IP50
Embedded Software Engineer
Python Programming

www.ctione.org

Version 2.0

Summer 2019

Coordinator: Harry Li, Ph.D. Director
(650) 400-1116, Email: harry.li@ctione.com
CTI One Corporation
3679 Enochs Street
Santa Clara, CA 95051

Contributors

2017-10	Establish this document	Harry Li
2018-10	Add MOTDRV Chapter	Zhixuan Zhou
2018-10	Add Ethernet and IP networking Chapter	Jerry Lee
2019-5-24	Update with MNIST, and remove other MOTDRV, remove IP networking	Harry Li

Table of Content

I. Full Stack Embedded Software Engineering For High School Programming/Robotics Summer School (3 Project Guidelines)
II. Training Program Schedule
III. Getting Started with GPIO Interface Design and Implementation
3.1 Design requirements
3.2 Assessment
3.3 Design Notes
IV. Handwritten Digits Detection in Deep Learning with Python
4.1 Design requirements
4.2 Assessment
4.3 Design Notes
V. IoT Sensor Interface and Greenhouse Automation Design and Implementation
5.1 Design requirements
5.2 Assessment
5.3 Design Notes
VI. Robot (6 DoF) pick and place or Autonomous Driving Vehicle Motor Controls (students select one of these 2 options).

Appendix A. Project Guidelines Meeting Notes

Appendix B. IEEE Style Report Template

I. Full Stack Embedded Software Engineer Training 6 Project Guidelines

Project	Description	Assessment
1. Getting Started with GPIO Interface Design and Implementation	1. Build the prototype board to realize the function of power regulator circuit, and GPP input and output testing circuit; 2. For LPC1769 Install MCU xpresso (the newer version) or Xpresso (you can still use it). Then Import the testing program, GPIO testing program, be sure to import 1769 patch program so the GPIO testing program can run properly; or For Pie-3 install OS on the board flash; 3. For Group1: Program Python and run the python code, demo on Pie-3 board 4. For Group2: Compile and build the program for C code, then test your prototype design. Once it worked, make demo.	See Rubrics
2. For Group 1: handwritten digits recognition in Deep Learning and Python/C++ Implementation	1. Learning Python/C++ with OpenCV to capture digital images and prepare image training database; 2. Learning Python/C++ for image preprocessing and enhancement; 3. Learning Neural Networks and programming in Tensor-flow and Keras for deep learning in facial detection; 4. Training deep learning convolutional neural network models; 5. Design and implement a prototype to realize facial detection functions.	
2. For Group 2: Facial Detection in Deep Learning and Python/C++	1. Learning Python/C++ with OpenCV to capture digital images and prepare image training database;	

Implementation	2. Learning Python/C++ for image preprocessing and enhancement; 3. Learning Neural Networks and programming in Tensor-flow and Keras for deep learning in facial detection; 4. Training deep learning convolutional neural network models; 5. Design and implement a prototype to realize facial detection functions.	
3. **For Group 2 only:** IoT (Internet of Things) Sensor Interface and Greenhouse Automation Design and Implementation	1. Design and prototype micro-processor system board, and enable ADC interface and digital interface for sensor input; 2. Programming Python/C++ to read sensor data and process sensor data to remove noise and to pot data curve; 3. Program micro-processor to drive GPIO port for lighting and actuation control (temperature, irrigation valve, etc.) 4. Program micro-processor timer and interrupt to realize timed payload deployment.	See Rubrics
4. **For Group 2. Only:** Robot (6 DoF) pick and place	Robot Option: 1. Learn robot control and programming interface principles; 2. Learning how to operate robot by using hand-held teach penal; 3. Programming in Python/C++ to control 6 DoF (degree-of-freedom) industrial robot; 4. Program micro-processor to drive the robot to realize pick and place function.	
5. **For Group 2. Only:** Driving Vehicle Motor Controls .	Self Driving Option: 1. Learn self driving robot control and programming interface principles; 2. Learning how to operate self driving by using wireless hand-held teach penal; 3. Programming in Python/C++ to AGV2000 or AGV4000, self driving robot (carries 1000 lbs or 400 lbs); 3. Program micro-processor to drive the AGV2000 to realize category I route.	

