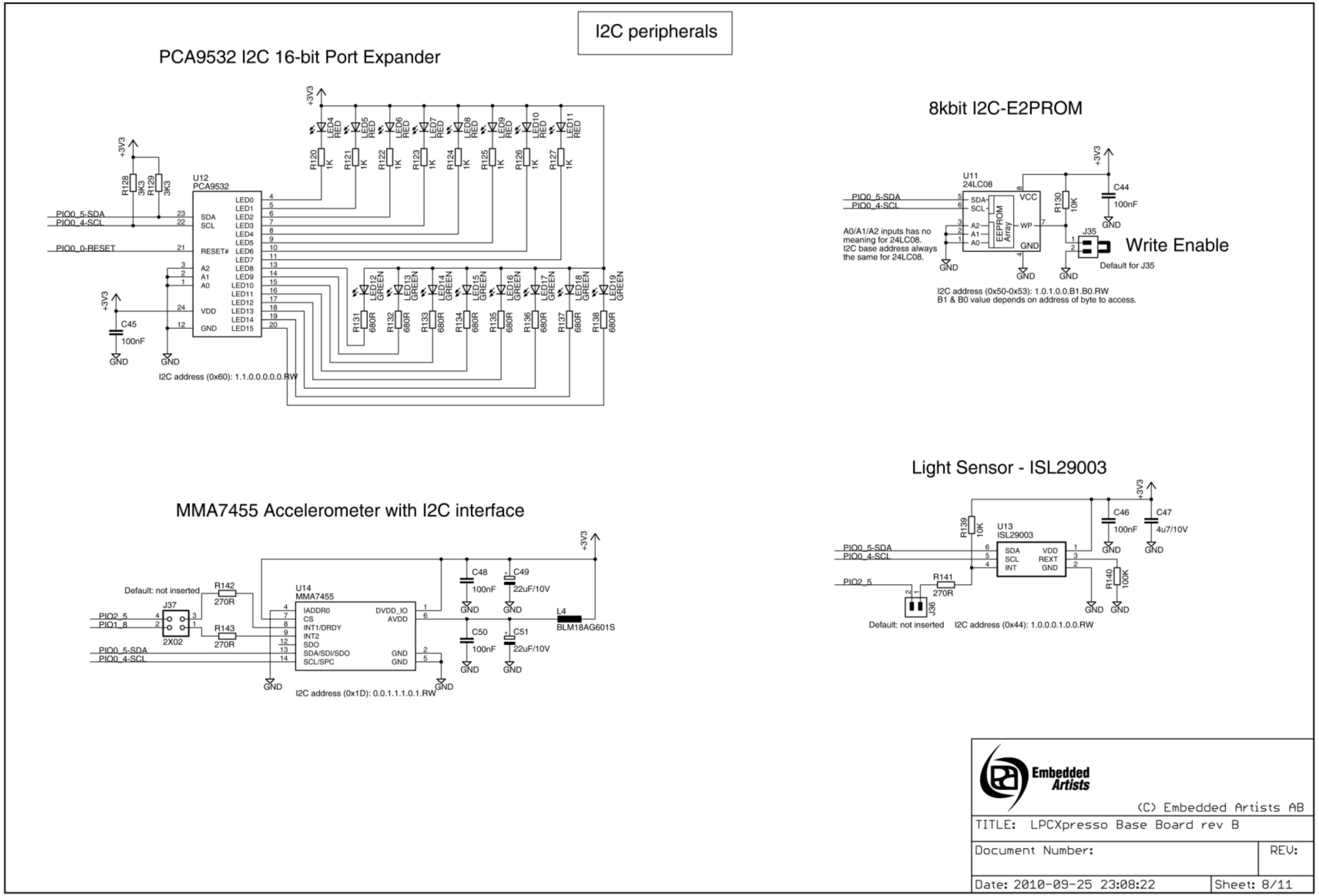


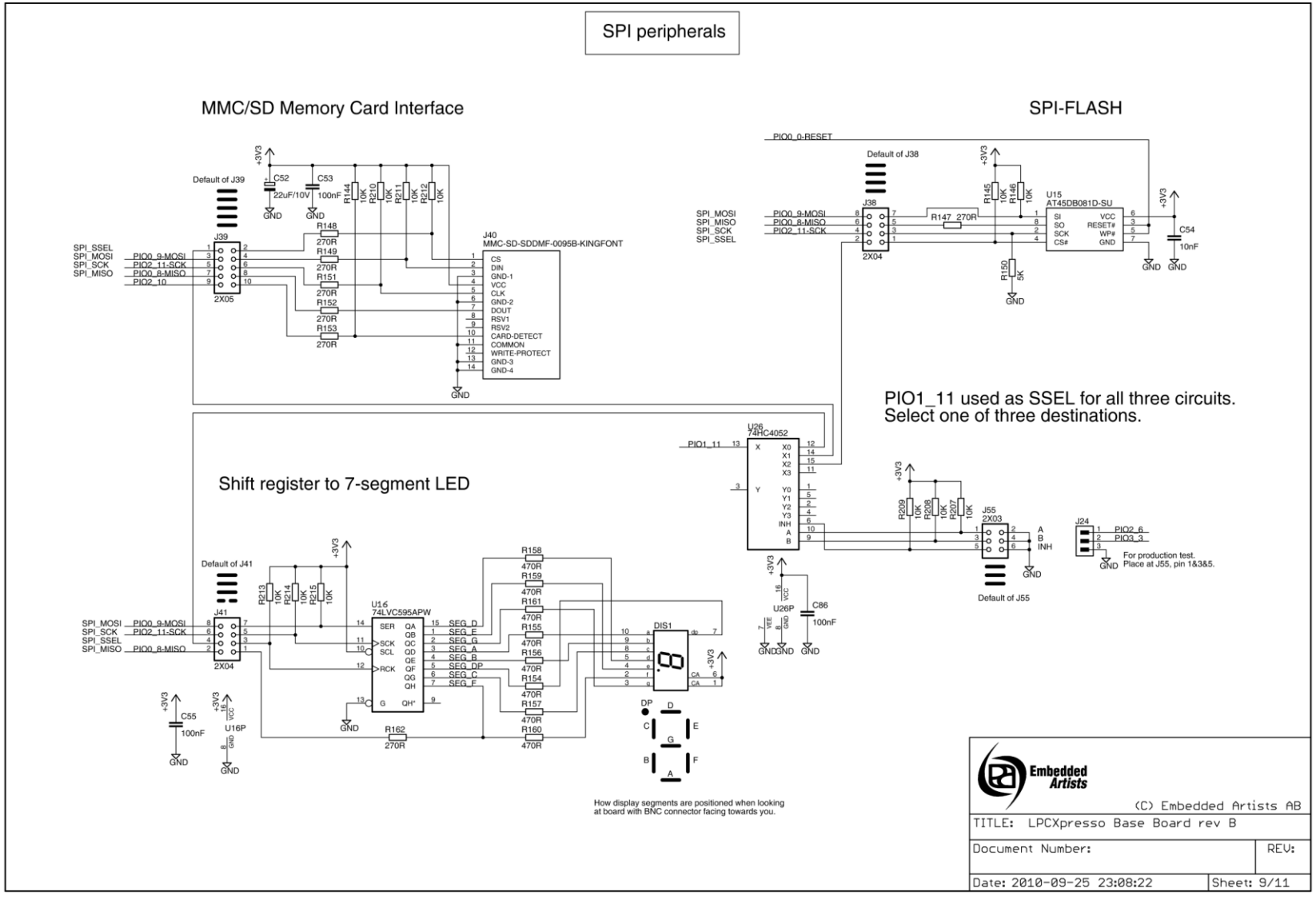






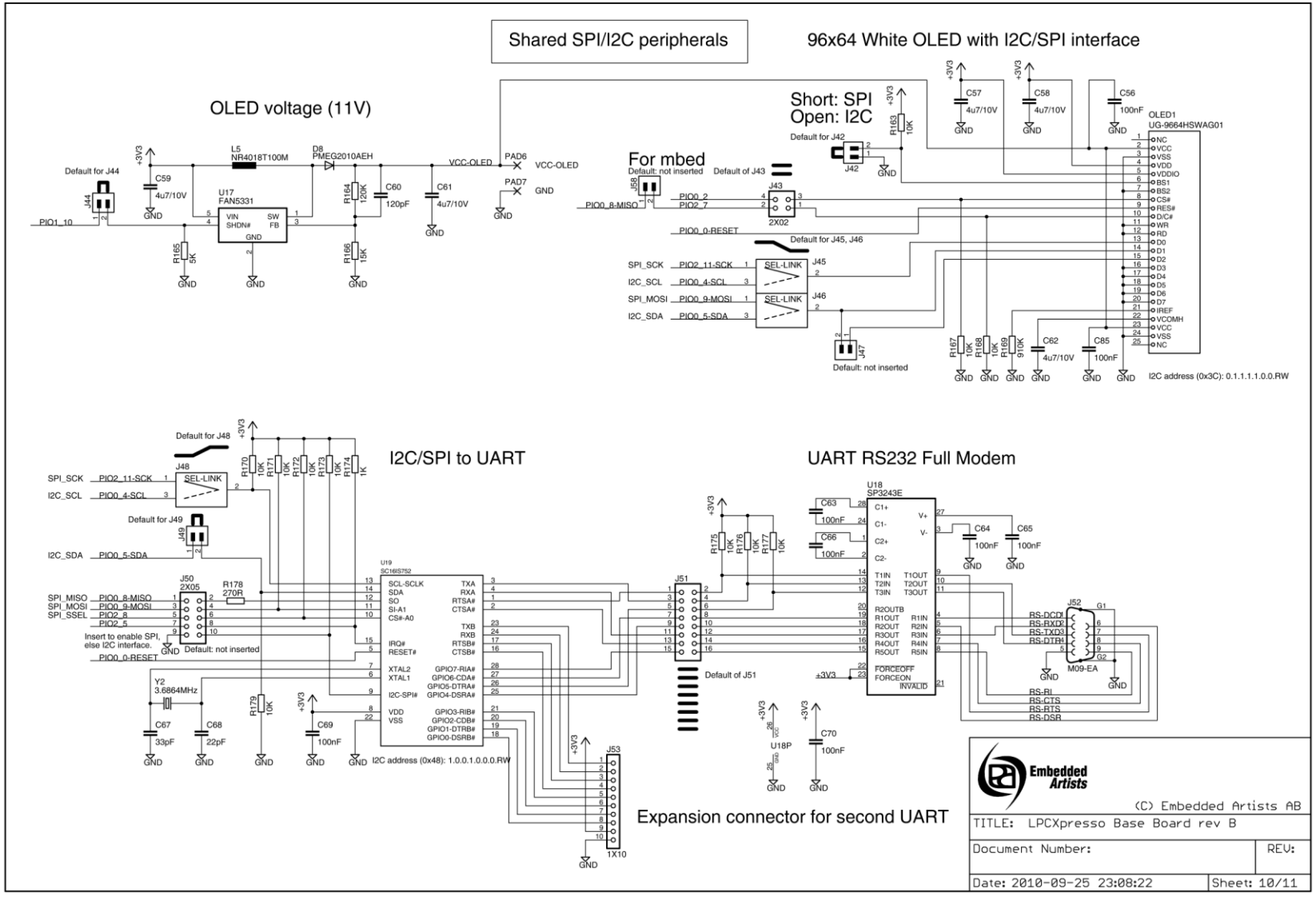


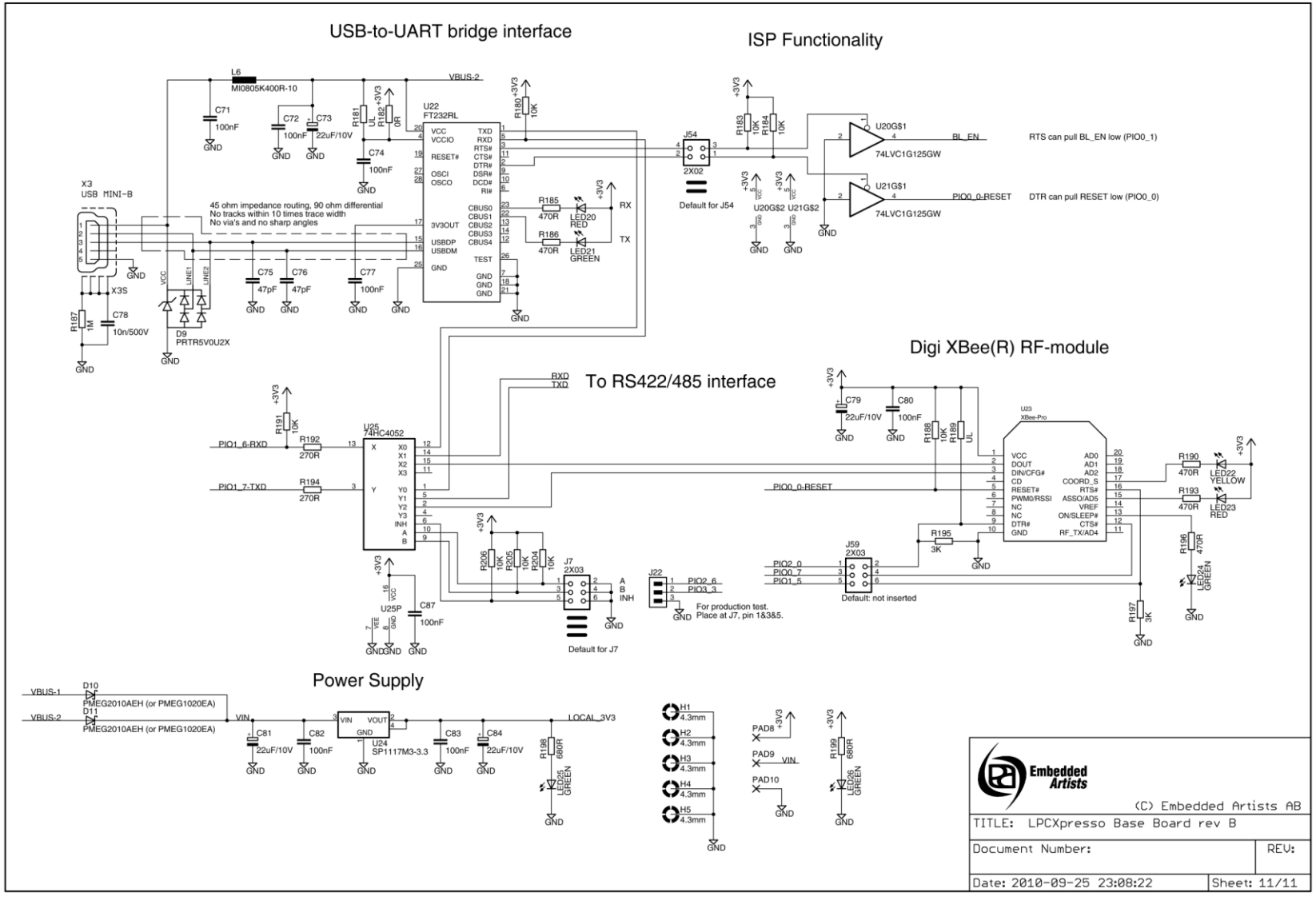




SORTED INTERFACES

