MEAS KMA36 DIGITAL COMPONENT SENSOR (DCS) DRIVER FOR MicroZed

Digital Position Sensor
Software Development Kit

Detailed example software and drivers are available that execute directly, without modification, on a number of development boards that support an integrated or synthesized microprocessor. The download contains several source files intended to accelerate customer evaluation and design. The source code is written in standard ANSI C format, and all development documentation including theory/operation, register description, and function prototypes are documented in the interface file.

Specifications

- Contactless angle measurement from 0° to 360°
- Programmable resolution up to 13 bits
- I^2C communication
- Very low hysteresis
- Incremental model
- Programmable zero position
- Low power consumption

Reference Material

- Detailed information regarding operation of the IC: KMA36 Datasheet
- Detailed information regarding the Peripheral Module: KMA36 Peripheral Module
- Complete software sensor evaluation kit for MicroZed: KMA36_MicroZed.zip

Drivers & Software

Detailed example software and drivers are available that execute directly, without modification, on a number of development boards that support an integrated or synthesized microprocessor. The download contains several source files intended to accelerate customer evaluation and design. The source code is written in standard ANSI C format, and all development documentation including theory/operation, register description, and function prototypes are documented in the interface file.

Functions Summary

Enumerations	
enum	**kma36_address { kma36_i2c_address_GND, kma36_i2c_address_DCOILP, kma36_i2c_address_DCOILN, kma36_i2c_address_DVCC_SE, kma36_i2c_address_VCC }**
enum	**kma36_status { kma36_status_ok, kma36_status_i2c_transfer_error, kma36_status_crc_error }**
enum	**htu21d_battery_status { htu21d_battery_ok, htu21d_battery_low } kma36_oversampling { kma36_oversampling_2, kma36_oversampling_4, kma36_oversampling_8, kma36_oversampling_32 }**
Functions	
void	**kma36_init (u32)** Initializes the AXI address of the AXI IIC Core, initializes the I^2C address to 0x59 (GND).
enum kma36_status	**kma36_set_i2c_address** (enum kma36_address) Sets the configurable I^2C address of the KMA36 device.
enum kma36_status	**kma36_read_angle (float* angle)** Reads the magnetic angle data in degrees.
enum kma36_status	**kma36_sleep_enter (void)** Request KMA36 to enter sleep mode.
enum kma36_status	**kma36_sleep_exit (void)** Request KMA36 to exit sleep mode.
enum kma36_status	**kma36_enable_low_power_mode (void)** Request KMA36 to enable low power mode. In this mode, only 180 degree measurements are possible.
enum kma36_status	**kma36_disable_low_power_mode (void)** Request KMA36 to disable low power mode.
enum kma36_status	**kma36_enable_counter (void)** Request KMA36 to enable full turn counting.
enum kma36_status	**kma36_disable_counter (void)** Request KMA36 to disable full turn counting.
enum kma36_status	**kma36_enable_fast_rate (void)** Request KMA36 to enable fast measurement update rate. In fast mode, measurement accuracy is reduced. Update rate = 1 / (1.4ms * oversampling / const)
enum kma36_status	**kma36_disable_fast_rate (void)** Request KMA36 to disable fast measurement update rate.
enum kma36_status	**kma36_set_accuracy (enum kma36_oversampling)** Set KMA36 accuracy. Resolution impacts the measurement update rate. Update rate = 1 / (1.4ms * oversampling / const)
enum kma36_status	**kma36_set_resolution (u16 res)** Set KMA36 resolution.

Project Setup

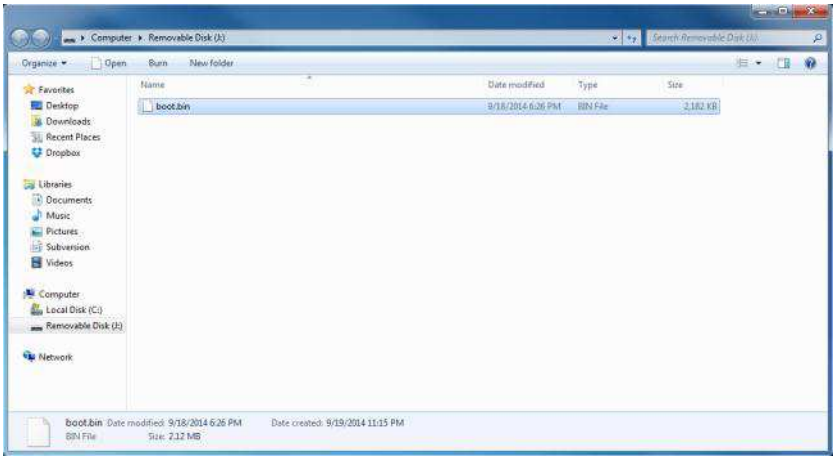
This project is based on the MicroZed board with I/O carrier card. The FPGA hardware and the console application will be loaded via micro SD card.

