TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

TEXAS INSTRUMENTS

Module 10

Introduction: Debugging Real-time Systems

Introduction: Debugging Real-time Systems

Educational Objectives:

REVIEW C programming arrays
UNDERSTAND how flash memory operates
EXPLORE debugging techniques for real-time systems
LEARN how to generate periodic interrupts using SysTick
INTERFACE bump switches to the robot
DESIGN, BUILD & TEST A SYSTEM
Stores input data into a black-box recorder

Prerequisites (Modules 6, 8, and 9)
- GPIO digital inputs (Module 6)
- Switches and LEDs (Module 8)
- SysTick timer (Module 9)

Recommended reading materials for students:
- Chapter 10, **Embedded Systems: Introduction to Robotics,** Jonathan W. Valvano, ISBN: 9781074544300, copyright © 2019

System verification is an important task when developing embedded systems, especially if the system is to be deployed in safety critical situations. Furthermore, in a **real-time system**, it is not only important to get the correct answer, it is important to get the correct answer at the correct time. **Latency** is the time between when a service is requested and the time when service is initiated. Similarly, **response time** is the time between when a service is requested and the time when service is complete. A real-time system is one that can guarantee a worst-case latency. Alternatively, we can categorize a system as real time if there is an upper bound on the response time.

Some requests occur periodically, and in this module we will use **SysTick interrupts** to execute tasks on a regular basis.

The second component to this module is to develop debugging techniques for real-time systems. **Intrusiveness** is defined as the degree to which the debugging code itself alters the performance of the system being tested. Breakpoints, single stepping, and printf output are high intrusive, and thus not appropriate for debugging real-time systems. Rather we will learn how to dump strategic information into arrays, providing similar observations as the classical printf statement, but in a minimally intrusive manner. For logging, debugging data for long periods of time, we can dump data into the flash ROM of the microcontroller.

In the lab associated with this module, you will interface bump sensors with the microcontroller, see Figure 1. These switches will allow you to know if and where the robot has contacted an obstacle. Data from the line sensor and bump sensors will be collected periodically using SysTick interrupts. Using interrupts to handle the line sensor provides a processor-efficient solution.

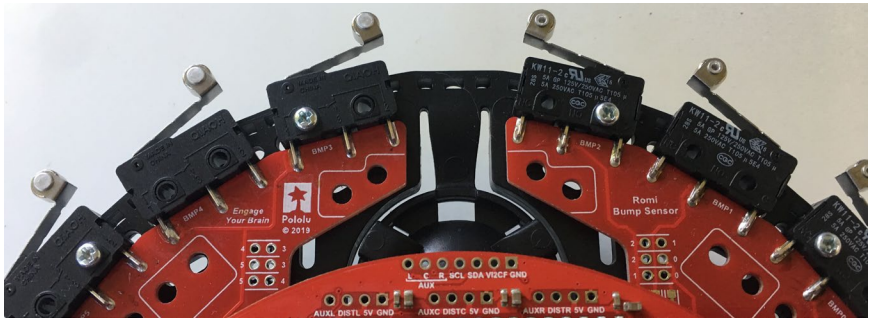

Figure 1. Bump sensors, positioned at the front of the robot.

The basic approach to a system requiring multiple software tasks is to deploy multithreading. A **thread** is defined as the action caused by executing software. From an etymological point of view, consider the thread as the bond that stitches instructions together as software executes. One software thread is the traditional main program, which runs most of the time. Interrupts will be used to create additional threads. An **interrupt** is a hardware-triggered software execution. In this module, the SysTick interrupt will execute a software task periodically. In Module 13, we will use timers to create PWM outputs. In Module 14, we will use edge-triggered interrupts so a software task is executed immediately if any of the bump sensors are activated.

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