SELECTED DEVICES



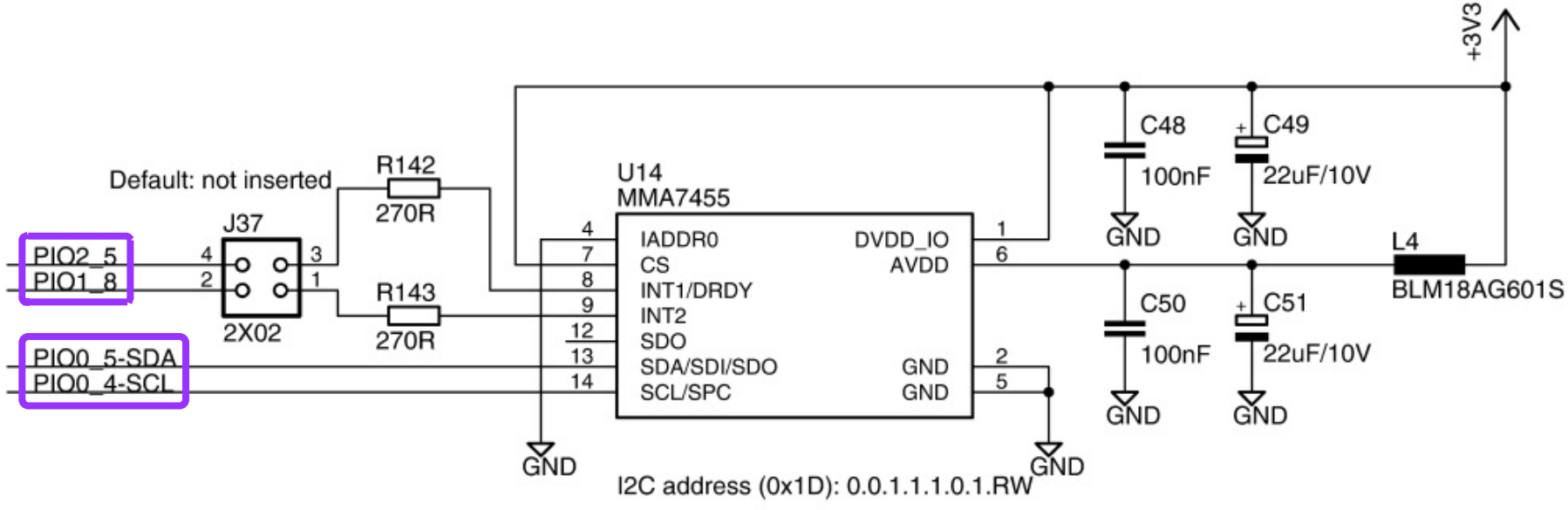


Show Device Location on BB

Main Interface: I²C

Show More Devices With a Similar Interface

MMA7455 Accelerometer [Full Schematic]