II. Full Stack Embedded Software Engineer Intern Training Program Schedule (30 hours Lecture)

The schedule is subject to change with fair notice in class.

Table 2.1 Schedule

Lecture Hours/ Lab Hours	**Topics, Readings, Assignments, Deadlines**
Unit 1	Organizational Meeting and Introduction, Overview of a RISC Microprocessor System with CPU datasheet.
Unit 2	Review of the RISC CPU architecture: CPU core, peripheral controllers, internal buses, and memory controllers as well as graphics engine. Design a microprocessor system with CPU module and with RS232 debugging capability.
Unit 3	Continue to review RISC CPU and peripheral controllers, in particular to UART and SPI controller for design and building FLASH memory interface. I/O Interface Design, from UART, RS232 to RS485, SPI and IIC interface.
Unit 4	Memory Map, Power-up Address, ROM memory unit and its 8-bit, 16-bit, 32-bit banks design implementation. Description and design of a bidirectional system bus and I/O. System memories: SRAM and SRAM bus interface.
Unit 5	SPI Interface and SPI FLASH memory interface design, implementation of data logger based on SPI FLASH memory design.
Unit 6	External controller and implementation of ExINT techniques. Interrupt techniques and Timer design as well as applications based on interrupt timer design. Interrupt controller design and interface design. Advanced MCU design: GE (graphics engine) design, 2D graphics vector graphics processing for ARM Cortex Core. LCD Display adapter design. Implementation of LCD display with 2D GE vector graphics.
Unit 7	CPU Architecture Theory, Von Neumann Architecture. Intel CPU interrupt techniques, interrupt vector table, interrupt service routine (ISR) implementations.
Unit 8	Midterm
Unit 9	ADC interface design and FFT based data validation.
Unit 10	Advanced MCU design: GE (graphics engine) design, 3D graphics vector graphics processing, world to viewer transformations, and linear decoration algorithm for ARM Cortex Core. Implementation of 3D perspective projection and linear decoration algorithm.

III. Getting Started with Interface Design and Implementation (sample)

3.1 Design requirements

1. Build the prototype board to realize the function of power regulator circuit, and GPP input and output testing circuit;

2. Install MCU xpresso (the newer version) or Xpresso (you can still use it). Then Import the testing program, GPIO testing program, be sure to import 1769 patch program so the GPIO testing program can run properly;

3. Compile and build the program, then test your prototype design. Once it worked, make demo.

3.2 Assessment

1. General Guidelines

Use IEEE paper template to generate your report.

(1) One line title, factual and right-to-the-point, avoid marketing terms.

(2) Abstract (50-100 words) with design objectives, technical challenges,

methodology, hardware and software resources description, deliverables,

and the implementation result.

(3) Strickly follow the IEEE paper style, no modification of spacing, fonts,

section enumeration etc. be sure to provide Appendix section with source code and schematics.

2. System and Hardware Level Design

(1) provide system block diagram to capture the entire desing/testing/prototyping setup, for example, laptop computer with Ethernet/RS232 link to microprocessor system.

(2) Block diagram for the micropocessor system with detailed pin connectivity information, labels of each individual block of the system.

(3) Schematics of each basic building blocks and/or subsystems, and/or entire system.

(4) Photos of the actual implementation of the entire system and/or subsystems.

3. Software Design

(1) Description of the softweare development environment and its set up procedure, such as Eclips based IDE (integrated development environment) set

up.

(2) Algorithm description in a well-organized, step-by-step fashion, for example steps from 1 through 5.

(3) Flow chart(s) to give further details of the algorithm, if needed, multiple flow charts can be utilized.

(4) Pseudo code to match up the flow charts. Due to the nature of the hardware and software co-design, algorithmic type Pseudo code is usually too abstract, details down to the level of registers and bits patterns of registers are needed.

(5) Source code (segment of code) to support the Pseudo Code.

