User’s Guide

LP8758 Quick Sample Module User Guide

ABSTRACT

This user's guide presents an overview and general description of the module and provides first steps for getting started using the module.

Table of Contents

Trademarks

All trademarks are the property of their respective owners.

1 Introduction

1.1 LP8758 Module Purpose

- The LP8758 device has an effective design, is small, and has a low cost compared to other 4 buck PMICs.
- New OTP samples need up to 6 weeks for delivery which delays the ramp up time for customer applications, and needs a high volume for new OTP kick-off.
- This module can provide customers with new OTP samples for real application evaluations of prototypes in only a few hours.
- The MCU in the module can be removed after a completed evaluation on customer prototype boards for mass productions if the business size meets the volume for new OTP kick-off.
- The module can be mounted on customer boards for real applications if the business size does not meet the volume for new OTP kick-off.

1.2 Module Design Details

The module is designed as a castellated PCB module. It is composed of a PMIC LP8758-Ex (or any Ex variant) which has 4 single-phase bucks, and a simple and low cost MCU PIC16F15223. The LP8758-Ex PMIC has no output because the "EN" pin is pull-down by the MCU at power up moment. The MCU starts its built-in program to reconfigure the PMIC through its I2C port. After reconfiguration is done, the MCU raises up the "EN" and "RESET" pins to allow the PMIC to start its power on sequence; all 4 PMIC power rails will power up according to the customers' application needs which is implemented by the MCU program.

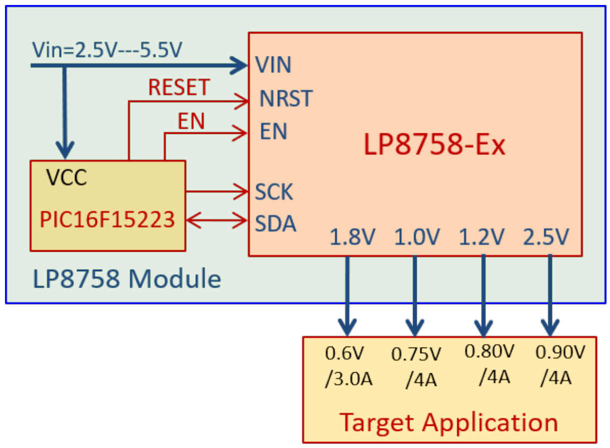

Figure 1-1. Block Diagram

Schematics of the module as shown by the figure below.