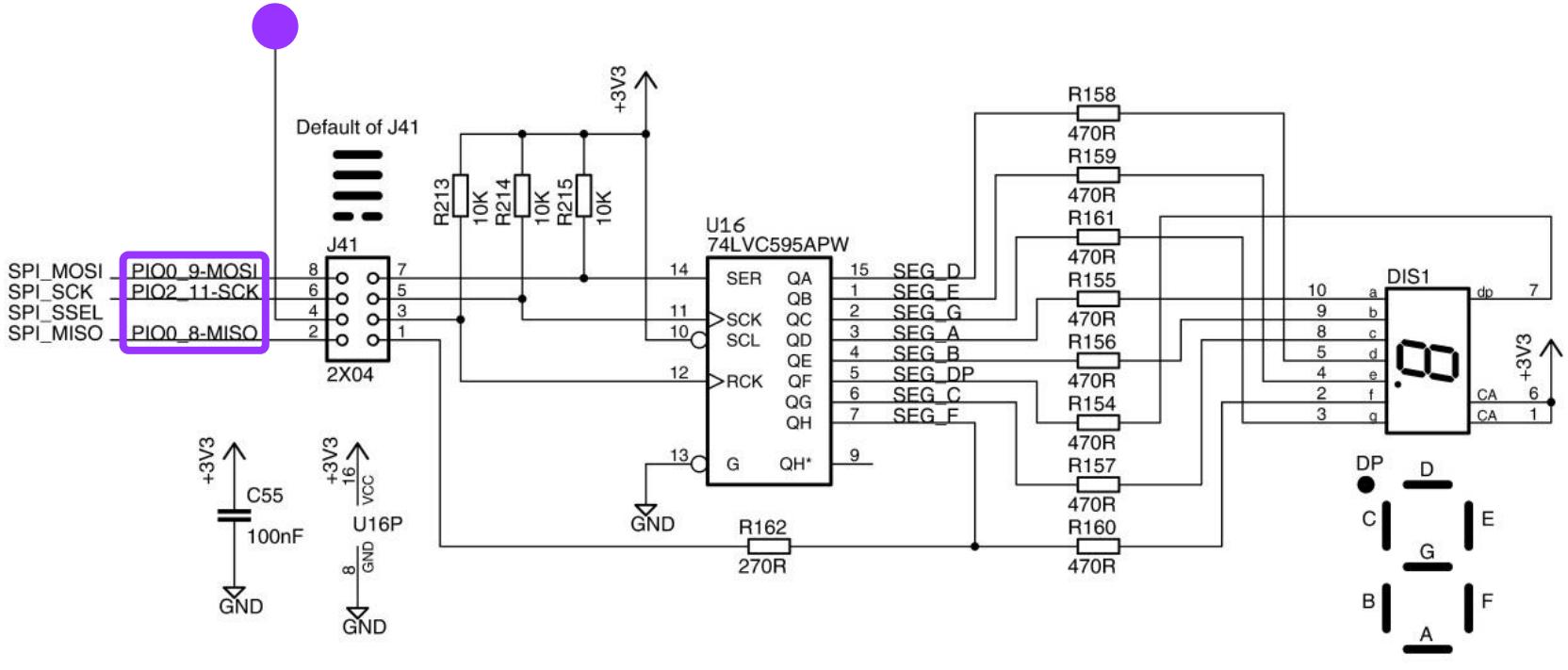


Show Device Location on BB

Main Interface: SPI

Show More Devices With a Similar Interface

Shift Register to 7-Segment LED [Partial Schematics]



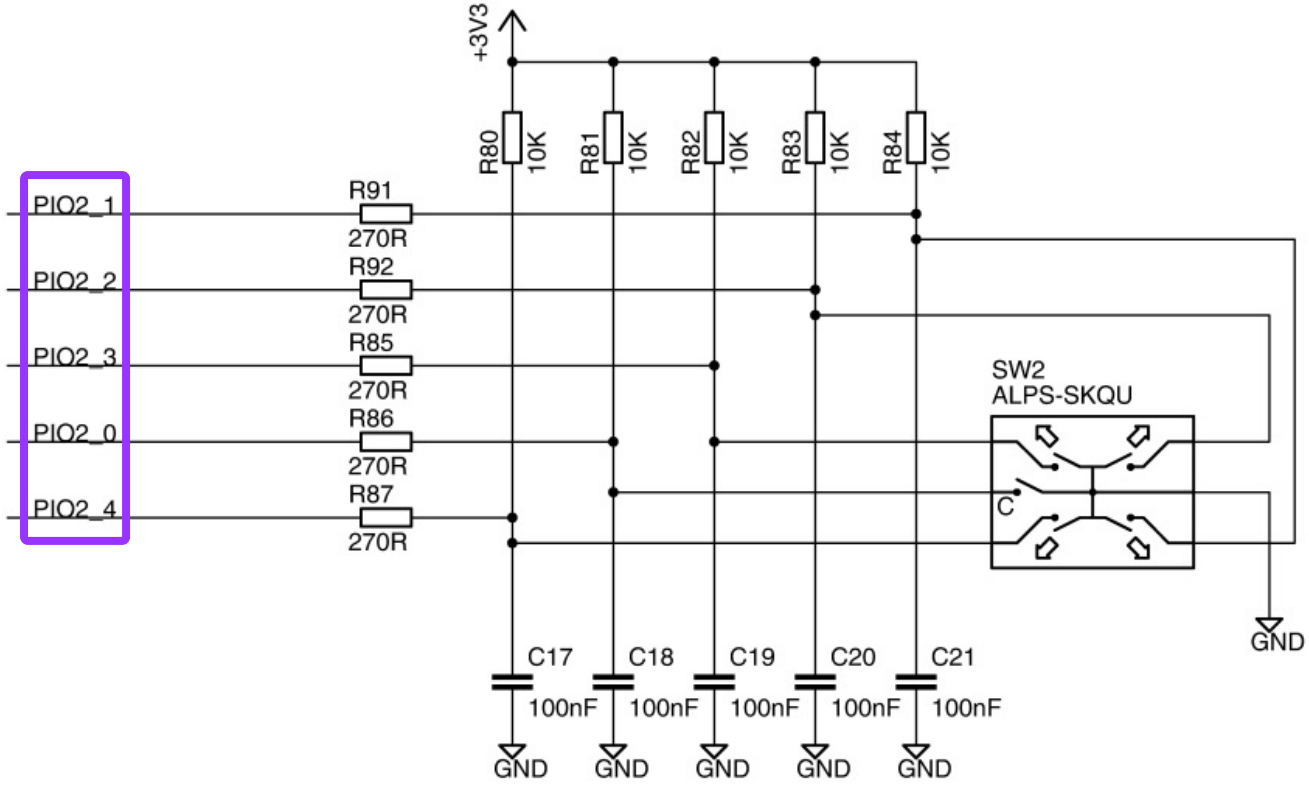
How display segments are positioned when looking at board with BNC connector facing towards you.

Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

Joystick Switch SW2 [Full Schematics]

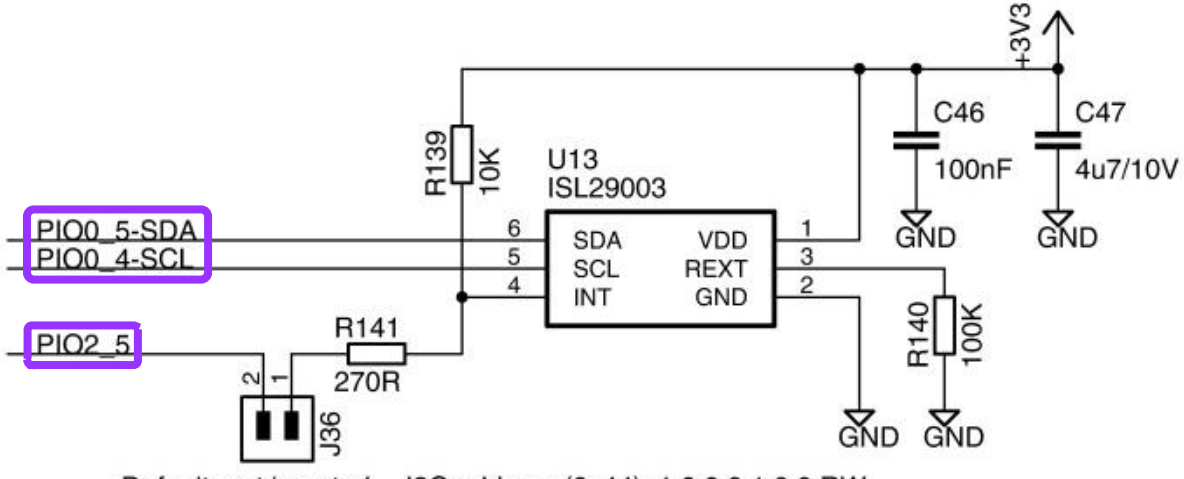


Show Device Location on BB

Main Interface: I²C

Show More Devices With a Similar Interface

ISL29003 Light Sensor [Full Schematics]

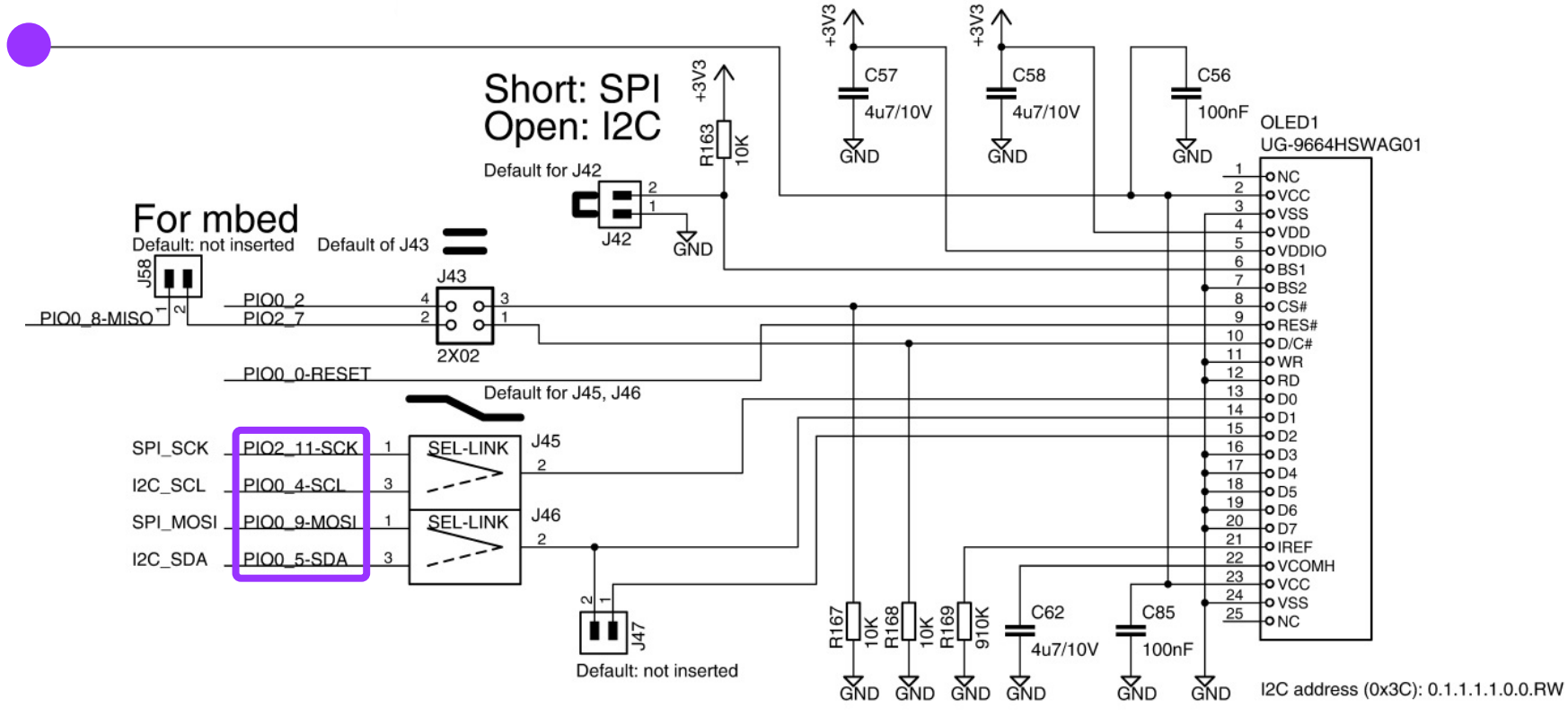


Show Device Location on BB

Shared Interface: I²C/SPI

Show More Devices With a Similar Interface

96x64 White OLED With I²C/SPI Interface [Partial Schematics]