4. Testing and Verification

Report will have a Testing and Verification section, which will cover

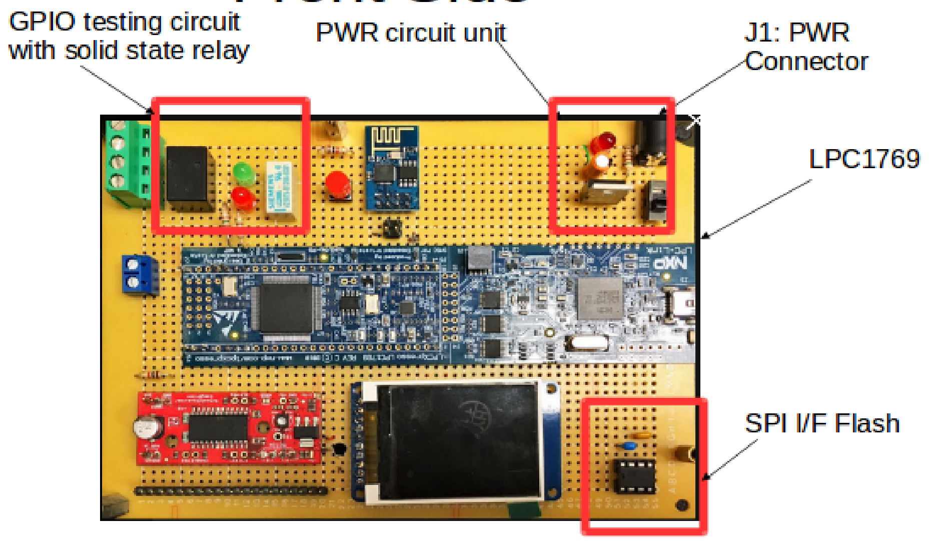
(1) Hardware testing: photos of the waveforms from oscilloscope and/or logic analyzer. Provide data from the system testing result, and/or SPICE simulation capture if needed.

(2) Software testing: Screen capture of the execution result, data from program execution.

