Circuit Cellar / Texas Instruments DesignStellaris 2010 Design Contest

Submission Deadline: June 23rd, 2010

Abstract

Project Number: **TI2845**

Luminary Device: **LM3S9B96**

Project Name:

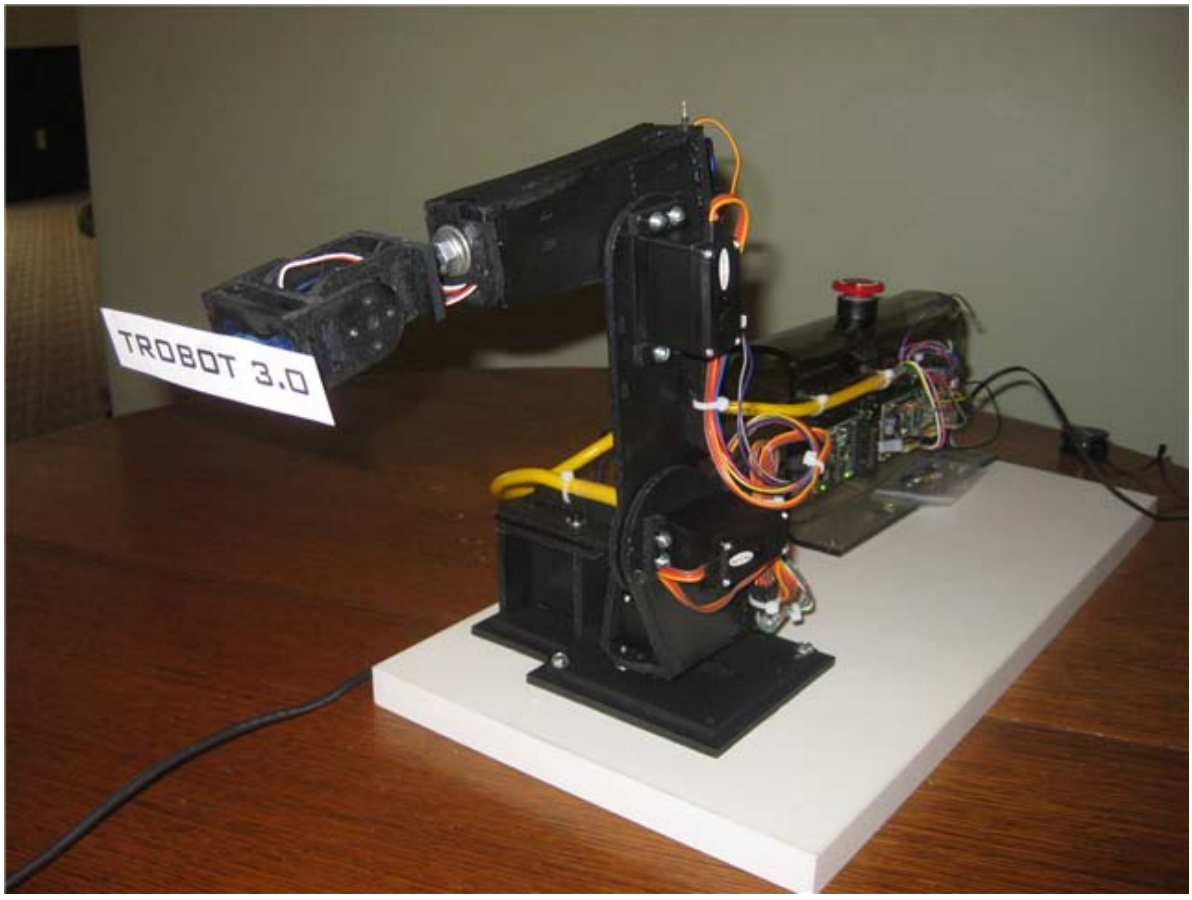
TROBOT 3.0: An advanced ABB Robot Studio interface for a miniature articulated robot.

The Project:

The TROBOT 3.0 is a miniature 6-axis robot powered by small (RC style) servo motors. TROBOT utilizes a Texas Instruments EKK-LM3S9B96 Evaluation kit, which acts as servo controller interface between the robot and a PC running ABB's Robot Studio.

ABB's Robot Studio (www.robotstudio.com) is a powerful software tool used for simulation and offline programming of the full line of ABB Industrial robots. The 3D virtual environment makes it possible to setup, simulate, and program complex robot cells in a virtual world. ABB's Virtual Robot Controller technology can simulate almost all of the features of their robot programming language known as RAPID on an IRC5 robot controller.

This project adapts the Robot Studio application to control a custom made robot called simply the "TROBOT". This project is actually the third iteration of the design and is the most advanced version yet. The powerful LM3S9B96 processor allows for new features and capabilities that couldn't be achieved with previous versions.