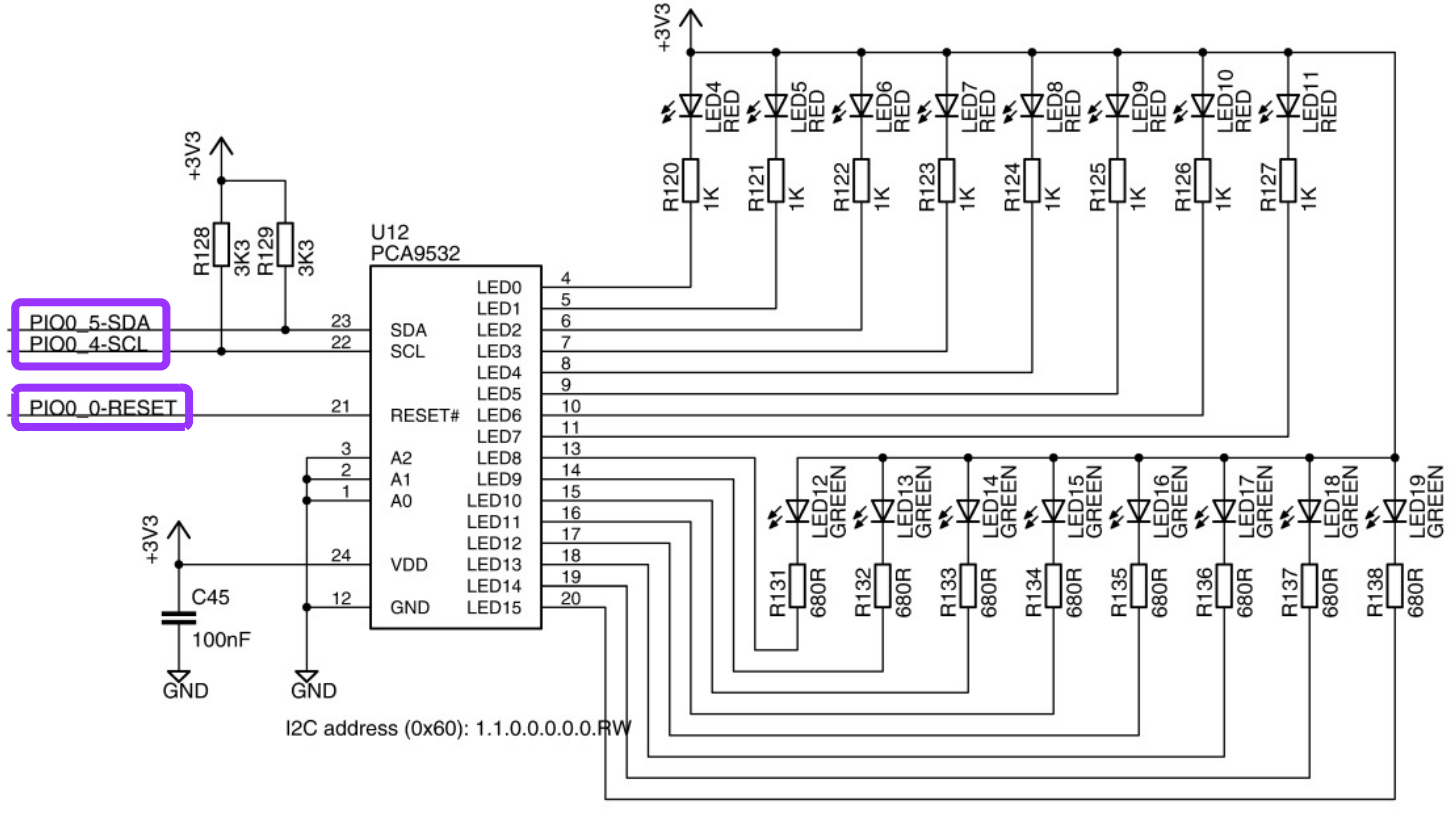


Show Device Location on BB

Main Interface: I²C

Show More Devices With a Similar Interface

PCA9532 I²C 16-bit Port Expander [Full Schematics]

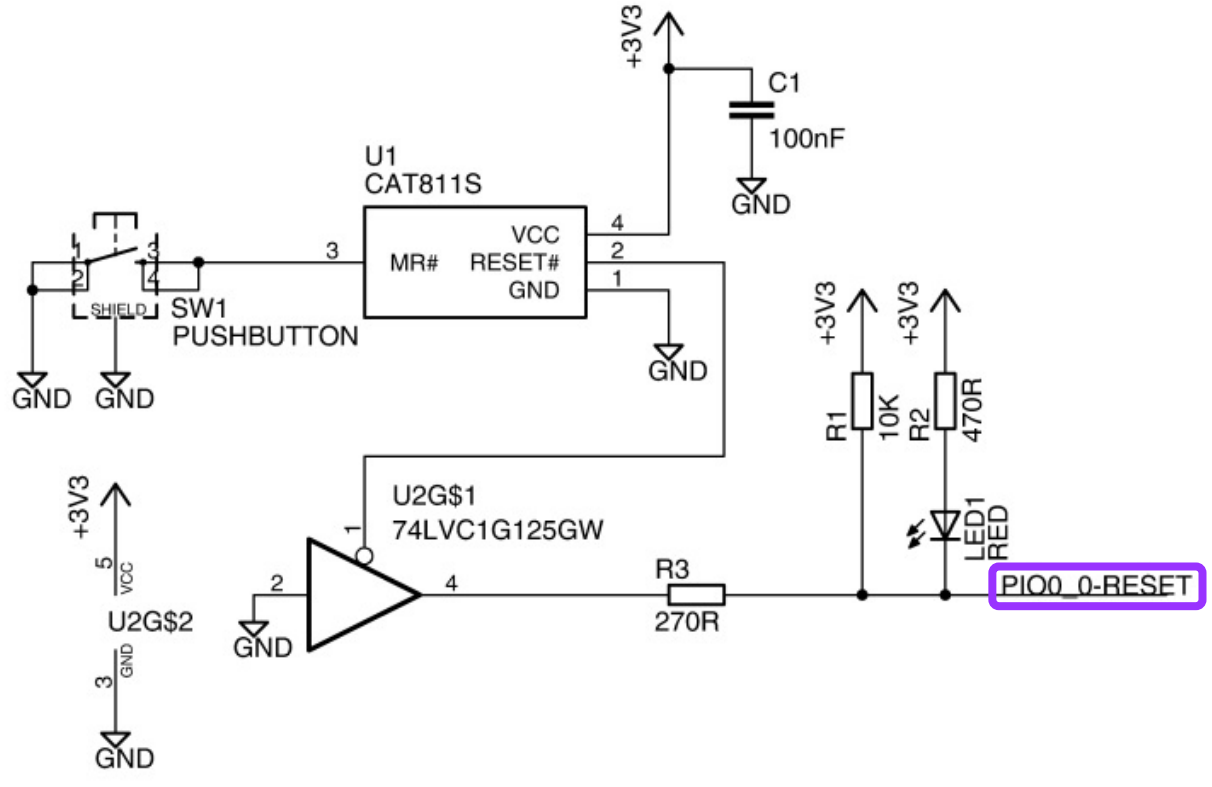


Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

Pushbutton SW1 [Full Schematics]

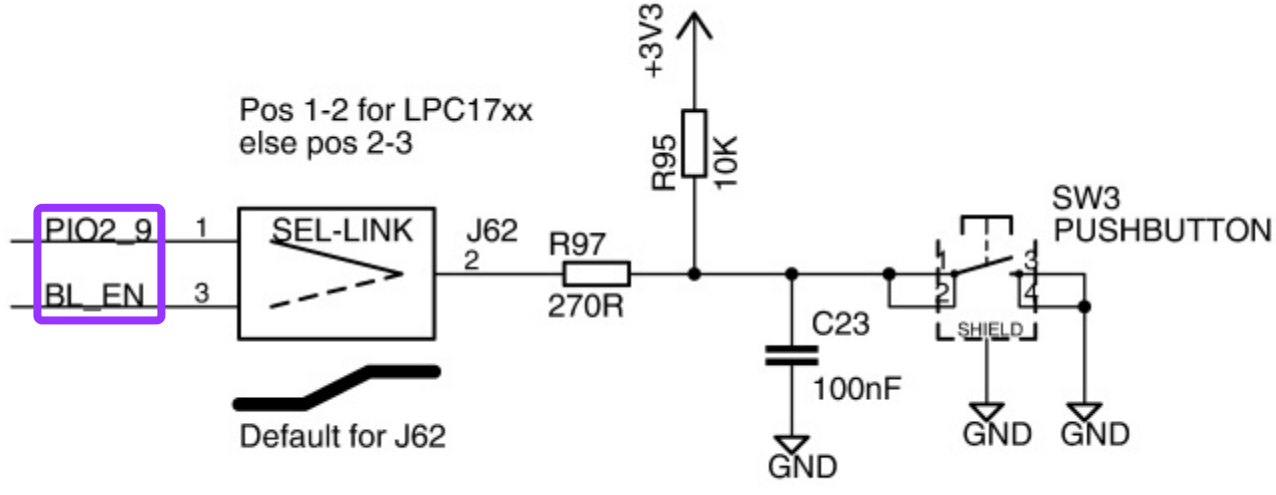


Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

Pushbutton SW3 [Full Schematics]

