Mbed Pin Input/Output Tutorial

Introduction

The mbed NXP LPC1768 development board is a streamlined tool designed for rapid prototyping. It uses the NXP LPC1768, which is an ARM Cortex-M3 based microcontroller designed by NXP Semiconductors. This document will help familiarize the reader with the process of programming this MCU in ARMv7-M Thumb assembly language. Specifically, this document will demonstrate the proper way to establish a digital input pin on the LPC1768 within the mbed environment.

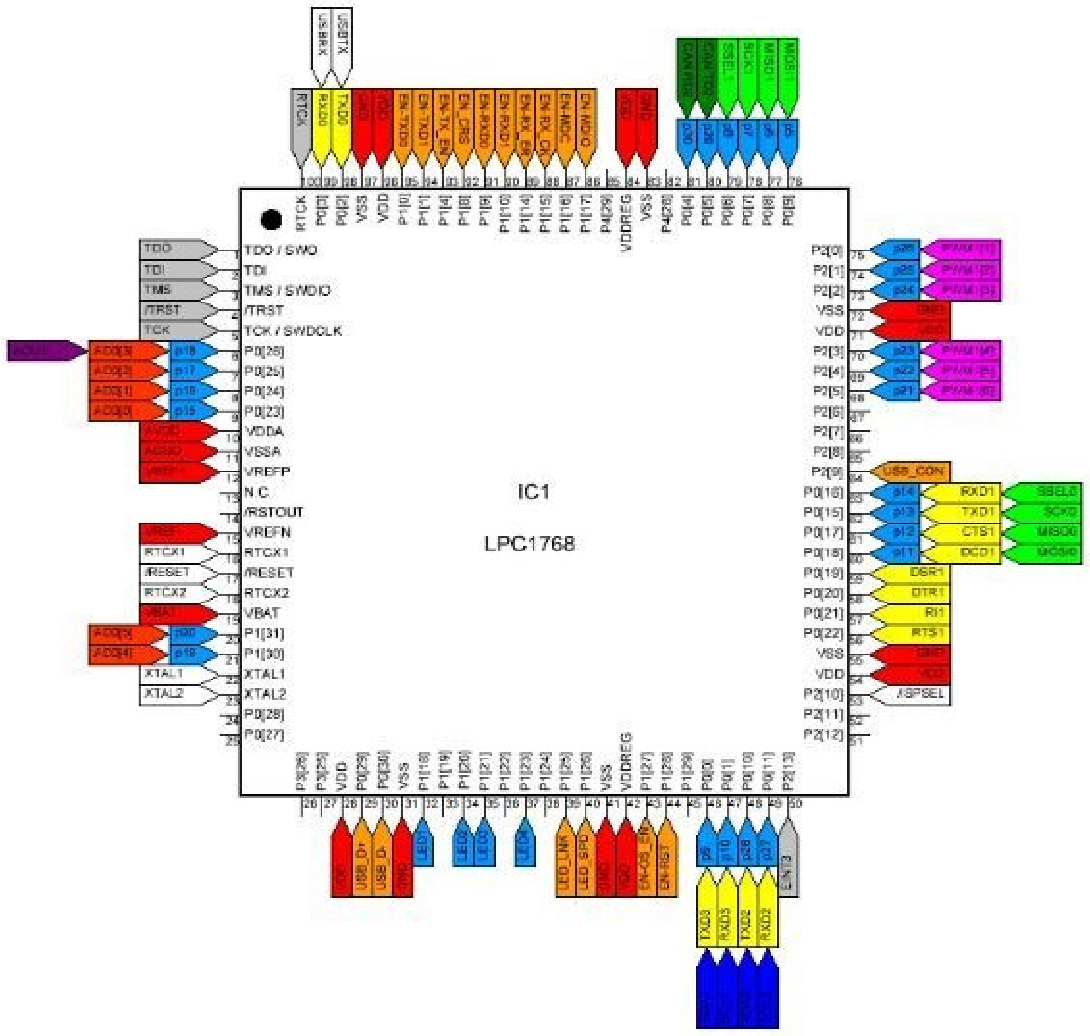


Figure 1: NXP LPC1768 Pinout

Important Sources of Information

This document references three sources heavily:

1. **NXP LPC17XX User's Manual**. In this manual, I reference only chapters 8 and 9. Chapter 8 (LPC17XX Pin Connect Block) explains the different functionalities of the pins and how to designate a given pin to have a certain functionality. Chapter 9 (LPC17XX General Purpose Input/Output) explains how to initialize and use the digital I/O features for a given pin (once you've designated said pin to have GPIO functionality according to chapter 8). This manual can be found here:
http://www.nxp.com/documents/user_manual/UM10360.pdf

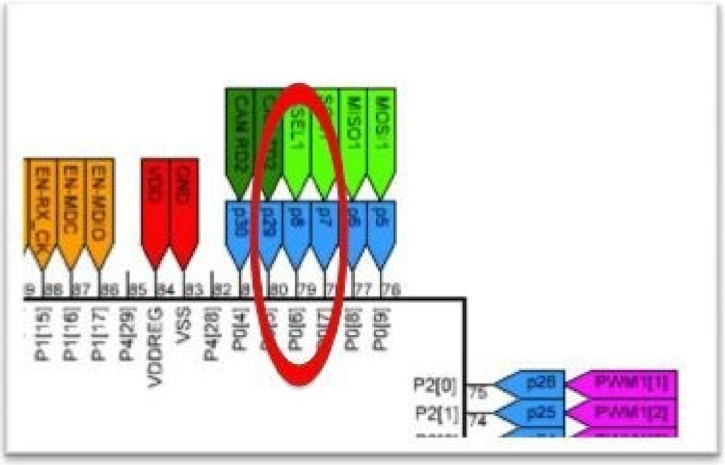
2. **LPC1768 pinout diagram**, which is labeled with the corresponding mbed pins. This document illustrates the different pins on the mbed development board, and how they connect with the ports and pins on the NXP LPC1768 (which is the internal MCU of the mbed as mentioned in the Introduction). This pinout can be found here and is shown in Figure 1:
http://mbed.org/users/synvox/notebook/lpc1768-pinout-with-labelled-mbed-pins/

3. **General mbed pinout diagram**. This pinout can be found here:
http://mbed.org/nxp/lpc2368/quick-reference/

Digital Input Explained

This section demonstrates how to establish digital input on an mbed. The mbed pinout diagram [3] indicates that pins p5-p30 (which are highlighted in blue) each have general-purpose input/output functionality. Each of these pins can be used for digital input or digital output. For this tutorial, we are going to establish digital input on mbed pin P8 using ARMv7-M Thumb assembly.

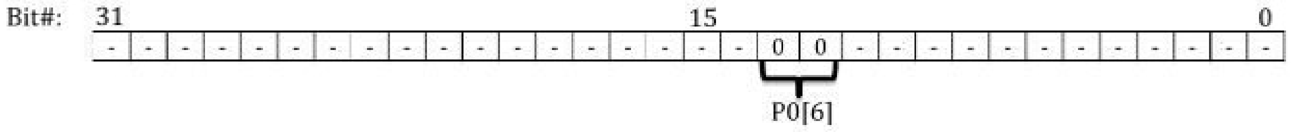
According to the LPC1768 – mbed pinout [2], mbed pin P8 corresponds to Port-0 Pin-6 of the LPC1768. One can deduce this by locating the blue p8 flag on the pinout and examining its connection to LPC1768 pin 79 via P0[6]. Please examine the figure below:



Step 1: Set the Pin's Function to General-Purpose Input/Output (GPIO)
The first step in establishing the digital input on mbed pin P8 (which corresponds to LPC1768 Port-0 Pin-6) is to set it up for general purpose input/output. According to

chapter 8.5 in the NXP LPC17XX User's Manual [1], this can be accomplished using the PIN FUNCTION SELECT register. Specifically, the pin function select register that corresponds to Port-0 Pin-6 is called PINSEL0 (see Table 74 in [1]) and is located at address 0x4002C000 (according to Table 78 in [1]). Reading further in chapter 8.5, one can see that bits 12 and 13 of PINSEL0 correspond to Port-0 Pin-6 (see [1] Table 79). In order to establish GPIO, we want to set both of these bits to 0 (Table 79 specifies that the function is GPIO when bits 13:12 = 00). The following diagram illustrates Step 1.