Sample Program Code:

The TROBOT application was written by starting out with the example "qs-safertos" application provided with the EKK-LM3S9B96 evaluation kit. This sample code allowed me to get the basic communication portion talking to the Robot Studio. The lwIP

module provided all of the resources needed to establish basic socket communication and pass data over Ethernet from the computer running Robot Studio to the LM3S9B96.

Modifying the SafeRTOS allowed me to easily create my own additional PWM task which handles configuring and updating the onboard PWM's. The LED task was left intact but is now used as a basic "heartbeat" signal indicator to let me know the board is on and functioning.

The PWM task continuously runs and updates the servo values. The velocity that the servos move is handed in a group of routines that compare the current servo position with the target position and then adjust the PWM values according to the rate value.

The EKK-LM3S9B96 TROBOT application is setup to receive a socket message containing a string of data. Formatted as follows:

7500:7500:7500:7500:7500:7500;

The message contains six colon (:) delimited 16-bit numbers followed by a semicolon (;). These numbers represent the values for the PWM outputs for each servo.

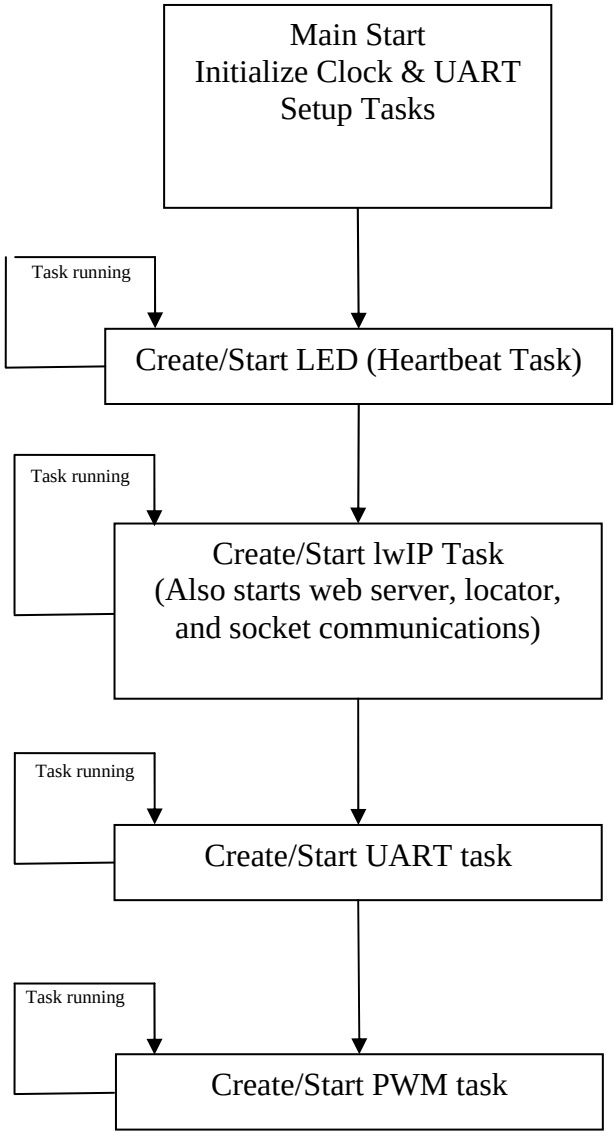
The following example code was used to extract the numbers from the string data and assign them to the variables used to set the PWM outputs for each servo motor.

```
  /* extract first string from string sequence */
  str1 = strtok(str, ":");
printf("%i: %s\n", x, str1);
  x++;
str2 = strtok(NULL, ":");
  printf("%i: %s\n", x, str2);
  x++;
str3 = strtok(NULL, ":");
printf("%i: %s\n", x, str3);
  x++;
str4 = strtok(NULL, ":");
printf("%i: %s\n", x, str4);
  x++;
str5 = strtok(NULL, ":");
printf("%i: %s\n", x, str5);
  x++;
str6 = strtok(NULL, ";");
printf("%i: %s\n", x, str6);
  /*  Set variables equal to the token values  */
  axis1 = atoi(str1);
  axis2 = atoi(str2);
  axis3 = atoi(str3);
  axis4 = atoi(str4);
  axis5 = atoi(str5);
  axis6 = atoi(str6);
```

