

# TI-RSLK **MAX**

Texas Instruments Robotics System Learning Kit



# Module 4

Activity: Software Design using MSP432



# Activity: Software Design using MSP432

## Question 1

Write a C function that returns true if an ASCII character is a letter, and false otherwise. The letters exist from 0x41 to 0x5A and from 0x61 to 0x7A inclusive. The prototype for this function is

```
int bLetter(char data);
```

## Question 2

Write a C function to calculate the average of three numbers. Assume the three numbers are passed by value into your function. The prototype for this function is

```
int32_t Average(int32_t n1, int32_t n2, int32_t n3);
```

## Question 3

Write a C function to find the maximum of three numbers. Assume the three numbers are passed by value into your function. The prototype for this function is

```
int32_t Max(int32_t n1, int32_t n2, int32_t n3);
```

## Question 4

Write a C function to calculate the quadratic equation

$$y = 2x^2 - 3x + 1$$

assuming  $x$  and  $y$  are 32-bit numbers. Some values of  $x$  will cause the calculation of  $y$  to extend beyond the values allowed by 32-bit signed numbers. Determine the largest possible value for  $x$ , such that  $y < 2^{31}$ . Use this threshold to return  $y = 0x7FFFFFFF$  ( $2^{31}-1$ ) if the input value would create overflow. Determine the smallest possible value for  $x$ , such that  $y > -2^{31}$ . Use this threshold to return  $y = 0x80000000$  ( $-2^{31}$ ) if the input value would create underflow. The prototype for this function is

```
int32_t Quadratic(int32_t x);
```

## Question 5

Write a C function that calculates the square distance between two points  $(x_1, y_1)$  and  $(x_2, y_2)$

$$d = (x_1 - x_2)^2 + (y_1 - y_2)^2$$

assuming  $x_1$ ,  $x_2$ ,  $y_1$ , and  $y_2$  are signed 32-bit numbers. You may assume the numbers are small enough that overflow does not occur. The prototype for this function is

```
int32_t SquareDistance(int32_t x1, int32_t y1,  
int32_t x2, int32_t y2);
```

## Question 6

Write a C function that returns true if  $10 \leq x < 99$ , and false otherwise. The prototype for this function is

```
int bTwoDigit(uint32_t x);
```

## Question 7

Unsigned 32-bit numbers range from 0 to  $2^{32}-1$  (4294967295). Write a C function that takes an unsigned 32-bit number and returns a result from 0 to 10 defining the number of decimal digits required to represent the number. For example, the input of 0 returns 0, the input of 1 – 9 returns 1, the input of 10 – 99 returns 2, etc. The prototype for this function is

```
uint32_t NumDigits(uint32_t x);
```

## Question 8

Write a C function that multiplies two unsigned 32-bit numbers. Implement overflow detection such that if the product were to exceed  $2^{32}-1$ , the function returns  $0xFFFFFFFF$  ( $2^{32}-1$ ). The prototype for this function is

```
uint32_t Product(uint32_t n1, uint32_t n2);
```

**[ti.com/rslk](https://ti.com/rslk)**



# Module 4

Quiz: Software Design using MSP432



# Quiz: Software Design using MSP432

## Q1 Conditional

Write a C function to find the minimum of three numbers. Assume the three numbers are passed by value into your function. The prototype is  
`int16_t Min(int16_t n1, int16_t n2, int16_t n3);`

## Q2 Conditional

Write a C function to returns a true if an ASCII character is a hex digit. Hex digits are 0x30 to 0x39 and 0x41 to 0x46 inclusive. The prototype is  
`int isHex(char data);`

## Q3 Conditional

Write a C function to returns the absolute value of a number. The input is signed, but the output will be unsigned. The prototype is  
`uint32_t Abs(int32_t data);`

## Q4 Calculations

Write a C function to calculate the equation

$$y = 1000/x - (3*x+1)/4$$

assuming x and y are 32-bit numbers. Return y = 0x7FFFFFFF ( $2^{31}-1$ ) if the input value is zero, otherwise you can ignore overflow. The prototype for this function is  
`int32_t Calculate(int32_t x);`

## Q5 Calculations

Assume x1, x2, x3, x4 are four measurements collected at 1ms time intervals. Calculate the discrete derivative using this equation

$$d = x1+3*x2-3*x3-x4$$

If the units of x1 is mV, then the units of d will be mV/ms (or V/s). Assume the inputs are 16-bit signed numbers ranging from 0 to 3300. Solve overflow by limiting the output to -1000 to +1000 V/s. Hint, calculate the intermediate result in 32-bit math, check for overflow, and then return a 16-bit result. The prototype for this function is  
`int16_t Derivative(int16_t x1, int16_t x2, int16_t x3, int16_t x4);`

## 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](http://www.ti.com/legal/termsofsale.html)) or other applicable terms available either on [ti.com](http://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  
Copyright © 2019, Texas Instruments Incorporated