TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

TEXAS INSTRUMENTS

Module 6

Introduction: General Purpose Input Output

Introduction: General Purpose Input Output

Educational Objectives:

REVIEW C programming
UNDERSTAND direction registers, input, and output
EXPLORE conversion from light to voltage, and voltage to binary
LEARN how to write software to initialize GPIO pins
DESIGN, BUILD & TEST A SYSTEM
Detect position relative to a black line on a white field

Prerequisites (Modules 1, 2, and 4)
- Running code on the LaunchPad using CCS (Module 1)
- Voltage, current, resistance, capacitance (Module 2)
- Basic C programming (Module 4)

Recommended reading materials for students:
- Chapter 6, **Embedded Systems: Introduction to Robotics**, Jonathan W. Valvano, ISBN: 9781074544300, copyright © 2019

Figure 1. QTRX-8 line sensor top view (Courtesy pololu.com)

The simplest I/O port on a microcontroller is the parallel port, or **general purpose input output** (GPIO). A parallel I/O port is a mechanism that allows the software to interact with external devices. It is called parallel because multiple signals can be accessed all at once. Ports 1 – 10 are 8 bits wide meaning we read and write port pins 8 bits at time. However, not every port on the MSP432 LaunchPad has all 8 pins.

An **input port** allows the software to read external digital signals. That means a read cycle access from **P1->IN** returns the values existing on the inputs of Port 1 at that time. To make a pin input, we write a 0 to its direction register. A write cycle access to an input port usually produces no effect. Input pins on some microcontrollers are 5V-tolerant, meaning input voltages can vary from 0 to 5.0 V. However, pins on the MSP432 are not 5-V tolerant, meaning the input voltages must be between 0 and 3.3 V.

While an input device usually just involves the software reading the port, an **output port** can participate in both the read and write cycles very much like a regular memory. A write cycle to **P1->OUT** will affect the values on the output pins of Port 1. To make a pin output, we write a 1 to its direction register. Since it is a readable output, a read cycle access from the port address returns the current values existing on the port pins. We can either read from **P1->OUT**, returning the values previously written, or read from the pins themselves to see the pin values, **P1->IN**.

To make the microcontroller more marketable, the ports on most microcontrollers can be software-specified to be either inputs or outputs. Microcontrollers use the concept of a **direction register** to determine whether a pin is an input (direction register bit is 0) or an output (direction register bit is 1). We define an initialization ritual as a program executed once during start up that initializes hardware and software. If the ritual makes the direction bit zero, the port pin behaves like a simple input, and if it makes the direction bit one, the port pin becomes a readable output. Each digital port pin has its own direction bit. This means some pins on a port may be inputs while others are outputs.

In the lab associated with this module, you will interface a line sensor to the microcontroller, see Figure 1. Proper function of the sensor will require you to fully understand the direction register and how to perform input and output. This lab does provide an opportunity to improve your C programming skills including debugging with CCS and with an oscilloscope. Since there are measurements in this lab, you will be able to discover performance metrics such as accuracy, monotonicity, specificity, standard deviation (noise), and coefficient of variation. In previous modules, you developed code on the MSP432 using CCS, but in this module you will create a major component required to build the robot explorer. Other labs will provide additional sensors for the robot controller. In **10. Debugging** you will add bumper switches, and implement this line sensor interface using interrupts.

The basic approach to system development is to create components and then piece the components together to create the system. In this module, you will design develop and test the line sensor measurement required for the for robot explorer.

ti.com/rslk

IMPORTANT NOTICE AND DISCLAIMER

TI PROVIDES TECHNICAL AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS AND IMPLIED, INCLUDING WITHOUT LIMITATION ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT OF THIRD PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for skilled developers designing with TI products. You are solely responsible for (1) selecting the appropriate TI products for your application, (2) designing, validating and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. TI grants you permission to use these resources only for development of an application that uses the TI products described in the resource. Other reproduction and display of these resources is prohibited. No license is granted to any other TI intellectual property right or to any third party intellectual property right. TI disclaims responsibility for, and you will fully indemnify TI and its representatives against, any claims, damages, costs, losses, and liabilities arising out of your use of these resources.

TI’s products are provided subject to TI’s Terms of Sale (www.ti.com/legal/termsofsale.html) or other applicable terms available either on ti.com or provided in conjunction with such TI products. TI’s provision of these resources does not expand or otherwise alter TI’s applicable warranties or warranty disclaimers for TI products.

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