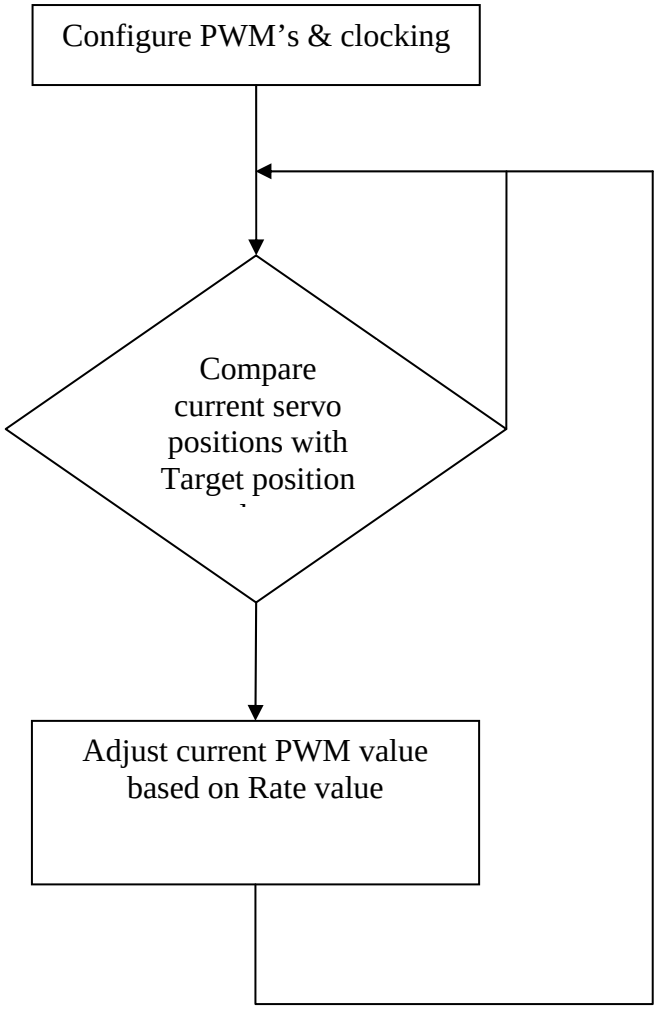
As the LM3S9B96 SafeRTOS runs the data is received over Ethernet and the PWM values are controlled to move the servos at the programmed speed. The code and logic below is used to control the velocity of the robot and set the PWM values based on the speed value set in the program or through the web-enabled interface (further described below).

```
    static void
    PWMSpd1(unsigned long g_ulRate)
    {
           if (g_cur_PWM1 < g_ulPWM1){
           g_cur_PWM1++;
           bInPosAxis1 = 0;}
           else if (g_cur_PWM1 > g_ulPWM1){
           g_cur_PWM1--;
           bInPosAxis1 = 0;}
           else {
           g_cur_PWM1 = g_ulPWM1;
           bInPosAxis1 = 1;
           }
    }
    while(1)
    {
    while (g_cur_PWM1 != g_ulPWM1 ||  g_cur_PWM2 != g_ulPWM2 ||  g_cur_PWM3
    != g_ulPWM3 ||  g_cur_PWM4 != g_ulPWM4 ||  g_cur_PWM5 != g_ulPWM5 ||
    g_cur_PWM6 != g_ulPWM6)
    {
    PWMSpd1(1);
    PWMSpd2(1);
    PWMSpd3(1);
    PWMSpd4(1);
    PWMSpd5(1);
    PWMSpd6(1);
    //
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, g_cur_PWM1 + Trim_PWM1);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_1, g_cur_PWM2 + Trim_PWM2);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, g_cur_PWM3 + Trim_PWM3);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_3, g_cur_PWM4 + Trim_PWM4);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_4, g_cur_PWM5 + Trim_PWM5);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_5, g_cur_PWM6 + Trim_PWM6);
    //
    TBdelay(Spd_PWM1);
    }
    //
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_0, g_cur_PWM1 + Trim_PWM1);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_1, g_cur_PWM2 + Trim_PWM2);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_2, g_cur_PWM3 + Trim_PWM3);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_3, g_cur_PWM4 + Trim_PWM4);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_4, g_cur_PWM5 + Trim_PWM5);
    ROM_PWMPulseWidthSet(PWM_BASE, PWM_OUT_5, g_cur_PWM6 + Trim_PWM6);
    xTaskDelay( 1 );
    //
}
```