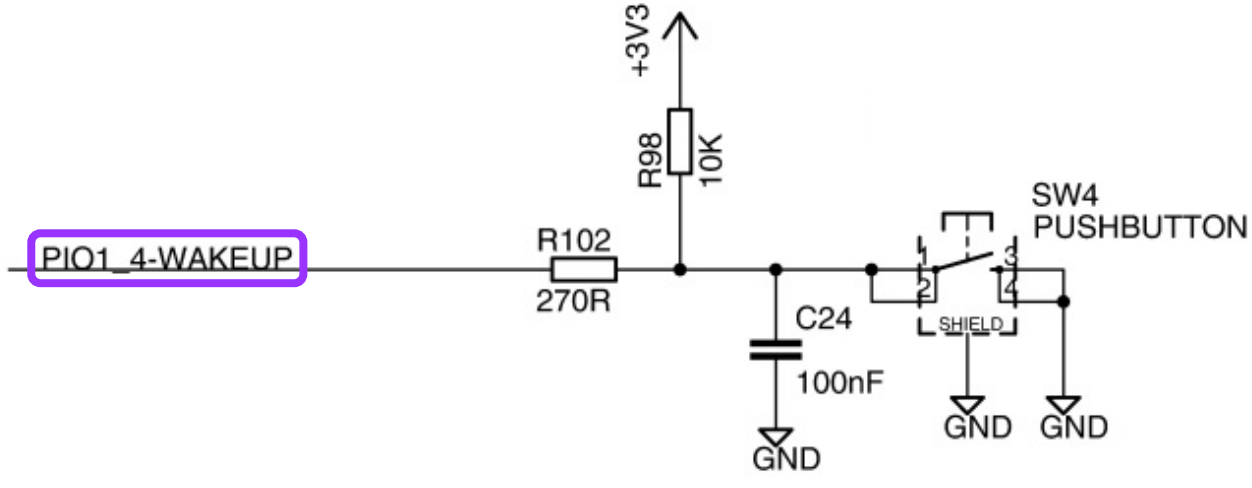


Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

Pushbutton SW4 [Full Schematics]

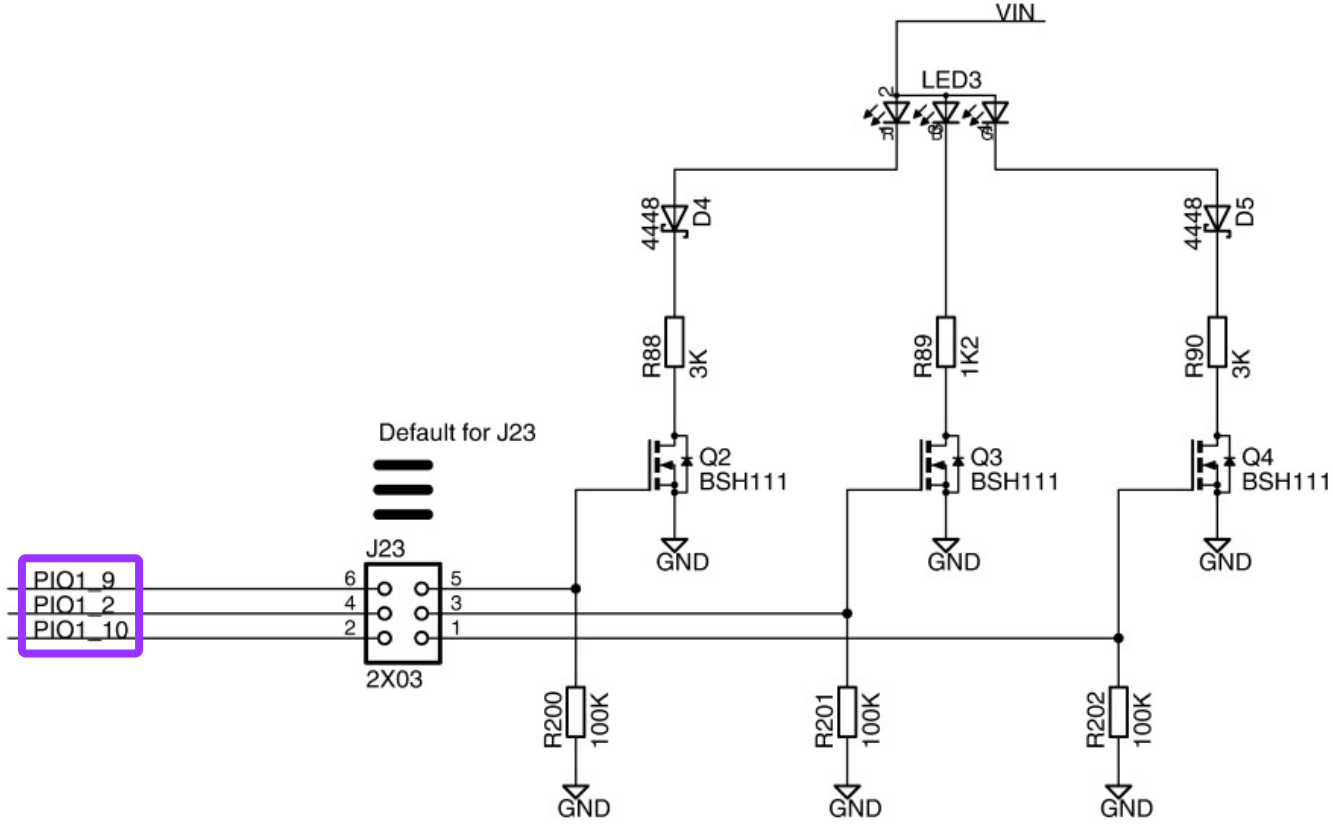


Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

RGB LED [Full Schematics]

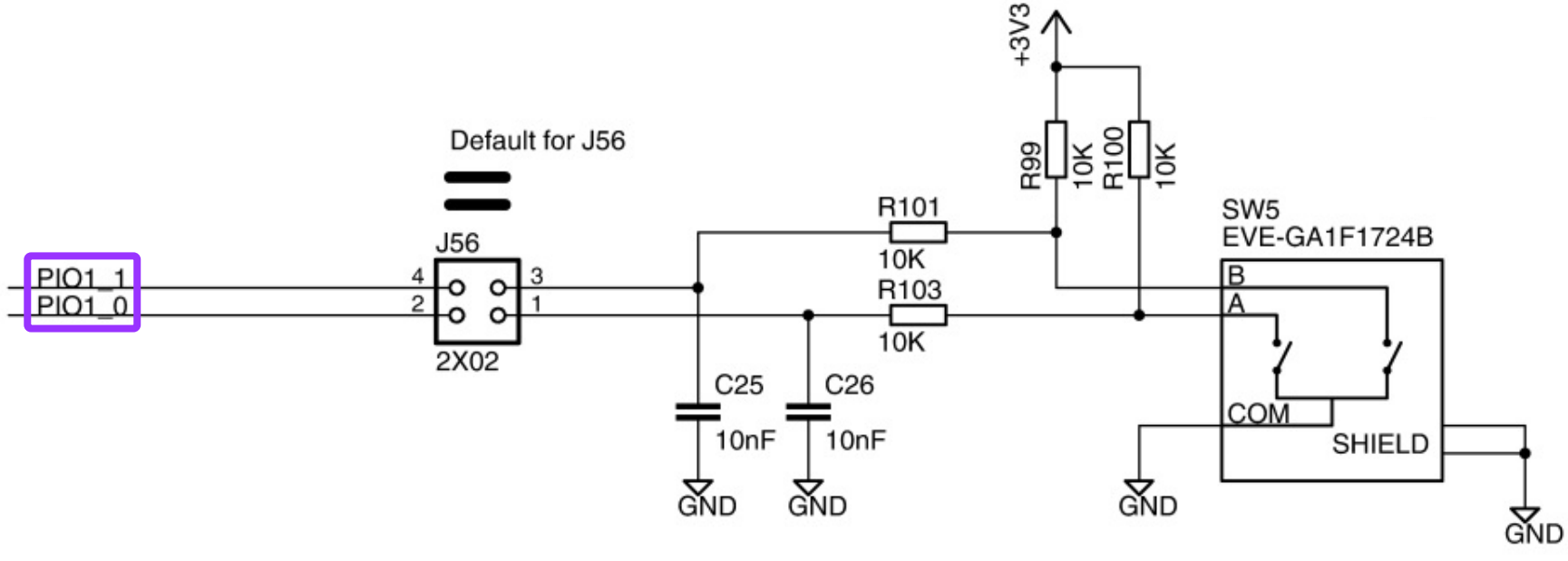


Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

Rotary Switch SW5 - Quadrature Signals [Full Schematics]