5. Reference section with detailed technical reference and datasheet, etc.

3.3 Design Notes (Samples)

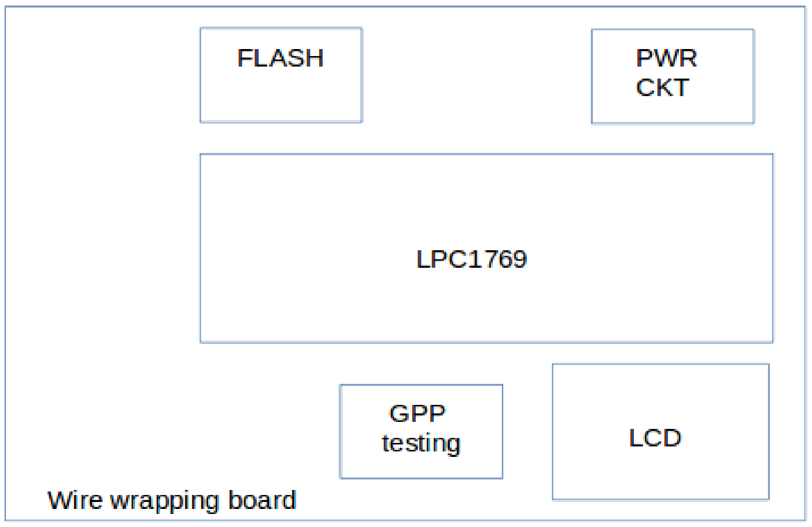
System Layout Design Front Side



Dimension: 16 x 11 mm or 6.25 x 4.50 inch

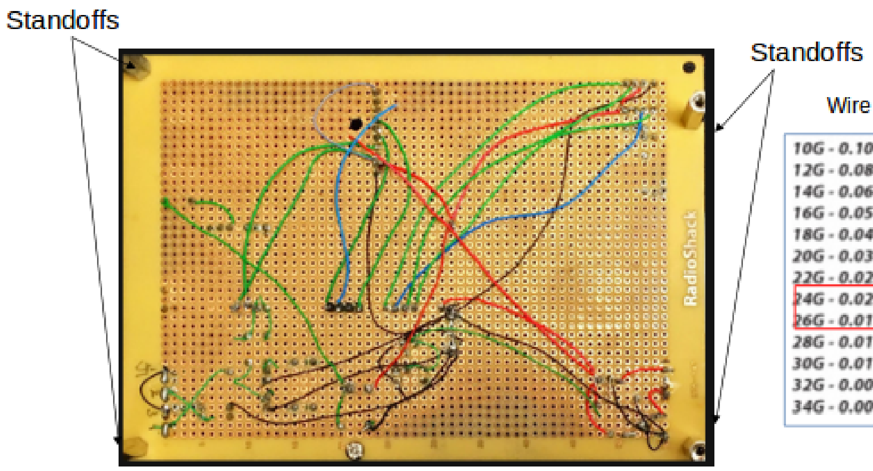
Harry Li, Ph.D.

First Thing First, Layout Design

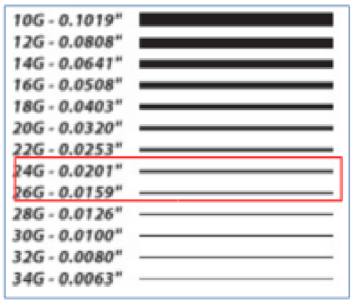


Harry Li, Ph.D.

System Layout Design Back Side



Wire Wrapping: 18g to 24G



10G - 0.1019"	
12G - 0.0808"	
14G - 0.0641"	
16G - 0.0508"	
18G - 0.0403"	
20G - 0.0320"	
22G - 0.0253"	
24G - 0.0201"	
26G - 0.0159"	
28G - 0.0126"	
30G - 0.0100"	
32G - 0.0080"	
34G - 0.0063"	

Wire soldered or wire wrapped

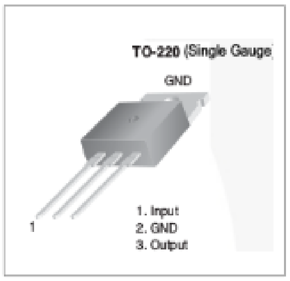
http://www.softflexcompany.com/WSWrapper.jsp?mypage=Tips_Wire_Temper.html

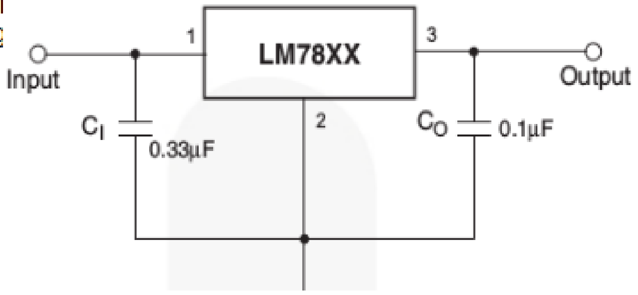
PWR Regulator LM7805

FAIRCHILD SEMICONDUCTOR®

LM78XX / LM78XXA
3-Terminal 1 A Positive Voltage Regulator

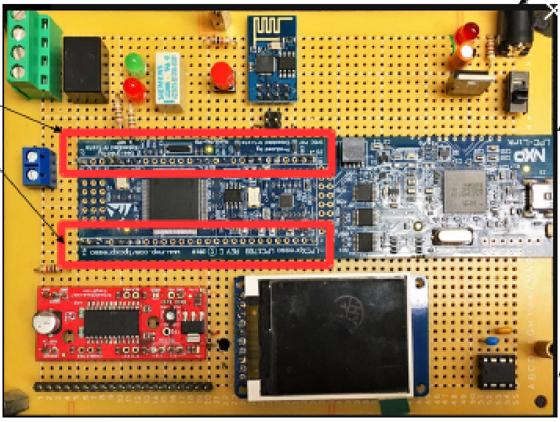
The LM78XX series of three-terminal positive regulators is available in the TO-220 package and with several fixed output voltages, making them useful in a wide range of applications. Each type employs internal current limiting, thermal shut-down, and safe operating area protection. If adequate heat sinking is provided, they can deliver over 1 A output current. Although designed primarily as fixed-voltage regulators, these devices ca
nal components for adjustable voltag





Note: 1. Voltage drop 1.5 volt;
2. current output 1 A.