You will need:

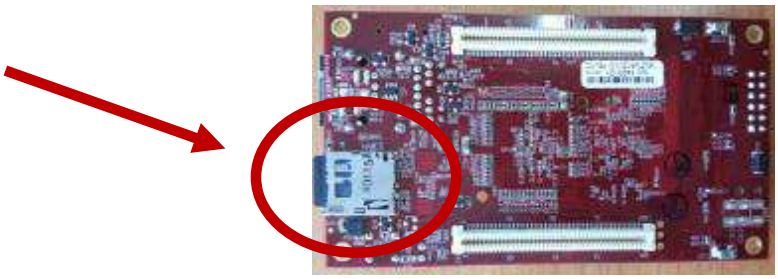
- MicroZed board
- I/O carrier card
- KMA36 sensor for Digilent Pmod™ board
- Micro SD card
- I/O carrier card power adapter
- USB-to-MicroUSB cable for UART communications
- A computer with a card reader to write to the SD card and to host a terminal emulator

The following steps will guide you through setting up the hardware platform:

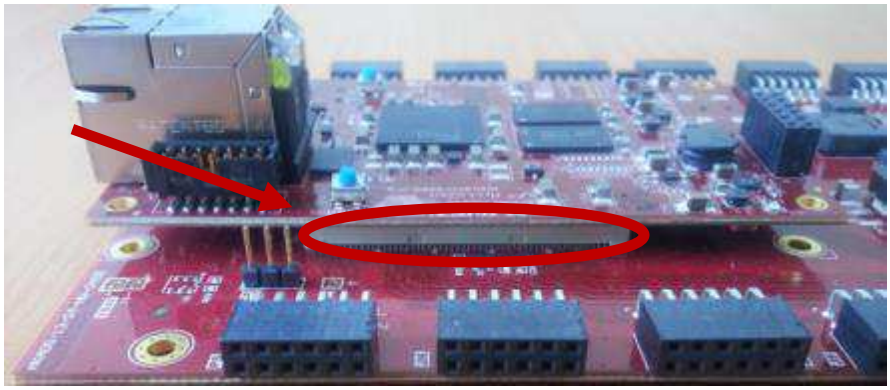
1. First, if you have not connected your computer to a ZedBoard or MicroZed device before, you will likely need to download and install the Silicon Labs CP2104 USB-to_UART driver. The setup guide for installing the driver can be found at the address below: http://www.zedboard.org/sites/default/files/documentations/CP210x_Setup_Guide_1_2.pdf
2. Next, attach the SD card to your computer via a card reader or through the built-in SD card slot. Download the “boot.bin” file that pertains to the KMA36 from the MicroZed software link and copy it onto the SD card so that it is the only file present on the file system.



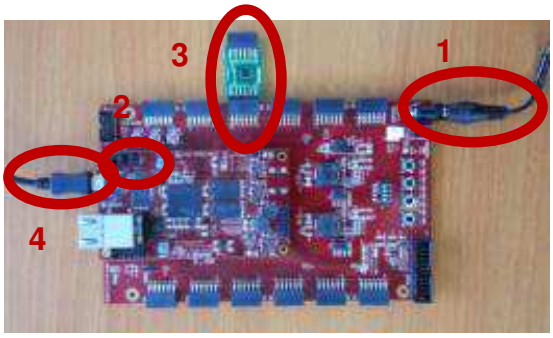
3. Safely eject the micro SD card from your computer. Insert the micro SD card into the card slot on the back of the MicroZed board.