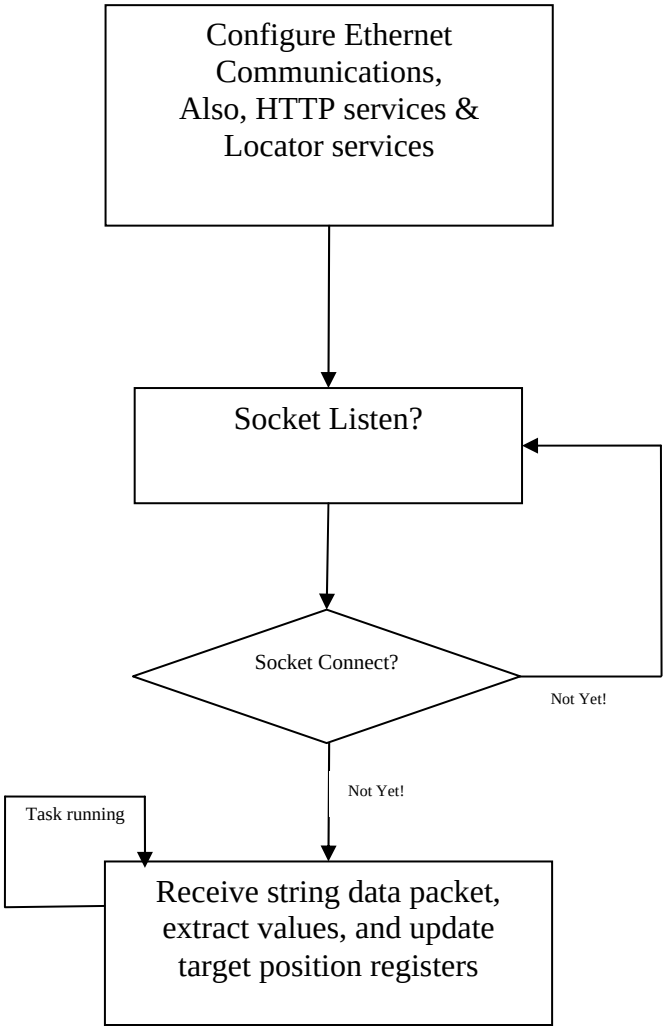
TROBOT (LM3S9B96 / SafeRTOS) Block Diagram



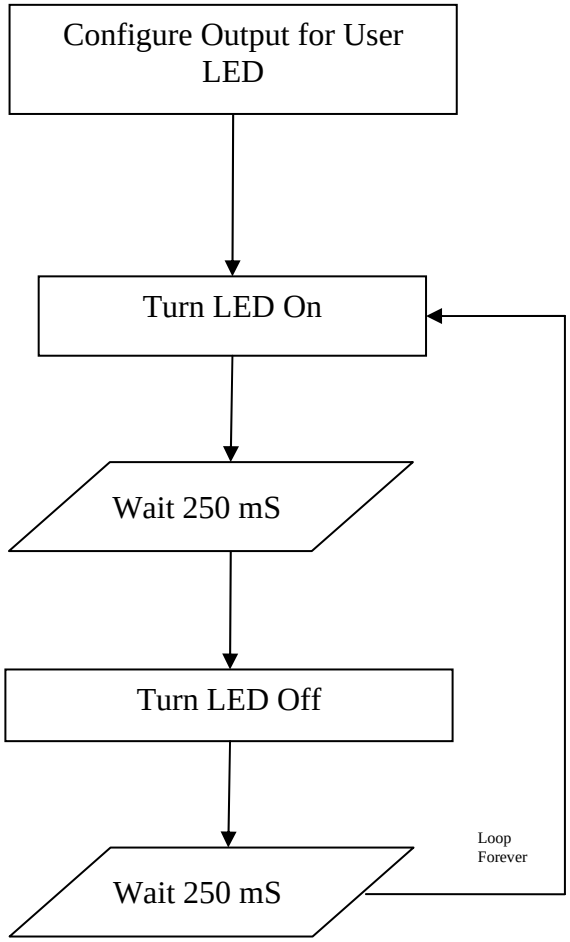
PWM Task


Configure PWM's & clocking
Compare current servo positions with Target position
Adjust current PWM value based on Rate value

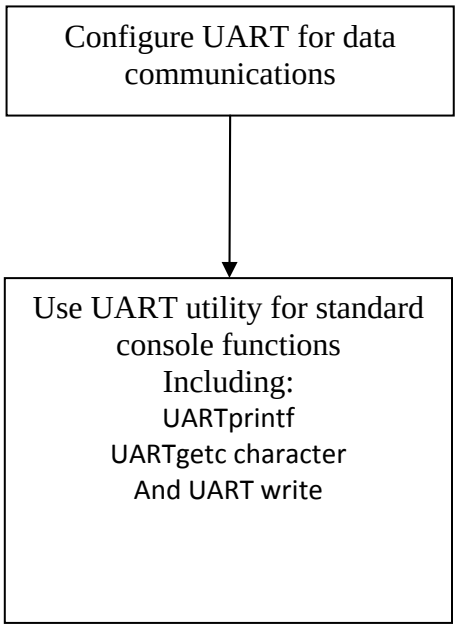
lwIP Task


Configure Ethernet Communications, Also, HTTP services & Locator services
Socket Listen?
Socket Connect?
Not Yet!
Not Yet!
Task running
Receive string data packet, extract values, and update target position registers

LED Task


Configure Output for User LED
Turn LED On
Wait 250 mS
Turn LED Off
Wait 250 mS
Loop Forever

UART Task


Configure UART for data communications
Use UART utility for standard console functions
Including:
UARTprintf
UARTgetc character
And UART write

Schematic Diagram