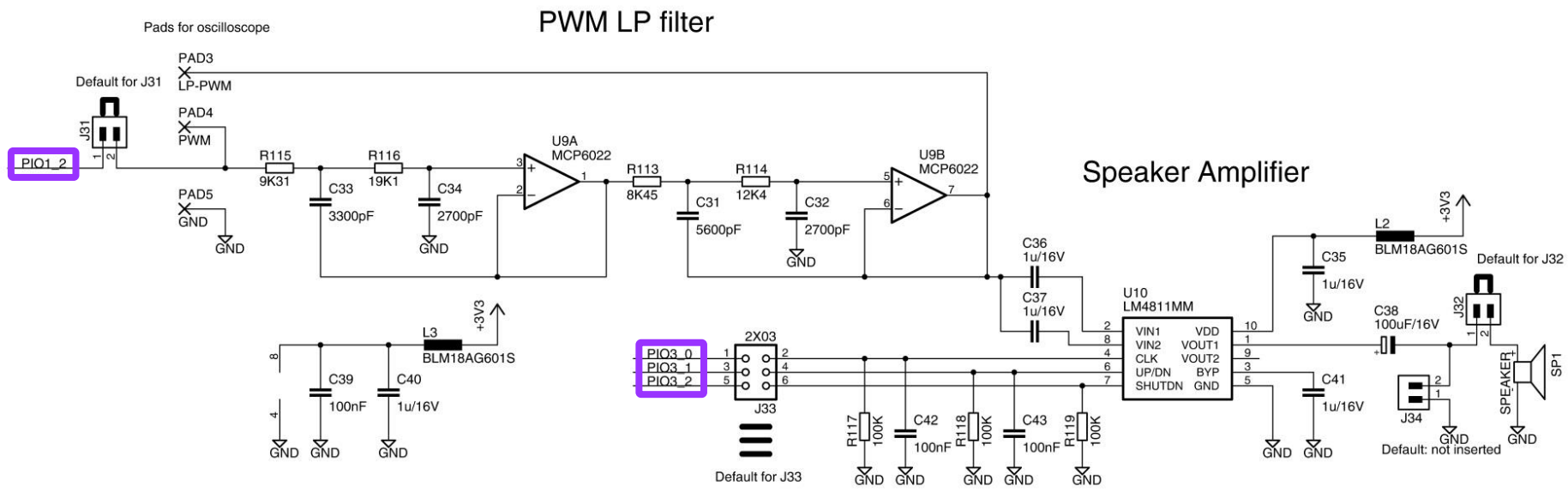


Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

Speaker Amplifier and PWM LP Filter [Full Schematics]

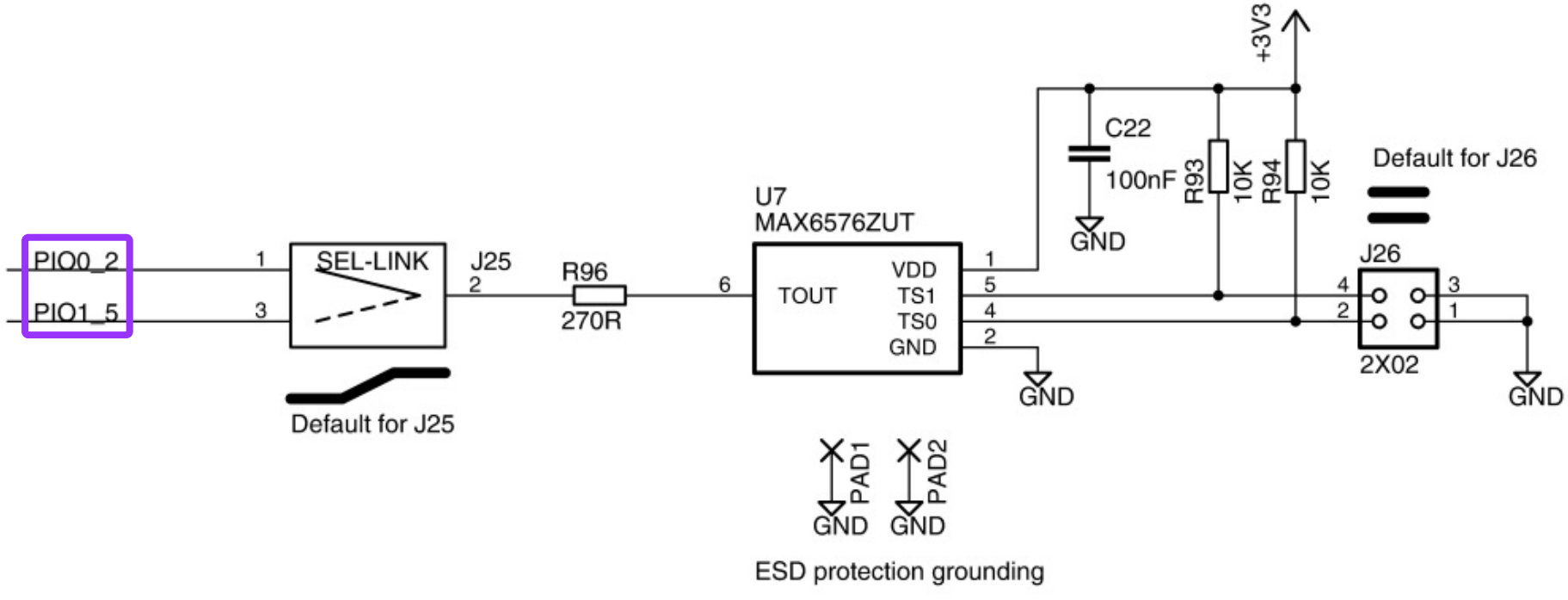


Show Device Location on BB

Main Interface: GPIO

Show More Devices With a Similar Interface

MAX6576 (Temp-to-Period) Temperature Sensor [Full Schematics]

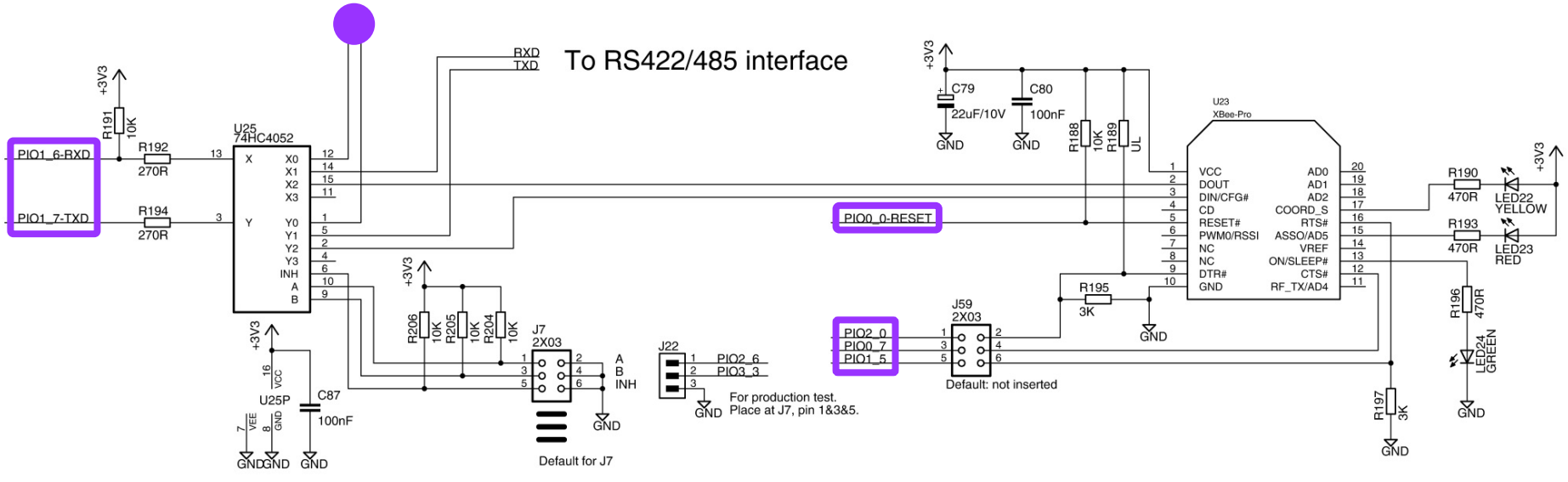


Show Device Location on BB

Main Interface: UART

Show More Devices With a Similar Interface

Digi XBee® RF-Module [Partial Schematics]



OVERVIEW OF THE ASSIGNMENT GUIDE TAB:

⇨ **Design guidelines**
- Some jumper settings will be recommended before starting your assignment 2
- General advices to make the coding and debugging manageable

⇨ **Care specific**
- Additional information for the assignment 2 of AY1617,S1
- Quick introduction to the illustration of system designs through flowcharts

⇨ **Device mappings**
- Quick reference linking the base board devices to the LPC1769 pins
- Simplified pin mapping table

Release Note: *Locked contents might be released after specific weeks depending on the semester's learning requirements and needs*

Design guidelines

⇨ The default jumper positions are illustrated in section 3.6, page 15, of the base board user guide:

LPCXpresso_BaseBoard_rev_B_Users_Guide_Rev_PA18.pdf

Some minor jumper position changes have to be done before starting the project. You are encouraged to understand why the jumpers need to be set as such by referring to the appropriate supporting document.