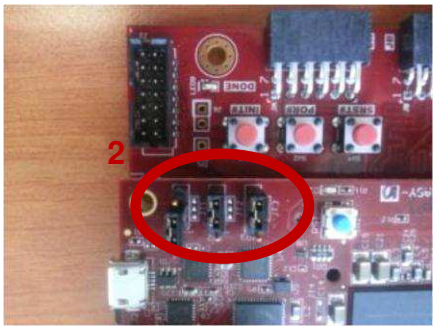


4. Carefully line up the MicroZed board with the I/O carrier card and push them together until snug.

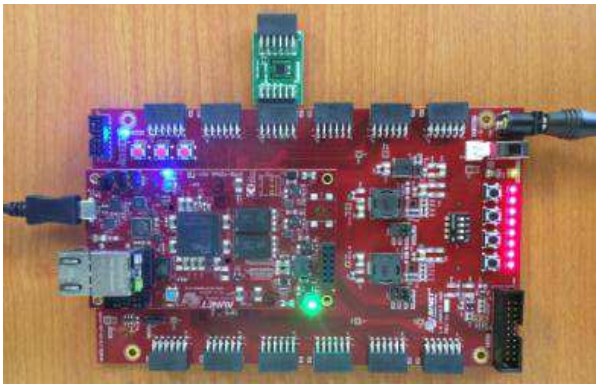


5. Connect the KMA36 digital magnetic encoder sensor to the “JC” Digilent Pmod™ port of the I/O carrier card, ensure that jumpers J1, J2, and J3 are configure such that the MicroZed will boot from the SD card on start up, and connect the power adapter to the barrel jack on the I/O carrier card (shown on the right). Finally connect the micro-USB cable to the micro-USB port of the MicroZed (shown at the left). The USB cable will facilitate UART transmissions for the console application.





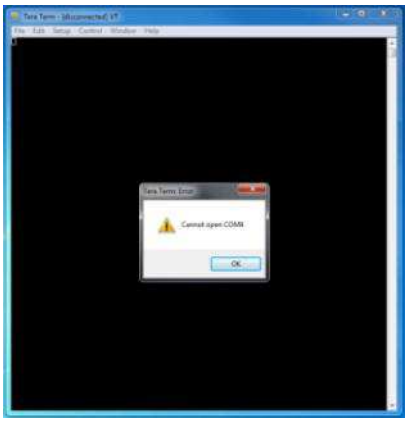
6. Turn on the power to the board with the switch next to the barrel jack. When the board powers up, the MicroZed will briefly illuminate a red LED, which will then turn off after less than a second. Once the FPGA has been successfully programmed by the boot image on the SD card, a blue “Done” LED will illuminate on both the MicroZed and the I/O carrier card. Your hardware should appear as shown below. If the board was powered on before this step, turn the power off and repeat this step.



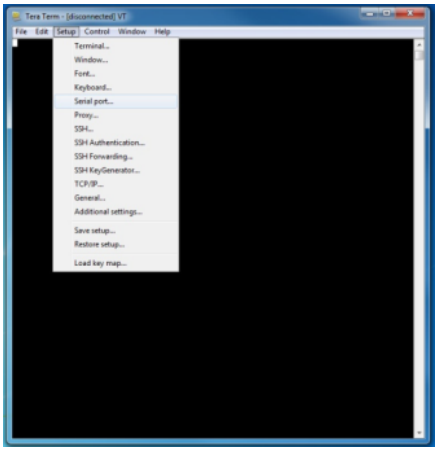
Launching the Console Application

Now that you have successfully set up your hardware platform, you are ready to run the console application.

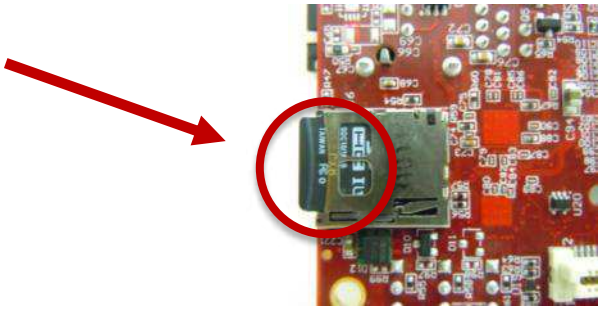
1. Upon power-on, the console application should already be running. It will be necessary to open a terminal and configure a serial connection to interact with the console application. Do this by opening tera term (which can be downloaded from http://en.sourceforge.jp/projects/ttssh2/releases/) or a similar terminal emulation software package.



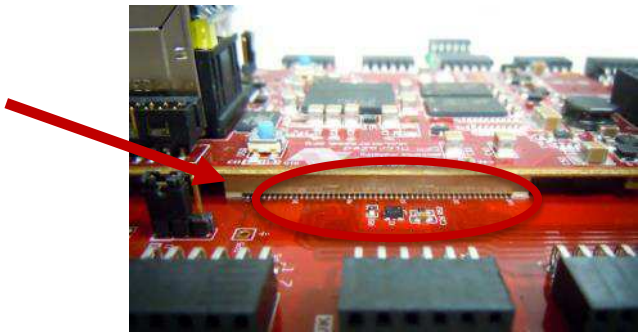
2. Tera Term may display an error when it starts up if it tries to connect to a COM port where no device is present. It is safe to ignore this warning, so click OK. Next, open the “Setup” menu and click the “Serial Port…” option.



