TI-RSLK MAX

Texas Instruments Robotics System Learning Kit

Texas Instruments

Module 7

Quiz: Finite State Machines

Quiz: Finite State Machines

Q1 Pointers and Strings

Write a function that compares two null-terminated strings. Pointers to two strings are passed by reference into the function. The return parameter will be 0 if the strings do not match, and will be nonzero if the strings match. The prototype is

```
int StringCompare(char *pt1, char *pt2);
```

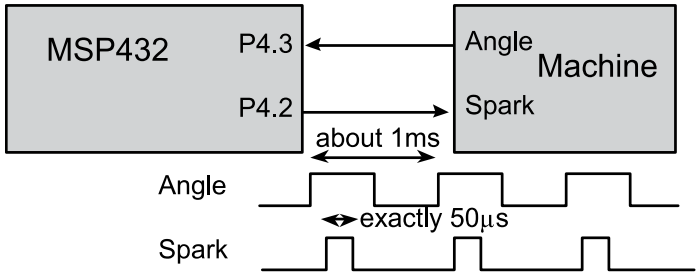
Q2 Arrays and Indexing

Write a function that calculates the dot-product of two arrays. Each array has ten 32-bit signed integers. The prototype is

```
int32_t DotProduct(int32_t buf1[10], int32_t buf2[10]);
```

Q3 Finite State Machine

Design a microcomputer-based controller using a linked list finite state machine. The system has one input and one output.



The input, **Angle**, is a periodic signal with a frequency of 20 to 1000 Hz (has a period of 1 to 50 ms). The output, **Spark**, should be a positive pulse (exactly 50 µs wide) every time **Angle** goes from 0 to 1. The delay between the rising edge of **Angle** and the start of the **Spark** pulse should be as short as possible. The period of **Angle** can vary from 1 ms to 50 ms. Since **Angle** is an input you cannot control it, only respond to its rising edge.

a) Design the one input, one output finite state machine for this system. Draw the state transition graph. Use descriptive state names.
b) Show the C code to create the statically-allocated linked list. Include statement(s) to place it in ROM on your microcomputer.
c) Show the C language controller. The controller can have no conditional statements (no **`if`**, no **`switch`**, no conditional operator). Assume this is the only task that the microcomputer executes. I.e., show ALL the instructions necessary. Make the program automatically start on a RESET. You may call functions in **Clock.c**.

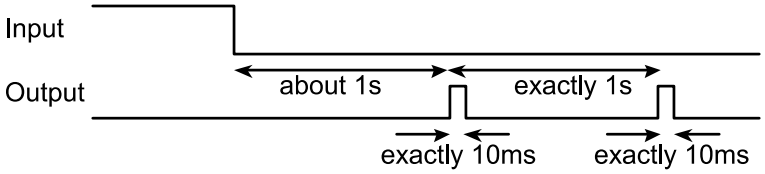
Q4 Finite State Machine

Design a FSM with two inputs and two outputs. The inputs are on Port 4 bits 0,1 and the outputs are on Port 5 bits 0,1. Initially the output is 00. You may also assume both inputs are initially 0. The machine waits for the first input to become 1. If P4.1 goes high first, set the output to 10. If P4.0 goes high first, the output goes to 01. If both inputs go high at the same time, the output goes to 11.

a) Design the two input, two output finite state machine for this system. Draw the state transition graph. Use descriptive state names
b) Show the C code to create the statically-allocated linked list. Implement next states as pointers, Include statement(s) to place it in ROM on your microcomputer.
c) Show the C language controller. The controller can have no conditional statements (no **`if`**, no **`switch`**, no conditional operator ?) Assume this is the only task that the microcomputer executes. I.e., show ALL the instructions necessary. Make the program automatically start on a RESET. You may call functions in **Clock.c**.

Q5 Finite State Machine

You will design a pacemaker using a Moore FSM. There is one input and one output. The input will be high if the heart is beating on its own. The input will be low if the heart is not beating on its own. If the heart is not beating your machine should pace the heart. If the heart is beating on its own, the input will be high and your output should be low. However, if the input is low, you should pace the heart by giving a 10 ms output pulse every 1000 ms. P4.0 is output, P4.1 is input.



a) Design the one input, one output finite state machine for this system. Draw the state transition graph. Use descriptive state names
b) Show the C code to create the statically-allocated linked list. Implement next states as indices. Include statement(s) to place it in ROM on your microcomputer.
c) Show the C language controller. The controller can have no conditional statements (no **`if`**, no **`switch`**, no conditional operator). Assume this is the only task that the microcomputer executes. I.e., show ALL the instructions necessary. Make the program automatically start on a RESET. You may call functions in **Clock.c**.

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