PINSEL0:



To accomplish this in practice, we simply need to clear bits 12 and 13 of PINSEL0 without altering the other bits. In other words,

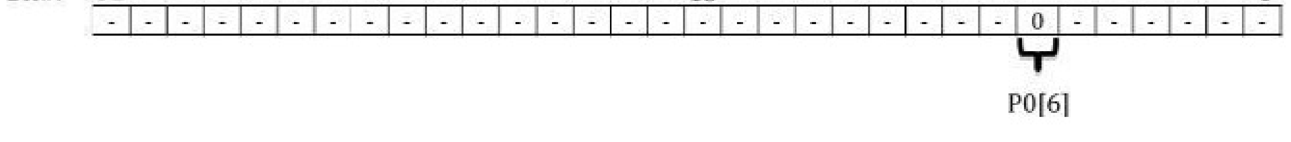
PINSEL0 = PINSEL0 & 0xFFFFCFFF

Note: The functionality of most I/O pins on the LPC1768 default to GPIO. However, a program will access and alter many different ports over the course of its runtime. Thus, it is still a good practice to physically establish this functionality manually before using digital input/output to be certain that it is GPIO.

<u>Step 2: Set the Pin's Direction to INPUT</u>

Once we have designated mbed pin P8 (which corresponds to LPC1768 Port-0 Pin-6) to have GPIO functionality, we now need to set the pin's I/O direction to input. According to chapter 9.5 in the NXP LPC17XX User's Manual [1], this can be accomplished using the Fast GPIO Port Direction Control Register. Specifically, the GPIO Port Direction Control Register that corresponds to Port-0 Pin-6 is called FIO0DIR. The 32-bit FIO0DIR register can be broken into 4 contiguous byte-addressable blocks: FIO0DIR0, FIO0DIR1, FIO0DIR2, FIO0DIR3. For this tutorial, we will consider the 32-bit FIO0DIR register in its entirety as a word-addressable entity. Thus, we can see that the FIO0DIR register is located at base memory address 0x2009C000 (according to Table 101 [1]). Reading further in chapter 9.5, one can see that bit 6 of the FIO0DIR register corresponds to Port-0 Pin-6 (Table 107 [1]). In order to establish this pin as INPUT, we want to set this bit to 0. The following diagram illustrates Step 2.

FIO0DIR:



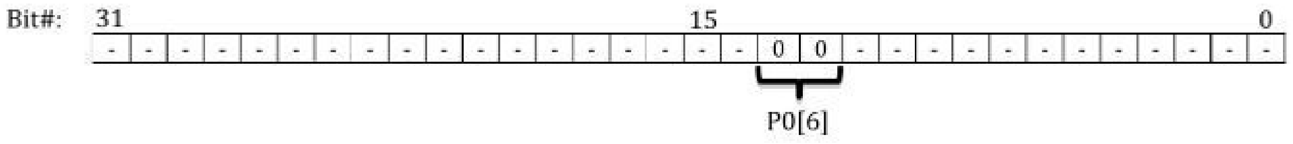
To accomplish this in practice, we simply need to clear bit 6 of FIO0DIR without altering any of the other bits. In other words,

FIO0DIR = FIO0DIR & 0xFFFFFFBF

Step 3: Pull Up/Down Logic?