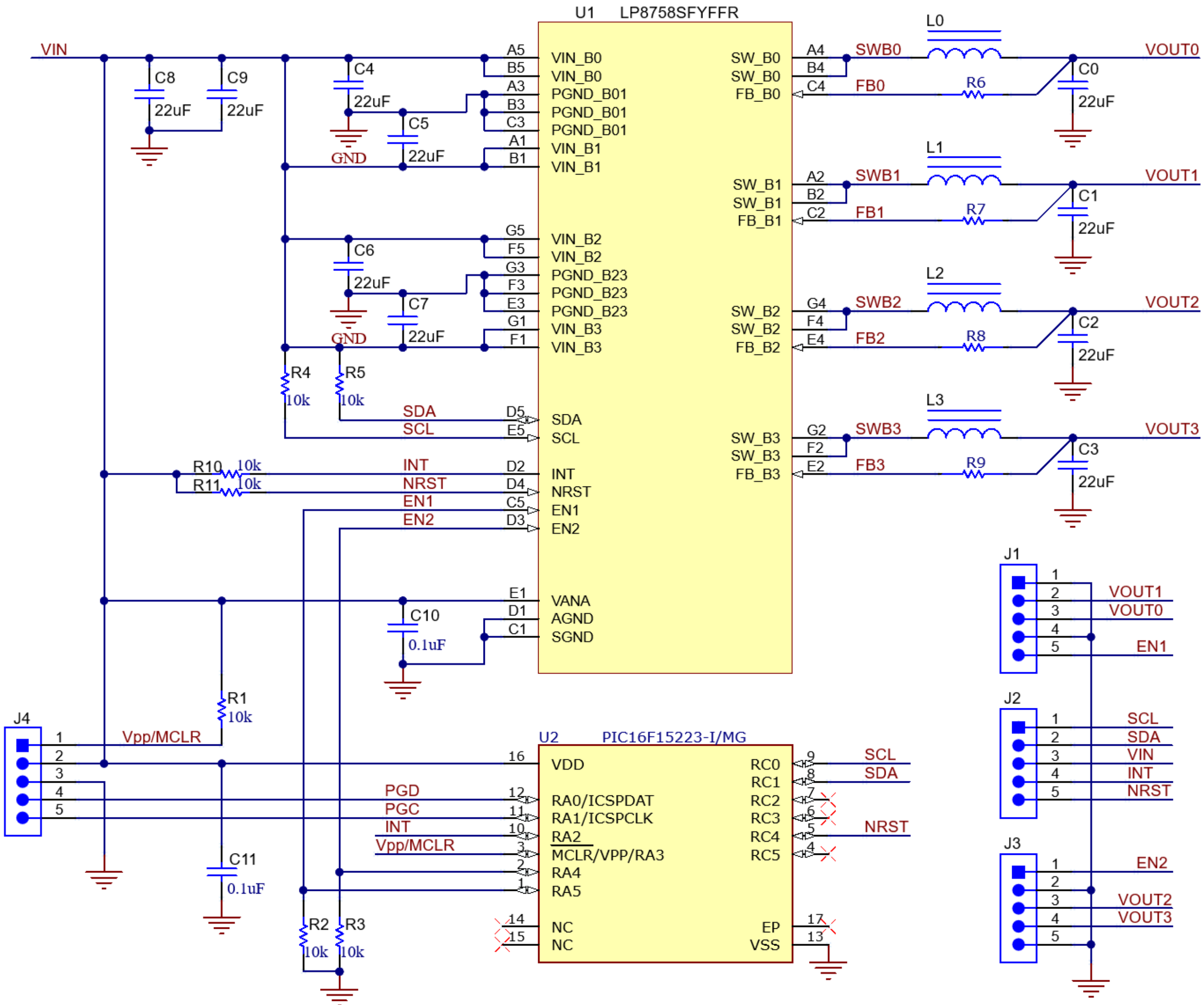

Figure 1-2. Module Schematic

4 PCB layers of the module as shown by the figure below.

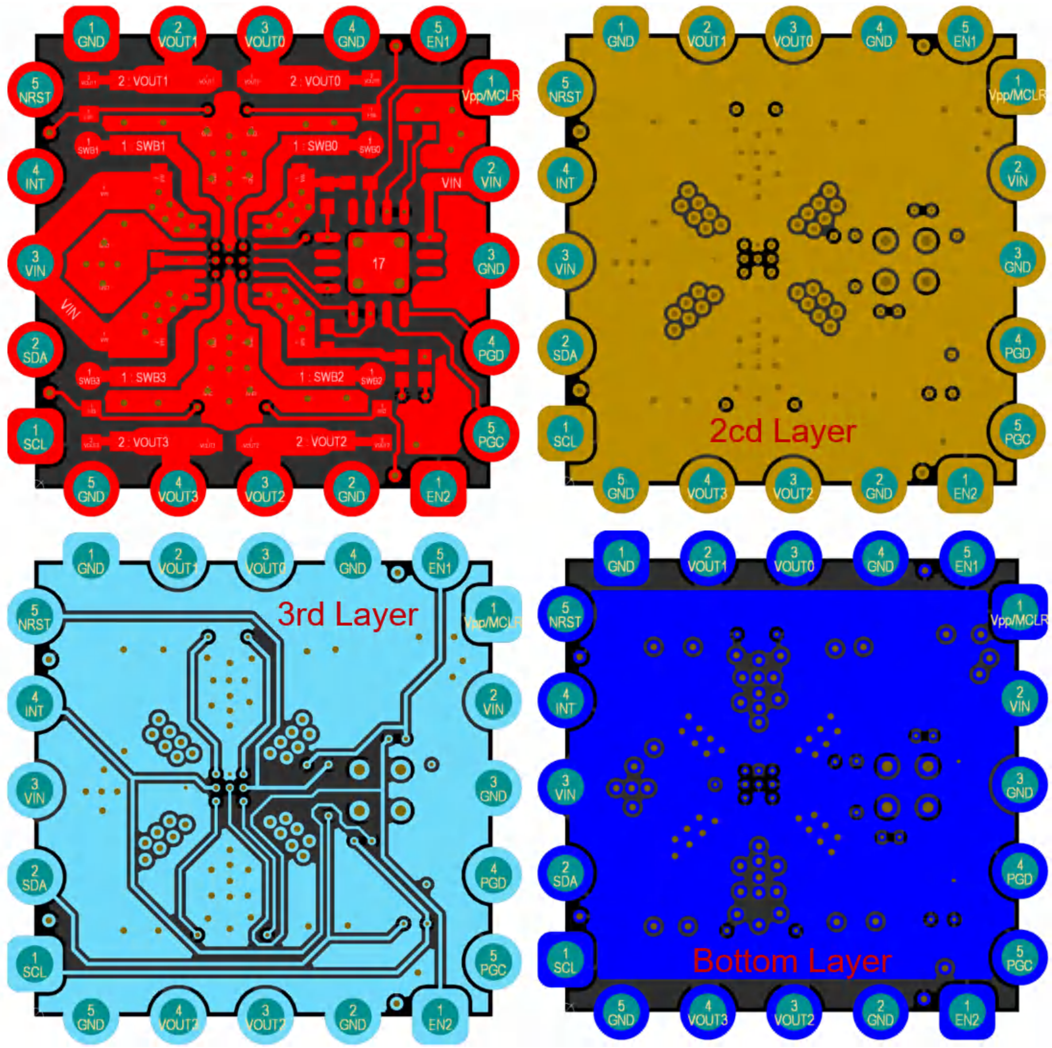

Figure 1-3. 4 PCB Module Layers

Measurement of the module as shown by the figure below.

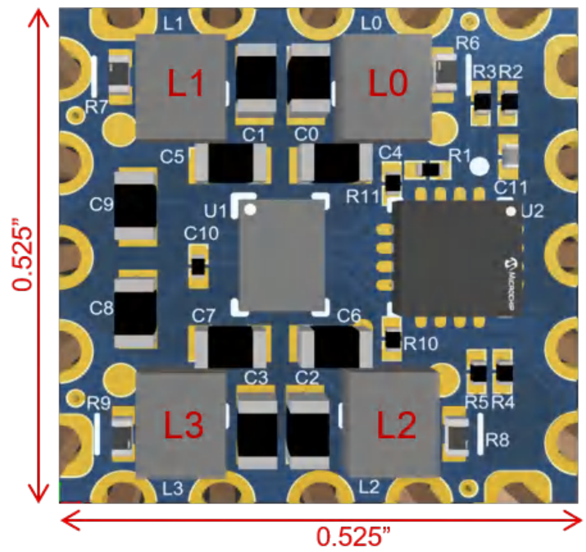

Figure 1-4. Module Measurement

1.3 Module Programming Set Up

Assembly Instructions

1. Connect the MPLAB Snap In-Circuit Debugger/Programmer to the computer using a Micro-B USB cable.
2. Connect the MPLAB Snap In-Circuit Debugger/Programmer to the LP8758 module with communication cable (soldering a 5-pin single row header if needed).