- Remove the 2 jumpers at J54
- Insert 1 jumper at J58
- Remove the GREEN-LED jumper from J23

Note: The use of some peripherals, for example, the SW3 pushbutton or GPIOs, may need further changes of the jumperconfigurations. For further details, refer to:

LPCXpresso_BaseBoard_rev_B_Users_Guide_Rev_PA18.pdf

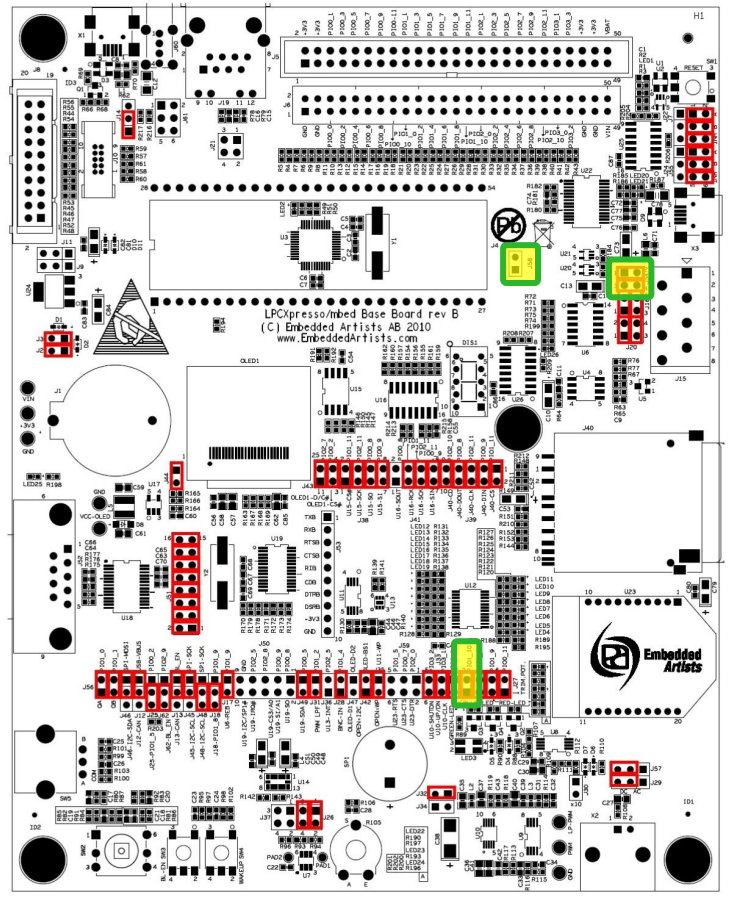


Figure 10 – LPCXpresso Base Board default Jumper Settings

Design guidelines

⇨ In order to start developing Assignment 2, students can simply copy the demo project, that is, right click to the project name at the Project Explorer panel, paste, and type in the new project's name. After that, it is recommended to delete all the code in main.c file and write up the assignment 2's code from beginning

⇨ It is highly encouraged to perform debugging after a few lines of codes to check where potential errors might be located. The frequent usage of *printf* statements will help greatly in debugging, except in situations where processing speed is critical. printf (const char * format, ...) is the function that outputs messages to the console. The following header file will allow for the usage of *printf* statements:

```
#include <stdio.h>
```

stdio.h is the header in the C standard library that contains macro definitions, constants, and declarations of functions and types used for various standard input and output operations

⇨ The following code may be used as an example of the usage of *printf* statements:

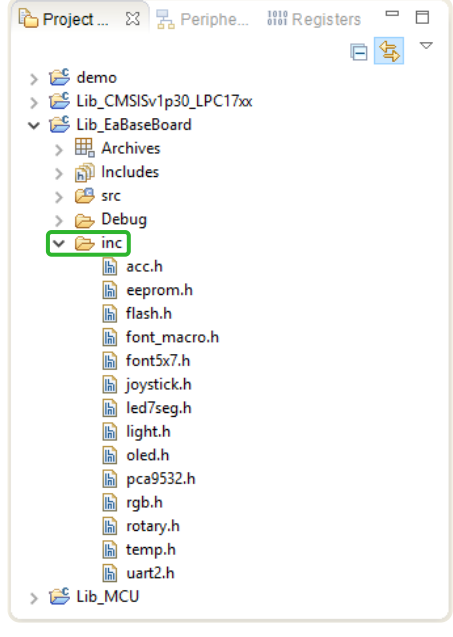
```
unsigned int my_variable;
my_variable = 22;
printf("Hello World \r\n");
printf("The value of my_variable is %u \r\n", my_variable);
```

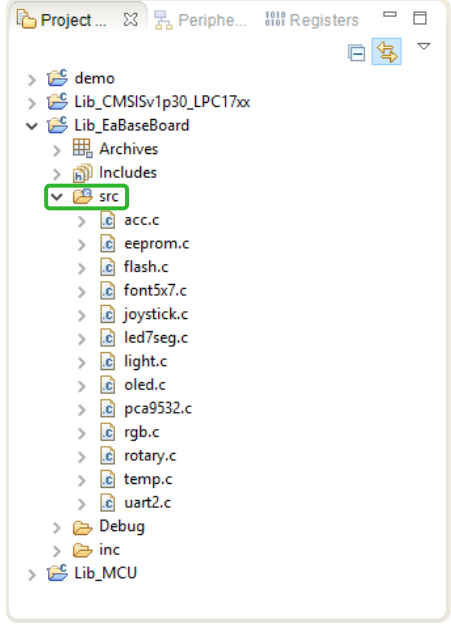
Please be reminded that the usage of *printf* statements in the final assessment is not allowed. *printf* should only be used for debugging purposes, and not during the final assessment

Design guidelines

⇨ It is recommended to implement functions instead of writing long sequences of codes. This will improve the readability of the codes, and make debugging easier

⇨ The **Lib_EaBaseBoard** library supports a lot of useful helper functions for the different peripherals available on the base board. The peripheral functions can be found in folder **src**, while the function prototypes can be found in folder **inc**. Students are highly encouraged to explore the **inc** and **src** files to find functions that will be helpful for their assignment's extras and extensions.





CARE flowchart requirement

⇨ For this assignment 2, the **CARE** scenario need to be described through the use of flowcharts, particularly focusing on the **PASSIVE** and **DATE** mode of **CARES**. The following points may be used as guideline:

- For the final report, about 3-4 pages of flowcharts are recommended, totalling an estimated 80 to 120 flowchart shapes. These amounts exclude any flowcharts illustrated in this assignment guide
- For the final presentation [PCI-8], please select and explain a few flowcharts which you think are most important for your program, and that are complex enough. Generally, this work out to be around 40 to 60 flowchart shapes. These amounts exclude any flowcharts illustrated in this assignment guide

⇨ A brief description of some basic flowcharts that you might use is also provided here

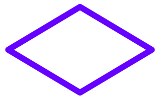
⇨ **Hint:** A simple way to use a timing peripheral to control the sampling or update rate of a device is illustrated here as a flowchart diagram. This is not the only possibility for timing control, but it gives a general idea on how a specific device, such as the seven segment on **CARES**, can be updated periodically

Basic flowchart shapes

Start / End:
Terminal symbols used at the beginning or finishing part of a process

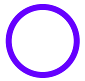
Process:
A brief description of what is happening. It can have only one entry point and one exit point

Sub-process / Subroutine / Pre-defined process:
For process that is further described through another set of flow chart



Decision:
The path to take. It can have only one entry point, one *True* exit point and one *False* exit point

Input / Output:
For reading or receiving data, and for generating data or updates



Connector:
To connect to another page or section of the flow chart, and whereby a line cannot be drawn

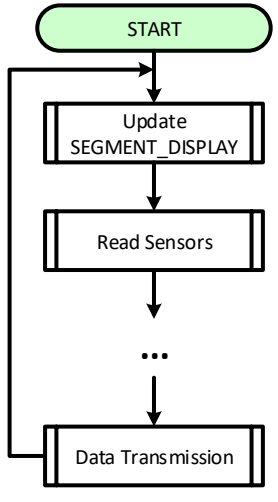
An example for the timing control of the CARES segment display