When using digital input within any system, we must use a set mechanism to ensure that our input settles to an expected logic level. To satisfy this criterion, we must employ the use of pull up/down resistors with our digital inputs. Now, we can most certainly use an external pull up or pull down resistor to accomplish this. However, the LPC1768 provides us the ability to use internal pull up or pull down resistors. To demonstrate this functionality, in this step we are going to demonstrate how to use the internal pull up resistors on mbed pin P8 (which corresponds to LPC1768 Port-0 Pin-6). According to chapter 8.5 in the NXP LPC17XX User's Manual [1], this can be accomplished using the PIN MODE SELECT register. Specifically, the pin mode select register that corresponds to Port-0 Pin-6 is called PINMODE0 and is located at address 0x4002C040 (Table 78). Reading further in chapter 8.5, one can see that bits 12 and 13 of PINMODE0 correspond to Port-0 Pin-6. In order to activate the internal pull up resistor, we want to set both of these bits to 0. The following diagram illustrates Step 3.

PINMODE0:



To accomplish this in practice, we simply need to clear bits 12 and 13 of PINMODE0 without altering the other bits. In other words,

PINSEL0 = PINSEL0 & 0xFFFFCFFF

Implementation of Digital Input

Steps 1-3 above explain conceptually how to set up a digital input on a given mbed pin. Now, we are going to implement **STEP 1** using ARMv7-M Thumb assembly language.

Clearing Bits in a Register Using the BIC Instruction

In ARMv7-M assembly, the BIT CLEAR instruction has the following syntax:

BIC <dest>, <src>

For any bit that contains a 1 in the source register, the corresponding bit in the destination register will be cleared. For example, let's say that register R1 contains 0xFFFFFFFF and register R2 contains 0x00000C00. The following diagram explains the instruction BIC R1, R2 :

R1:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

R2:

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0

R1 after BIC R1, R2:

1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1	1	1	1	1	1	1	1

Implementation of Digital Input using ARMv7-M Thumb

Now that we've discovered an efficient method for clearing bits in a register, we will study a program that uses ARMv7-M Thumb assembly to set mbed pin P8 to have GPIO functionality. Recall: this implementation corresponds to **STEP 1** above. Please examine the following snippet of code:

```
1     ; LOAD MEMORY ADDRESSES AND BITMASKS
2     LDR     R0, =0x4002C000 ; LOAD ADDRESS OF PINSEL0 REGISTER
3     LDR     R3, =0x00003000 ; FORM BITMASK FOR PINSEL0 REGISTER (0x3000 = 0x11 << 12)
4     ; SET PIN FUNCTION TO GPIO (WITH SPECIFIED BITMASK IN REGISTER R3
5     LDR     R4, [R0]        ; \
6     BIC     R4, R3          ; - APPLY BITMASK FOR PINSEL0 REGISTER (clear appropriate bits)
7     STR     R4, [R0]        ; /
```

Line 2 above loads the address of PINSEL0 into a temporary register, and line 3 loads the proper bitmask to clear the appropriate bits from the PINSEL0 register according to the syntax of the BIC instruction (outlined above). Line 5 loads the current word stored in PINSEL0 using its address. Line 6 clears bits 12 and 13, essentially applying our bitmask. Finally, line 7 stores the updated word back in PINSEL0 using its address.

Note: STEPS 2 and 3 (from the Digital Input Explained section) can be implemented in a similar fashion to what is outlined here.

Reading from a Digital Input

Now that we have established a digital input on a given pin using ARMv7-M Thumb assembly, we need to determine how to read from the digital input pin. This section of the tutorial will give you insight on how to read from mbed pin P8 (which corresponds to LPC1768 Port-0 Pin-6). According to chapter 9.5 in the NXP LPC17XX User's Manual [1], this can be accomplished using the Fast GPIO Port Direction Control Register. Specifically, the GPIO Port Direction Control Register that corresponds to Port-0 Pin-6 is called FIO0PIN. The 32-bit FIO0PIN register can be broken into 4 contiguous byte-addressable blocks: FIO0PIN0, FIO0PIN1, FIO0PIN2, FIO0PIN3. For this tutorial, we will consider the 32-bit FIO0PIN register in its entirety as a word-addressable entity. Thus, we can see that the FIO0PIN register is located at base memory address 0x2009C014. Reading further in chapter 9.5, one can see that bit 6 of the FIO0PIN register corresponds to Port-0 Pin-6. In order to read the value of LPC1768 Port-0 Pin-6, we must apply a bitmask to read only the 6th bit of the FIO0PIN register.