Note

Special care needs to be taken to align all pins on the debugger and the module.

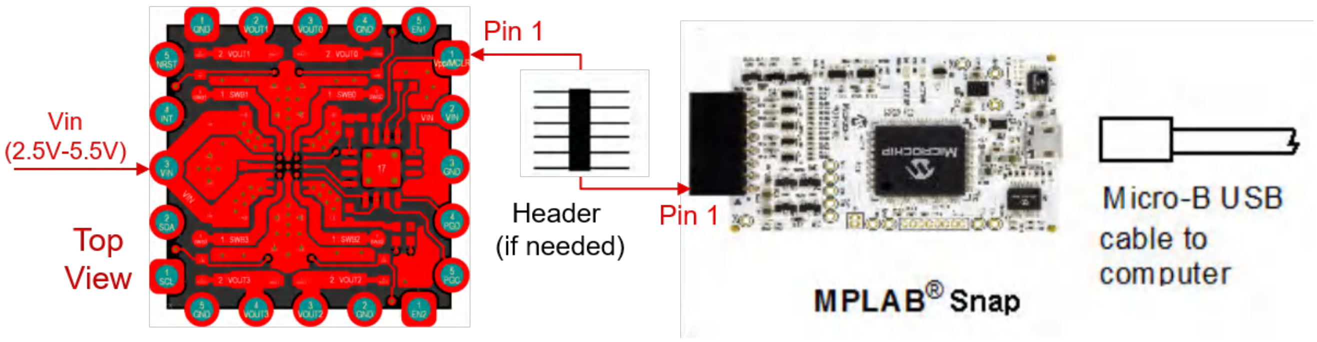

Figure 1-5. Module Programming Set Up

2 Programming the Module with Sample Code

2.1 Software Installation

1. Download the MPLAB X IDE software from http://www.microchip.com/mplabx and install onto local PC. The installer automatically loads the USB drivers and launch MPLAB X IDE.
2. Refer to the MPLAB X IDE User's Guide to install language tools, create or open a project, and configure project properties.
3. Select the "Mid-Range 8-bit PIC16F15223 MCU" which is the MCU used in th LP8758 module design.