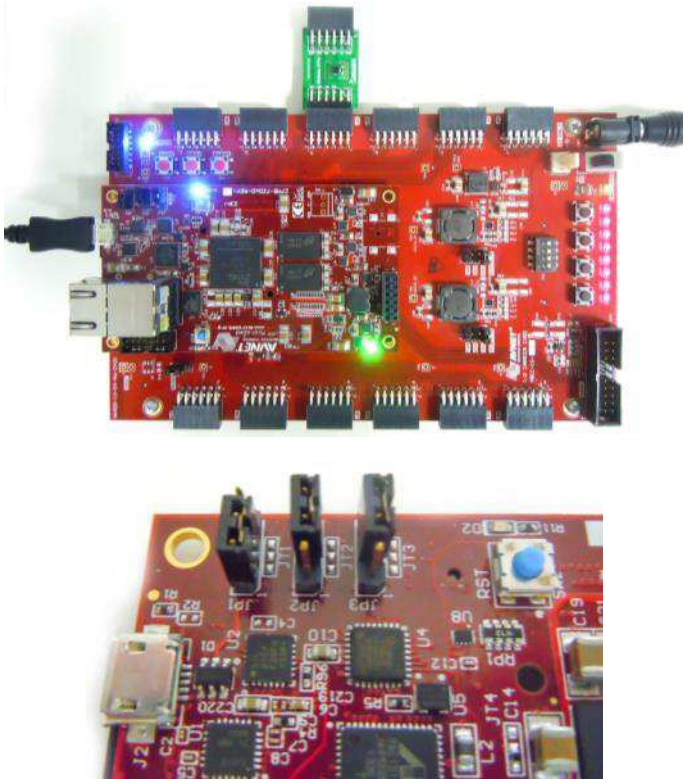
3. Safely eject the micro SD card from your computer. Insert the micro SD card into the card slot on the back of the MicroZed board.



4. Carefully line up the MicroZed board with the I/O carrier card and push them together until snug.

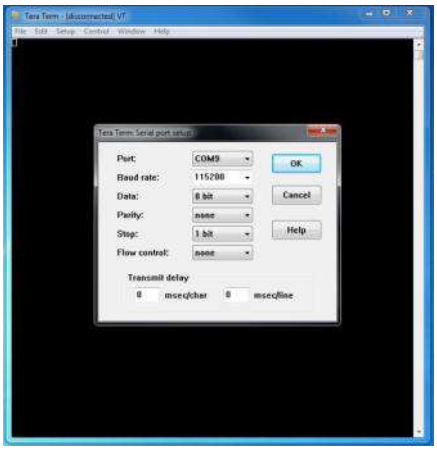


5. Connect the KMA36 digital position sensor to the “JC” Digilent Pmod™ port of the I/O carrier card, ensure that jumpers J1, J2, and J3 are configure such that the MicroZed will boot from the SD card on start up, and connect the power adapter to the barrel jack on the I/O carrier card (shown on the bottom). Finally connect the micro-USB cable to the micro-USB port of the MicroZed (shown at the top). The USB cable will facilitate UART transmissions for the console application.

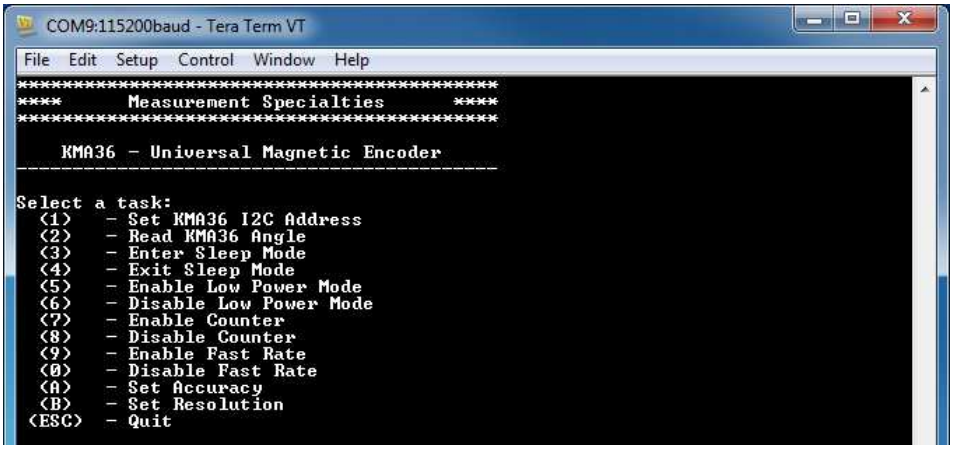


6. Turn on the power to the board with the switch next to the barrel jack. When the board powers up, the MicroZed will briefly illuminate a red LED, which will then turn off after less than a second. Once the FPGA has been successfully programmed by the boot image on the SD card, a blue “Done” LED will illuminate on both the MicroZed and the I/O carrier card. Your hardware should appear as shown below. If the board was powered on before this step, turn the power off and repeat this step.