J2/(or 6 for rev. B) Connector with CPU GPIO Pins



J2 (Rev. D) or J6 (Rev.B) connector with CPU GPIO Pin

J1: PWR Connector

LPC1769

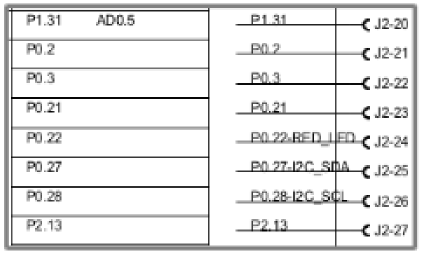
SPI I/F Flash

Dimension: 16 x 11 mm or 6.25 x 4.50 inch

Harry Li, Ph.D.

J2 (Rev. D) or J6 (Rev. B) Connector with CPU GPIO Pins

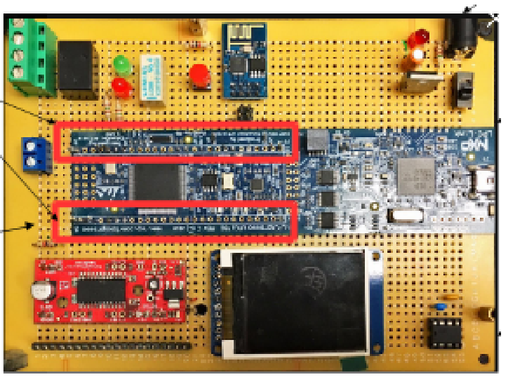
Table 1. J2 Pin Assignment



P1.31	AD0.5	P1.31	J2-20
P0.2		P0.2	J2-21
P0.3		P0.3	J2-22
P0.21		P0.21	J2-23
P0.22		P0.22-RED_LED	J2-24
P0.27		P0.27-I2C_SDA	J2-25
P0.28		P0.28-I2C_SCL	J2-26
P2.13		P2.13	J2-27

Reference: SCH design Rev D.

J2 connector with CPU GPIO Pin

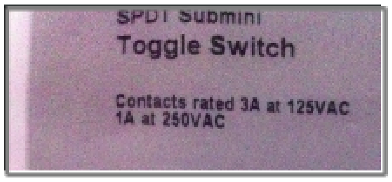


Board: 16 x 11 mm or 6.25 x 4.50 inch

SPDT switch for GPIO input testing

Table 2. J2 Connectivity

CPU	J2	Description
P0.2	J2-21	GPIO output
P0.3	J2-22	GPIO input

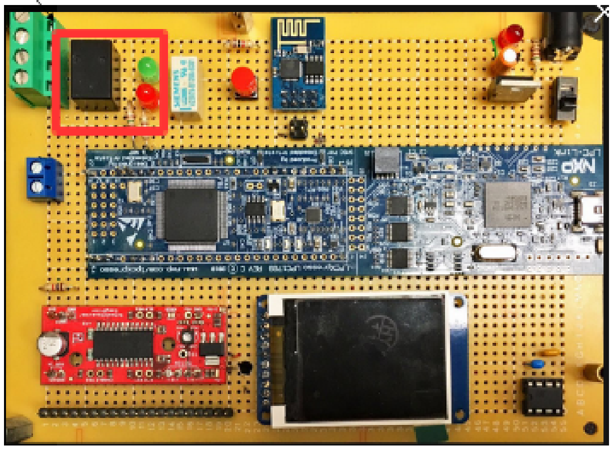


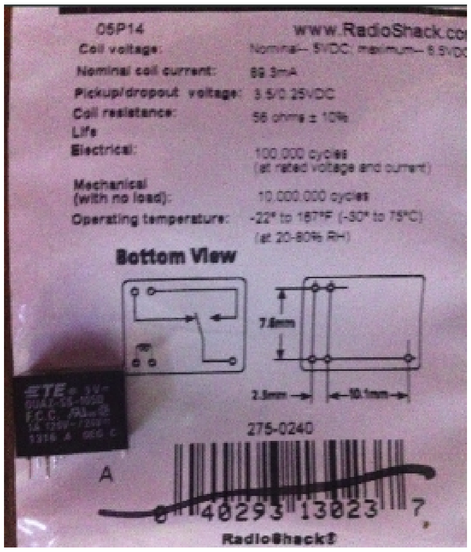
A Single Pole Double Throw (SPDT) switch is a switch that only has a single input and can connect to and switch between 2 outputs.