2.2 Powering Up the Module

1. Set an external power supply output voltage between 2.5V to 5.5V; and set its output current limit at 100mA.
2. Turn off the supply power and then connect its positive output to the VIN pin and its negative output to the GND pin of the LP8758 module.
3. Turn on the supply power.

2.3 Steps to Program the Module with Sample Code

1. The sample code using the "Pic-as Compiler Toolchain" and I2C to communicate between the LP8758 module and the MCU.
2. Refer to TI provided "LP8758 Module Sample Code" and execute the code by performing "RUN Main Project"to program the module.
3. After successful programming, the output window in MPLAB X IDE will show "Program/verify complete".
4. Change registers data (circled in red as an example) as much as needed in "#define" directives for LP8758 necessary settings for real applications.
5. The reference register address/data and output voltage codes can refer to either device data sheet or TI provided "LP8758_sample_module_programming_table".

Change the data (circled in red as an example) defined in the sample code to change the buck power rail output according to application requirements.

```
;Write Data values for LP8758 module control registers
#define BUCK0_CTRL1_DATA 0xC8; /*Register data for BUCK0_CTRL1*/
#define BUCK0_CTRL2_DATA 0x3A; /*Register data for BUCK0_CTRL2*/
#define BUCK1_CTRL1_DATA 0xC8; /*Register data for BUCK1_CTRL1*/
#define BUCK1_CTRL2_DATA 0x3A; /*Register data for BUCK1_CTRL2*/
#define BUCK2_CTRL1_DATA 0xC8; /*Register data for BUCK2_CTRL1*/
#define BUCK2_CTRL2_DATA 0x3A; /*Register data for BUCK2_CTRL2*/
#define BUCK3_CTRL1_DATA 0xC8; /*Register data for BUCK3_CTRL1*/
#define BUCK3_CTRL2_DATA 0x3A; /*Register data for BUCK3_CTRL2*/
#define BUCK0_VOUT_DATA 0x25; /*Register data for BUCK0_VOUT*/ Vout = 0.8V/
#define BUCK0_FLOOR_VOUT_DATA 0x0; /*Register data for BUCK0_FLOOR_VOUT*/
#define BUCK1_VOUT_DATA 0x25; /*Register data for BUCK1_VOUT*/ Vout = 0.8V/
#define BUCK1_FLOOR_VOUT_DATA 0x0; /*Register data for BUCK1_FLOOR_VOUT*/
#define BUCK2_VOUT_DATA 0x25; /*Register data for BUCK2_VOUT*/ Vout = 0.8V/
#define BUCK2_FLOOR_VOUT_DATA 0x0; /*Register data for BUCK2_FLOOR_VOUT*/
#define BUCK3_VOUT_DATA 0x25; /*Register data for BUCK3_VOUT*/ Vout = 0.8V/
#define BUCK3_FLOOR_VOUT_DATA 0x0; /*Register data for BUCK3_FLOOR_VOUT*/
```

Figure 2-1. Data Definition in Sample Code-1

Change the data (circled in red as an example) defined in sample code to make the buck power rail output according to application requirements.

```
#define BUCK0_DELAY_DATA 0x22; /*Register data for BUCK0_DELAY*/
#define BUCK1_DELAY_DATA 0x22; /*Register data for BUCK1_DELAY*/
#define BUCK2_DELAY_DATA 0x22; /*Register data for BUCK2_DELAY*/
#define BUCK3_DELAY_DATA 0x22; /*Register data for BUCK3_DELAY*/
#define RESET_DATA 0x0; /*Register data for RESET*/
#define CONFIG_DATA 0x6; /*Register data for CONFIG*/
#define INT_TOP_DATA 0x0; /*Register data for INT_TOP*/
#define INT_BUCK_0_1_DATA 0x0; /*Register data for INT_BUCK_0_1*/
#define INT_BUCK_2_3_DATA 0x0; /*Register data for INT_BUCK_2_3*/
#define TOP_MASK_DATA 0x0; /*Register data for TOP_MASK*/
#define BUCK_0_1_MASK_DATA 0x0; /*Register data for BUCK_0_1_MASK*/
#define BUCK_2_3_MASK_DATA 0x0; /*Register data for BUCK_2_3_MASK*/
#define SEL_I_LOAD_DATA 0x0; /*Register data for SEL_I_LOAD*/
#define I_LOAD_2_DATA 0x0; /*Register data for I_LOAD_2*/
#define I_LOAD_1_DATA 0x0; /*Register data for I_LOAD_1*/
```

Figure 2-2. Data Definition in Sample Code-2

Mailing Address: Texas Instruments, Post Office Box 655303, Dallas, Texas 75265