7. Now select the appropriate COM port that your MicroZed setup is connected to. If you are not sure which this is, refer to the Device Manager. Configure your serial connection with 115200 Baud, 8 bit data, no parity, 1 stop bit, and no flow control, and then click OK.



8. You should now have a live connection open to the console application running on the MicroZed. Press enter and the console application will display the main menu from which you can perform several tasks on the KMA36 digital magnetic encoder sensor.



Running the Console Application

The console application is intended to demonstrate the required operations when using the sensor.

a. The KMA36 software must have an I²C address set or it may not function. Do this by selecting (1) and selecting the correct address **BEFORE** performing any other options.

Now the sensor and the software are setup and ready to use. This first step only needs to be performed at power up.

a. The console application option (2) reads the magnetic rotation in degrees and displays it to the console.

b. The console application option (3) sends the I²C command to enter the KMA36 into sleep mode.

c. The console application option (4) sends the I²C command to exit sleep mode.

d. The console application option (5) sends the I²C command to enable low power mode.

e. The console application option (6) sends the I²C command to disable low power mode.

f. The console application option (7) sends the I²C command to enable counter.

g. The console application option (8) sends the I²C command to disable counter.

h. The console application option (9) sends the I²C command to enable fast rate.

i. The console application option (0) sends the I²C command to disable fast rate.

j. The console application option (A) displays a menu which allows the user to select from one of four possible over-sampling rates.

k. The console application option (B) displays a prompt for the user to enter an integer between 1 and 32767 to be written to the KMA36's 16-bit resolution register.