Harry Li, Ph.D.

GPIO (GPP) Output with SSR

SSR: Solid State Relay



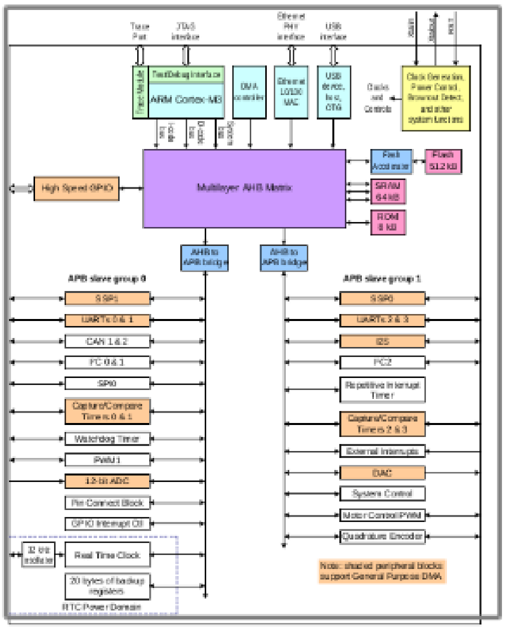

05P14
www.RadioShack.com
Coil voltage: Nominal– 5VDC; maximum– 6.5VDC
Nominal coil current: 89.3mA
Pickup/dropout voltage: 3.5/0.25VDC
Coil resistance: 56 ohms ± 10%
Life
Electrical: 100,000 cycles (at rated voltage and current)
Mechanical (with no load): 10,000,000 cycles
Operating temperature: -22° to 167°F (-30° to 75°C) (at 20-80% RH)
Bottom View
7.6mm
2.5mm
10.1mm
275-0240
RadioShack

SSR: Solid State Relay

GPIO (GPP) Controller and SPRs

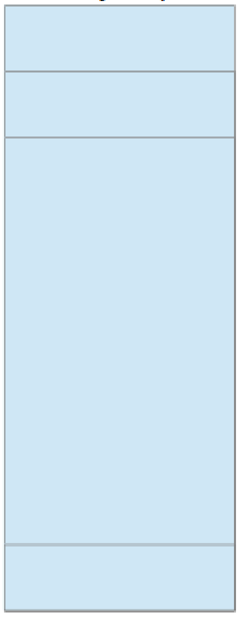
1. Definition of SPRs: Special purpose registers are those to perform init and config functions.
2. 32 bit each with unique address.
3. In IDE (software, e.g., eXpresso in this class), *.h file with something looks like the following #define SPR 0x2000_0000 map the CPU architecture to the arm gcc compiler