Values upon powering on CARES

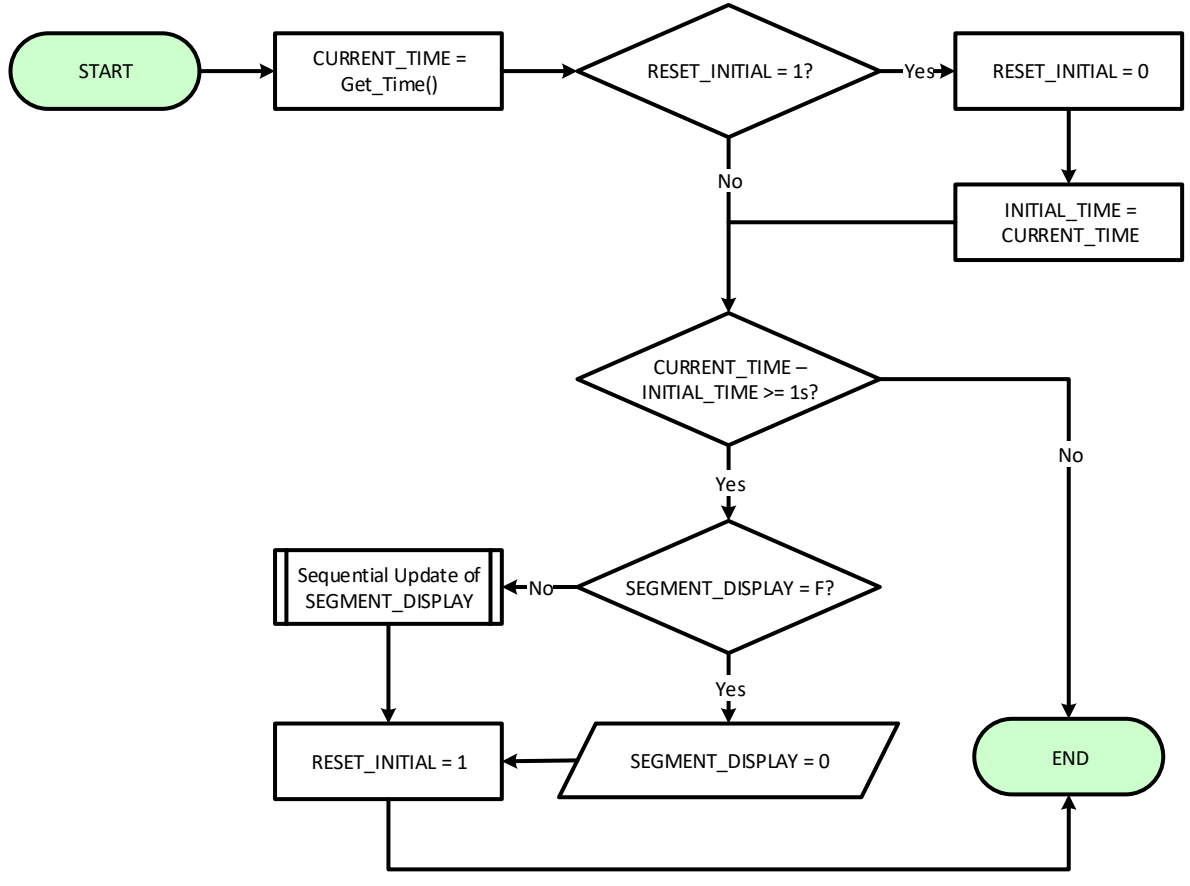
RESET_INITIAL = 1
INITIAL_TIME = 0
CURRENT_TIME = 0
SEGMENT_DISPLAY = 0

Get_Time() is a user-defined function

Main function



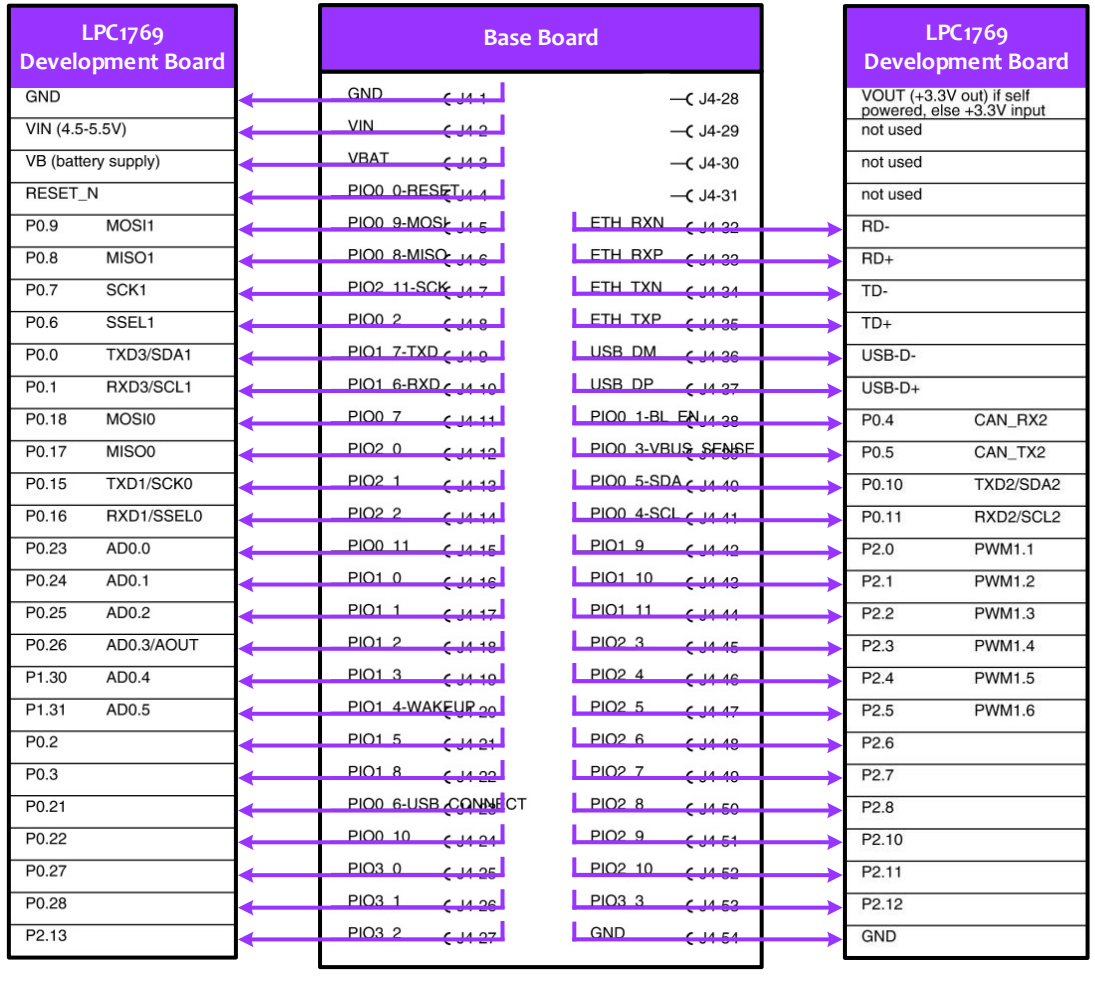
Update SEGMENT_DISPLAY



LPC1114/LPC1343	LPC176x	mbed	SWD	JTAG	USB	ETH	CAN	USB-to-UART	RF-module	RS422/ RS485	Joystick	PB-SW3	PB-SW4	Rotary Switch	RGB-LED	Temp-sensor	AIN-trimpot	BNC	PWM-to-Aout	PCA9532	MMA7455	24LC08	Light Sensor - ISL2900	SD/MMC I/F	7-segment	Dataflash	OLED	Dual UART
Reset / PIO0_0	RESET_N	nRES	reset	reset				reset	reset											reset						reset	reset	reset
PIO0_1 / CLKOUT / CT32B0_MAT2 / USB_FT	P0.4	can_rx2					use	bl_en				bl_en																
PIO0_2 / SSEL / CT16B0_CAP0	P0.6	gpio														Alt.2											ssel	
PIO0_3 / USB_VBUS	P0.5	can_tx2			use		use																					
PIO0_4 / SCL	P0.11	i2c_scl																		scl	scl	scl	scl				scl	scl
PIO0_5 / SDA	P0.10	i2c_sda																		sda	sda	sda	sda				sda	sda
PIO0_6 / USB_CONNECT / SCK	P0.21	not mbed			use																							
PIO0_7 / CTS	P0.18	gpio							cts																			
PIO0_8 / MISO/CT16B_MAT0	P0.8	spi1-miso																						spi-miso	spi-miso	spi-miso	spi-miso	spi-miso
PIO0_9 / MOSI / CT16B0_MAT1 / SWO	P0.9	spi1-mosi	(swo)																					spi-mosi	spi-mosi	spi-mosi	spi-mosi	spi-mosi
SWCLK / PIO0_10 / SCK / CT16B0_MAT2	P0.22		swclk	tck																								
TDI / PIO0_11 / AD0 / CT32B0_MAT3	P0.23	ain0		tdi													ain											
TMS / PIO1_0 / AD1 / CT32B1_CAP0	P0.24	ain1		tms										Qa														
TDO / PIO1_1 / AD2 / CT32B1_MAT0	P0.25	ain2		tdo										Qb														
TRST / PIO1_2 / AD3 / CT32B1_MAT1	P0.26	ain3/aout		trst											blue				pwm-out									
SWDIO / PIO1_3 / AD4 / CT32B1_MAT2	P1.30	ain4	swdio																									
PIO1_4 / AD5 / CT32B1_MAT3 / WAKEUP	P1.31	ain5											wakeup					ain										
PIO1_5 / RTS / CT32B0_CAP0	P0.2	not mbed							rts							Alt.1												
PIO1_6 / RXD / CT32B0_MAT0	P0.1	uart1-rx						RXD	RXD	RXD																		
PIO1_7 / TXD / CT32B0_MAT1	P0.0	uart1-tx						TXD	TXD	TXD																		
PIO1_8 / CT16B1_CAP0	P0.3	not mbed								TX-EN											irq2-out							
PIO1_9 / CT16B1_MAT0	P2.0	pwmout0								RX-EN#					red													
PIO1_10 / AD6 / CT16B1_MAT1	P2.1	pwmout1													green												Voled-en	
PIO1_11 / AD7	P2.2	pwmout2				led1																		spi-ssel	spi-ssel	spi-ssel		
PIO2_0 / DTR	P0.17	gpio							dtr		center																	
PIO2_1 / DSR	P0.15	gpio									down																	
PIO2_2 / DCD	P0.16	gpio									right																	
PIO2_3 / RI	P2.3	pwmout3									up																	
PIO2.4 (1343) or PIO3_4 (1114)	P2.4	pwmout4									left																	
PIO2.5 (1343) or PIO3_5 (1114)	P2.5	pwmout5				led2															irq1-out		irq-out					irq
PIO2.6	P2.6	not mbed																										
PIO2.7	P2.7	not mbed																									d/c	
PIO2.8	P2.8	not mbed																										spi-ssel
PIO2.9	P2.10	not mbed																										
PIO2.10	P2.11	not mbed																						cd				
PIO2_11 / SCK	P0.7	spi1-sck																						spi-sck	spi-sck	spi-sck	spi-sck	spi-sck
PIO3.0	P0.27	not mbed																	sp-clk									
PIO3.1	P0.28	not mbed																	sp-u/d									
PIO3.2	P2.13	not mbed																	sp-sh									
PIO3.3	P2.12	not mbed																										

mbed module "problem"

Simplified pin mapping table



I Want Useful Contents ☺

This is the last page ...

(◠‿◠;)

ABOUT THIS DOCUMENT

This document was made possible by taking note of implementable feedbacks and difficulties faced by previous EE2024(E) students over multiple semesters. Should there be suggestions to this document, please inform me directly.

I hope that by having this master document to help you in the EE2024(E) assignment 2, you will be able to learn the important practical parts of the subject matter faster but with lesser stress, and without the need to have multiple PDF documents for basic tasks.

This document format is optimised with hyperlinks for 16:9 screens, to make it easy for hardworking students to refer to anytime on their smartphones ☺

Thanks for your support,
Chris M. Shin LLC
☺ Caring for your EE2024(E) assignment 2 learning journey ☺