Application Code

This section is intended to provide a basic example of functionality.

```
/*
 * Copyright (c) 2009-2012 Xilinx, Inc.  All rights reserved.
 *
 * Xilinx, Inc.
 * XILINX IS PROVIDING THIS DESIGN, CODE, OR INFORMATION "AS IS" AS A
 * COURTESY TO YOU.  BY PROVIDING THIS DESIGN, CODE, OR INFORMATION AS
 * ONE POSSIBLE   IMPLEMENTATION OF THIS FEATURE, APPLICATION OR
 * STANDARD, XILINX IS MAKING NO REPRESENTATION THAT THIS IMPLEMENTATION
 * IS FREE FROM ANY CLAIMS OF INFRINGEMENT, AND YOU ARE RESPONSIBLE
 * FOR OBTAINING ANY RIGHTS YOU MAY REQUIRE FOR YOUR IMPLEMENTATION.
 * XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO
 * THE ADEQUACY OF THE IMPLEMENTATION, INCLUDING BUT NOT LIMITED TO
 * ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE
 * FROM CLAIMS OF INFRINGEMENT, IMPLIED WARRANTIES OF MERCHANTABILITY
 * AND FITNESS FOR A PARTICULAR PURPOSE.
 *
 */

/*
 * MEAS_KMA36_Main.c: Console Application for Testing the KMA36
 *
 * This application configures UART 16550 to baud rate 9600.
 * PS7 UART (Zynq) is not initialized by this application, since
 * bootrom/bsp configures it to baud rate 115200
 *
 * ----------------------------------------------
 * | UART TYPE   BAUD RATE                      |
 * ----------------------------------------------
 *   uartns550   9600
 *   uartlite    Configurable only in HW design
 *   ps7_uart    115200 (configured by bootrom/bsp)
```

```
 */

#include <stdio.h>
#include <unistd.h>
#include "platform.h"
#include "xparameters.h"
#include "kma36.h"

void kma36_main_menu(void);

int main()
{

    char key_input;
    u8 address_set_flag=0;
    kma36_status stat;
    float angle;
    u32 res=0;

    //Initialize the UART
    init_platform();

    // Set the AXI address of the IIC core and
    // initialize the i2c address to 0x77
    kma36_init(XPAR_AXI_IIC_JC_BASEADDR);

    // Display the main menu
    kma36_main_menu();

    // Infinite loop
    while(1){

        // Get keyboard input
        read(1, (char*)&key_input, 1);

        if(key_input == '1'){               //If the '1' key is pressed

            // Display address selection menu
            printf("\n");
            printf("Select an address:\n");
            printf("  (0)   - A0 is tied to GND     (Address=0x59)\n");
            printf("  (1)   - A0 is tied to DCOILP  (Address=0x5A)\n");
            printf("  (2)   - A0 is tied to DCOILN  (Address=0x5B)\n");
            printf("  (3)   - A0 is tied to DVCC_SE (Address=0x5C)\n");
            printf("  (4)   - A0 is tied to VCC     (Address=0x5D)\n");

            // Get keyboard input ignoring keypresses that are not '0' or '1' or '2' or '3' or '4'
            read(1, (char*)&key_input, 1);
            while(key_input!='0' && key_input!='1' && key_input!='2' && key_input!='3' && key_input!='4'){
                read(1, (char*)&key_input, 1);
            }

            if(key_input == '0'){           // If the '0' key is pressed
                // Set i2c address to 0x59
                kma36_set_i2c_address(kma36_i2c_address_GND);
                printf("Set KMA36 I2C Address to 0x59 (A0 tied to GND)\n");
            }else if(key_input == '1'){     // If the '1' key is pressed
                // Set i2c address to 0x5A
                kma36_set_i2c_address(kma36_i2c_address_DCOILP);
                printf("Set KMA36 I2C Address to 0x5A (A0 tied to DCOILP)\n");
            }else if(key_input == '2'){     // If the '2' key is pressed
                // Set i2c address to 0x5B
                kma36_set_i2c_address(kma36_i2c_address_DCOILN);
                printf("Set KMA36 I2C Address to 0x5B (A0 tied to DCOILN)\n");
            }else if(key_input == '3'){     // If the '3' key is pressed
                // Set i2c address to 0x5C
                kma36_set_i2c_address(kma36_i2c_address_DVCC_SE);
                printf("Set KMA36 I2C Address to 0x5C (A0 tied to DVCC_SE)\n");
            }else if(key_input == '4'){     // If the '4' key is pressed
                // Set i2c address to 0x5D
                kma36_set_i2c_address(kma36_i2c_address_VCC);
```

```
        printf("Set KMA36 I2C Address to 0x5D (A0 tied to VCC)\n");
    }

    address_set_flag = 1;
    printf("Reading initial register state...\n");
    stat = kma36_read_regs();
    if(stat==kma36_status_ok){
      printf("Register read successful.\n");
    }else{
      printf("Register read failed.\n");
    }
    // Wait for another key press and then display the main menu again
    printf("\nPress any key to continue...\n");
    read(1, (char*)&key_input, 1);
    kma36_main_menu();

}else if(key_input == '2'){         //If the '2' key is pressed

    if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
        printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
    }else{
                                    // Send the angle read command to the KMA36
                                    printf("\n");
                                    printf("Reading current angle from KMA36...\n");
                                    stat = kma36_read_angle(&angle);

                                    // Display the status returned from the angle read operation
                                    printf("KMA36 Angle Read Complete with status: ");
                                    if(stat==kma36_status_ok)
                                              printf("Ok.\n");
                                              printf("Angle: %4.1f%c\n",angle,248);
                                    if(stat==kma36_status_i2c_transfer_error)
                                              printf("Transfer Error.\n");
    }

    // Wait for another key press and then display the main menu again
                        printf("\nPress any key to continue...\n");
                        read(1, (char*)&key_input, 1);
                        kma36_main_menu();


}else if(key_input == '3'){         // If the '3' key is pressed

    if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
        printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
    }else{
                                    // Send request to KMA36 to enter sleep mode
                                    printf("\n");
                                    printf("KMA36 Entering Sleep Mode...\n");
                                    stat = kma36_sleep_enter();

                                    // Display status returned from enter sleep mode operation
                                    printf("Enter Sleep Mode Complete with status: ");
                                    if(stat==kma36_status_ok)
                                              printf("Ok.\n");
                                    if(stat==kma36_status_i2c_transfer_error)
                                              printf("Transfer Error.\n");
    }

    // Wait for another key press and then display the main menu again
    printf("\nPress any key to continue...\n");
    read(1, (char*)&key_input, 1);
    kma36_main_menu();

}else if(key_input == '4'){         // If the '4' key is pressed

    if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
        printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
    }else{
                                    // Send request to KMA36 to exit sleep mode
                                    printf("\n");
```

```
                                printf("KMA36 Exiting Sleep Mode...\n");