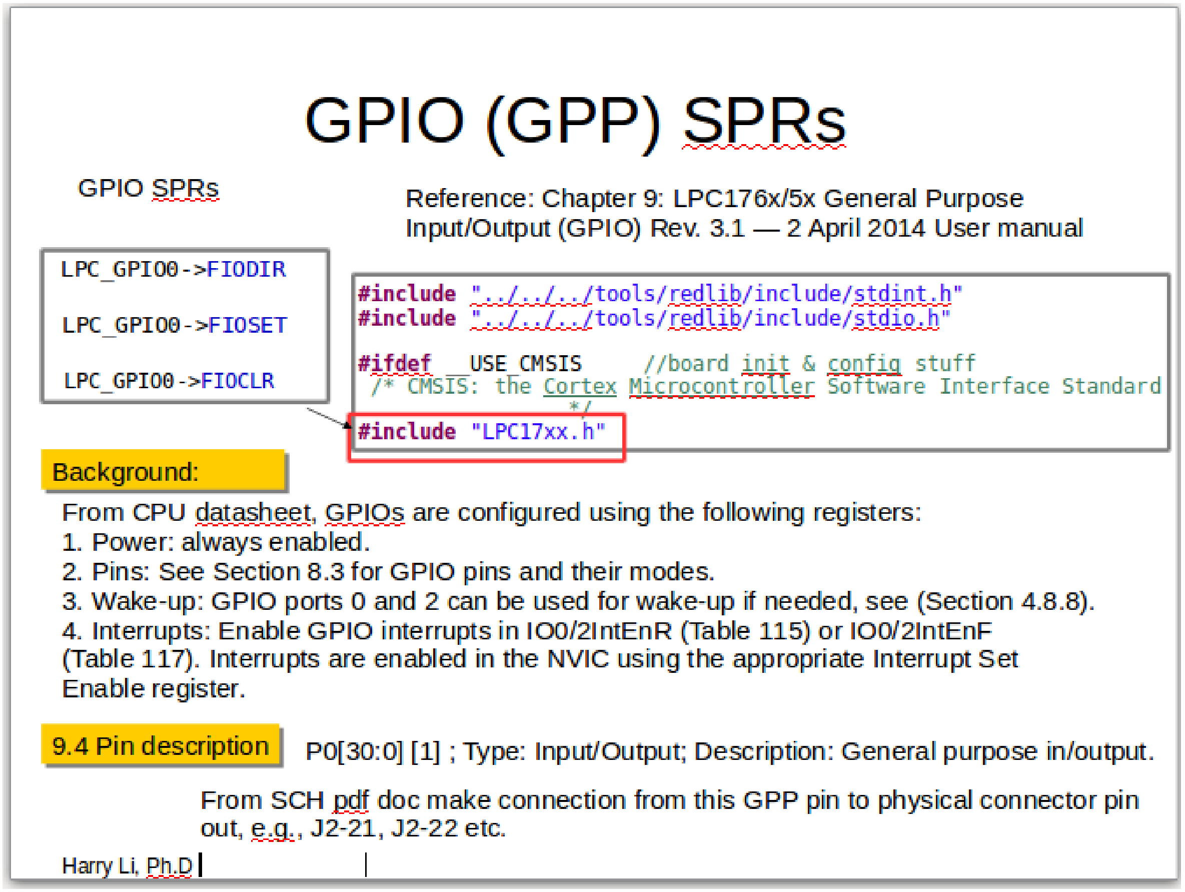
```
LPC_GPIO0->FIODIR

LPC_GPIO0->FIOSET

LPC_GPIO0->FIOCLR
```



CPU

Memory Map





GPIO (GPP) SPRs

GPIO SPRs

Reference: Chapter 9: LPC176x/5x General Purpose Input/Output (GPIO) Rev. 3.1 — 2 April 2014 User manual

```
LPC_GPIO0->FIODIR

LPC_GPIO0->FIOSET

LPC_GPIO0->FIOCLR
```

```
#include "../../../tools/redlib/include/stdint.h"
#include "../../../tools/redlib/include/stdio.h"

#ifdef __USE_CMSIS     //board init & config stuff
 /* CMSIS: the Cortex Microcontroller Software Interface Standard
                   */
#include "LPC17xx.h"
```

Background:

From CPU datasheet, GPIOs are configured using the following registers:
1. Power: always enabled.
2. Pins: See Section 8.3 for GPIO pins and their modes.
3. Wake-up: GPIO ports 0 and 2 can be used for wake-up if needed, see (Section 4.8.8).
4. Interrupts: Enable GPIO interrupts in IO0/2IntEnR (Table 115) or IO0/2IntEnF (Table 117). Interrupts are enabled in the NVIC using the appropriate Interrupt Set Enable register.

9.4 Pin description P0[30:0] [1] ; Type: Input/Output; Description: General purpose in/output.

From SCH pdf doc make connection from this GPP pin to physical connector pin out, e.g., J2-21, J2-22 etc.

Harry Li, Ph.D

(END)