                                stat = kma36_sleep_exit();

                                // Display status returned from exit sleep mode operation
                                printf("Exit Sleep Mode Complete with status: ");
                                if(stat==kma36_status_ok)
                                        printf("Ok.\n");
                                if(stat==kma36_status_i2c_transfer_error)
                                        printf("Transfer Error.\n");
    }

    // Wait for another key press and then display the main menu again
    printf("\nPress any key to continue...\n");
    read(1, (char*)&key_input, 1);
    kma36_main_menu();

}else if(key_input == '5'){         // If the '5' key is pressed

    if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
        printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
    }else{
                                // Send request to KMA36 to enable low power mode
                                printf("\n");
                                printf("KMA36 Enabling Low Power Mode...\n");
                                stat = kma36_enable_low_power_mode();

                                // Display status returned from enable low power operation
                                printf("Enable Low Power Mode Complete with status: ");
                                if(stat==kma36_status_ok)
                                        printf("Ok.\n");
                                if(stat==kma36_status_i2c_transfer_error)
                                        printf("Transfer Error.\n");
    }

    // Wait for another key press and then display the main menu again
    printf("\nPress any key to continue...\n");
    read(1, (char*)&key_input, 1);
    kma36_main_menu();

}else if(key_input == '6'){         // If the '6' key is pressed

    if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
        printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
    }else{
                                // Send request to KMA36 to disable low power mode
                                printf("\n");
                                printf("KMA36 Disabling Low Power Mode...\n");
                                stat = kma36_disable_low_power_mode();

                                // Display status returned from disable low power operation
                                printf("Disable Low Power Mode Complete with status: ");
                                if(stat==kma36_status_ok)
                                        printf("Ok.\n");
                                if(stat==kma36_status_i2c_transfer_error)
                                        printf("Transfer Error.\n");
    }

    // Wait for another key press and then display the main menu again
    printf("\nPress any key to continue...\n");
    read(1, (char*)&key_input, 1);
    kma36_main_menu();

}else if(key_input == '7'){         // If the '7' key is pressed

    if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
        printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
    }else{
                                // Send request to KMA36 to enable counter
                                printf("\n");
                                printf("KMA36 Enabling Counter...\n");
                                stat = kma36_enable_counter();
```

```
                                    // Display status returned from enable counter operation
                                    printf("Enable Counter Complete with status: ");
                                    if(stat==kma36_status_ok)
                                              printf("Ok.\n");
                                    if(stat==kma36_status_i2c_transfer_error)
                                              printf("Transfer Error.\n");
        }

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
        kma36_main_menu();

    }else if(key_input == '8'){         // If the '8' key is pressed

        if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
            printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
        }else{
                                    // Send request to KMA36 to disable counter
                                    printf("\n");
                                    printf("KMA36 Disabling Counter...\n");
                                    stat = kma36_disable_counter();

                                    // Display status returned from disable counter operation
                                    printf("Disable Counter Complete with status: ");
                                    if(stat==kma36_status_ok)
                                              printf("Ok.\n");
                                    if(stat==kma36_status_i2c_transfer_error)
                                              printf("Transfer Error.\n");
        }

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
        kma36_main_menu();

    }else if(key_input == '9'){         // If the '9' key is pressed

        if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
            printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
        }else{
                                    // Send request to KMA36 to enable fast rate
                                    printf("\n");
                                    printf("KMA36 Enabling Fast Rate...\n");
                                    stat = kma36_enable_fast_rate();

                                    // Display status returned from enable fast rate operation
                                    printf("Enable Fast Rate Complete with status: ");
                                    if(stat==kma36_status_ok)
                                              printf("Ok.\n");
                                    if(stat==kma36_status_i2c_transfer_error)
                                              printf("Transfer Error.\n");
        }

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
        kma36_main_menu();

    }else if(key_input == '0'){         // If the '0' key is pressed

        if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
            printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
        }else{
                                    // Send request to KMA36 to disable fast rate
                                    printf("\n");
                                    printf("KMA36 Disabling Fast Rate...\n");
                                    stat = kma36_disable_fast_rate();

                                    // Display status returned from disable fast rate operation
```

```
                                    printf("Disable Fast Rate Complete with status: ");
                                    if(stat==kma36_status_ok)
                                              printf("Ok.\n");
                                    if(stat==kma36_status_i2c_transfer_error)
                                              printf("Transfer Error.\n");
        }

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
        kma36_main_menu();

    }else if(key_input == 'a' || key_input == 'A'){              //If the 'a' or 'A' key is pressed

        if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
            printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
        }else{
                                    // Display oversampling selection menu
                                    printf("\n");
                                    printf("Select an oversampling rate:\n");
                                    printf("  (0)   - Oversampling Rate 2\n");
                                    printf("  (1)   - Oversampling Rate 4\n");
                                    printf("  (2)   - Oversampling Rate 8\n");
                                    printf("  (3)   - Oversampling Rate 32\n");

                                    // Get keyboard input ignoring keypresses that are not '0' or '1' or '2' or '3'
                                    read(1, (char*)&key_input, 1);
                                    while(key_input!='0' && key_input!='1' && key_input!='2' && key_input!='3'){
                                              read(1, (char*)&key_input, 1);
                                    }

                                    if(key_input == '0'){           // If the '0' key is pressed
                                              // Set oversampling to 2
                                              kma36_set_accuracy(kma36_oversampling_2);
                                              printf("Set KMA36 Oversampling Rate to 2\n");
                                    }else if(key_input == '1'){     // If the '1' key is pressed
                                              // Set oversampling to 4
                                              kma36_set_accuracy(kma36_oversampling_4);
                                              printf("Set KMA36 Oversampling Rate to 4\n");
                                    }else if(key_input == '2'){     // If the '2' key is pressed
                                              // Set oversampling to 8
                                              kma36_set_accuracy(kma36_oversampling_8);
                                              printf("Set KMA36 Oversampling Rate to 8\n");
                                    }else if(key_input == '3'){     // If the '3' key is pressed
                                              // Set oversampling to 32
                                              kma36_set_accuracy(kma36_oversampling_32);
                                              printf("Set KMA36 Oversampling Rate to 32\n");
                                    }
        }

        // Wait for another key press and then display the main menu again
        printf("\nPress any key to continue...\n");
        read(1, (char*)&key_input, 1);
        kma36_main_menu();

    }else if(key_input == 'b' || key_input == 'B'){         // If the 'b' or 'B' key is pressed

        if(address_set_flag==0){          // Address was not set yet--cannot perform this operation
            printf("KMA36 I2C Address has not yet been set.  Cannot complete this operation.\n");
        }else{
                                    res = 0;
                                    // If resolution is out of bounds, get a new number
                                    while(res<1 || res>32767){
                                              res = 0;
                                              // Display oversampling selection menu
                                              printf("\nSpecify a resolution between 1 and 32767:\n  ");
                                              // Get keyboard input ignoring keypresses that are not numbers or the enter key
                                              read(1, (char*)&key_input, 1);
                                              if(key_input=='0' || key_input=='1' || key_input=='2' || key_input=='3' || key_input=='4' ||
key_input=='5' || key_input=='6' || key_input=='7' || key_input=='8' || key_input=='9'){
                                                        res *= 10;
```

```
                                                  res += (key_input-0x30);
                                                  printf("%c",key_input);
                                                  fflush(stdout);
                                        }
                                        while(key_input!=(0x0D)){
                                                  read(1, (char*)&key_input, 1);
                                                  if(key_input=='0' || key_input=='1' || key_input=='2' || key_input=='3' ||
key_input=='4' || key_input=='5' || key_input=='6' || key_input=='7' || key_input=='8' || key_input=='9'){
                                                            res *= 10;
                                                            res += (key_input-0x30);
                                                            printf("%c",key_input);
                                                            fflush(stdout);
                                                  }
                                        }
                                        if(res<1 || res>32767){
                                                  printf("\n\nInvalid Resolution Value \"%u\".  Press any key to continue...\n",(unsigned
int)res);
                                                  read(1, (char*)&key_input, 1);
                                                  kma36_main_menu();
                                        }else{
                                                  kma36_set_resolution((u16)res);
                                                  printf("\n\nSet Resolution to %u\n",(unsigned int)res);
                                        }
                              }
            }

            // Wait for another key press and then display the main menu again
            printf("\nPress any key to continue...\n");
            read(1, (char*)&key_input, 1);
            kma36_main_menu();

        }else if(key_input == 27){    // If the 'ESC' key is pressed

            // Print done and exit.
            printf("Done.\n");
            break;

        }else{                      // If some other key is pressed

            // Redisplay the main menu
              kma36_main_menu();

        }

    }

    return 0;

}

void kma36_main_menu(void){

    //Clear the screen
    printf("\033[2J");

    //Display the main menu
    printf("*******************************************\n");
    printf("****      Measurement Specialties      ****\n");
    printf("*******************************************\n");

    printf("\n");
    printf("    KMA36 - Universal Magnetic Encoder     \n");
    printf("-------------------------------------------\n");

    printf("\n");
    printf("Select a task:\n");
    printf("  (1)   - Set KMA36 I2C Address\n");
    printf("  (2)   - Read KMA36 Angle\n");
    printf("  (3)   - Enter Sleep Mode\n");
    printf("  (4)   - Exit Sleep Mode\n");
    printf("  (5)   - Enable Low Power Mode\n");
```

```
    printf("  (6)   - Disable Low Power Mode\n");
    printf("  (7)   - Enable Counter\n");
    printf("  (8)   - Disable Counter\n");
    printf("  (9)   - Enable Fast Rate\n");
    printf("  (0)   - Disable Fast Rate\n");
    printf("  (A)   - Set Accuracy\n");
    printf("  (B)   - Set Resolution\n");
    printf(" (ESC)  - Quit\n");
    printf("\n");

    return;
}
```


te.com/sensorsolutions

PRODUCT SHEET

MEAS France SAS,
a TE Connectivity company.
Impasse Jeanne Benozzi CS 83 163
31027 Toulouse Cedex 3, FRANCE
Tel:+33 (0) 5 820 822 02
Fax: +33 (0) 5 820 821 51
customercare.tlse@te.com