✓RoHS

MEAS MS5637 DIGITAL COMPONENT SENSOR (DCS) DRIVER FOR ZedBoard

Digital Pressure and Temperature 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

- Measures pressure from 300mbar to 1200mbar
- Measures temperature from -40°C to 125°C
- I^{2}C communication
- Fully calibrated
- Fast response time
- Very low power consumption

Reference Material

- Detailed information regarding operation of the IC: MS5637 Datasheet
- Detailed information regarding the Peripheral Module: MS5637 Peripheral Module
- Complete software sensor evaluation kit for ZedBoard: MS5637_ZedBoard.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	**ms5637_status { ms5637_status_ok, ms5637_status_i2c_transfer_error, ms5637_status_crc_error }**
enum	**ms5637_resolution_osr { ms5637_resolution_osr_256, ms5637_resolution_osr_512, ms5637_resolution_osr_1024, ms5637_resolution_osr_2048, ms5637_resolution_osr_4096, ms5637_resolution_osr_8192 }**
Functions	
void	**ms5637_init (u32)** Initializes the AXI address of the AXI IIC Core and the internal resolution variable to ms5637_resolution_osr_8192 to reflect the sensor's initial resolution value on reset.
enum ms5637_status	**ms5637_reset (void)** Sends I^{2}C reset command to the MS5637 device.
enum ms5637_status	**ms5637_read_prom (void)** Reads the factory calibrated coefficients for use in temperature and pressure conversion.
enum ms5637_status	**ms5637_set_resolution (enum ms5637_resolution)** Read the user register from the device, modify its contents to reflect the resolution that is passed in to this function, and then write the updated user register value to the MS5637 device.
enum ms5637_status	**ms5637_read_temperature_and_pressure (float* t, float* p)** Send the I^{2}C commands to start a temperature conversion, wait for completion, read the temperature value, start a pressure conversion, wait for completion, read the pressure value, and use the PROM coefficients to calculate compensated values.

Project Setup

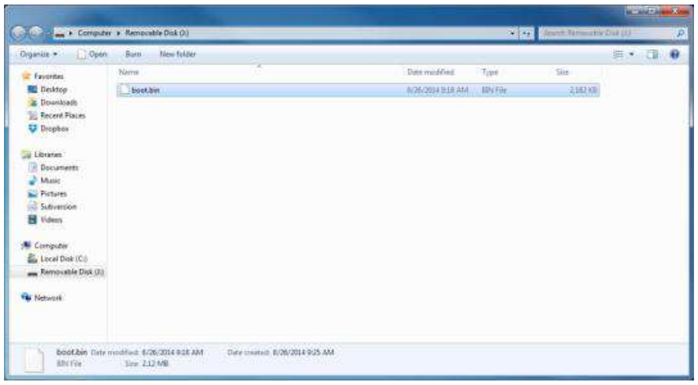
This project is based on a ZedBoard. The FPGA hardware and the console application will be loaded via SD card.

You will need:
- ZedBoard
- MS5637 sensor for Digilent Pmod™ board
- SD card
- ZedBoard 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 of 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 MS5637 from the software link and copy it onto the SD card so that it is the only file present on the file system.

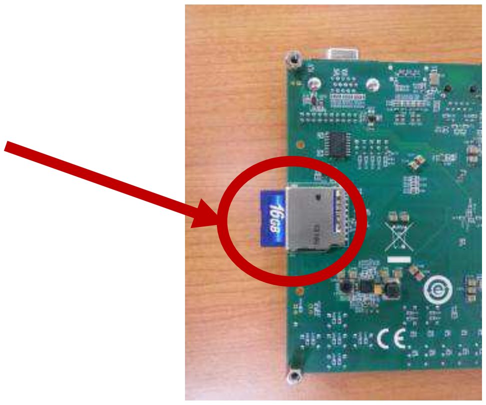
ZedBoard, MicroZed and Digilent Pmod™ are trademarks.



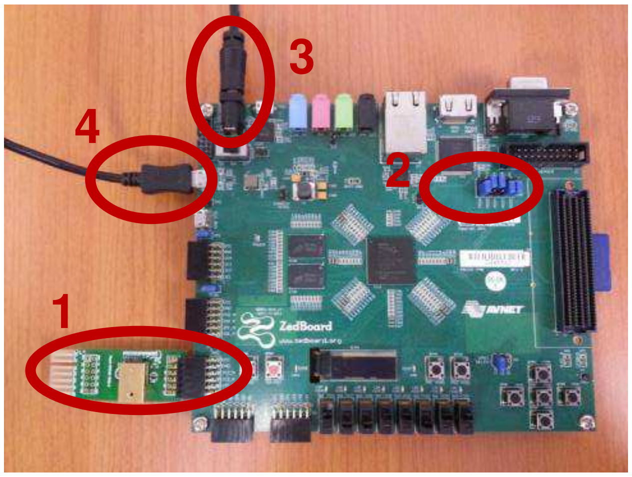
Name	Date modified	Type	Size
boot.bin	8/26/2014 9:18 AM	BIN File	2,182 KB

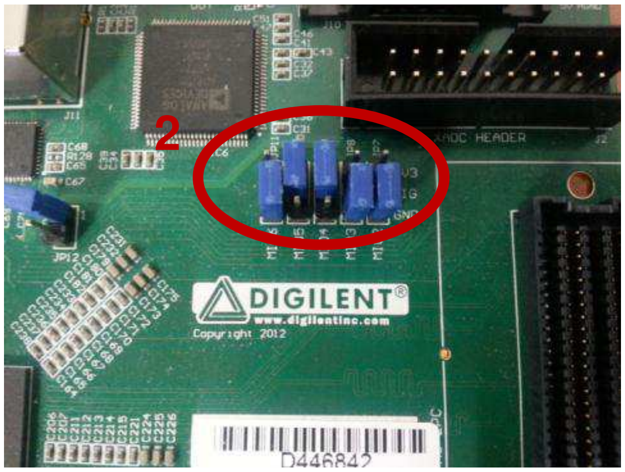
boot.bin Date modified: 8/26/2014 9:18 AM Date created: 8/26/2014 9:25 AM
BIN File Size: 2.12 MB

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



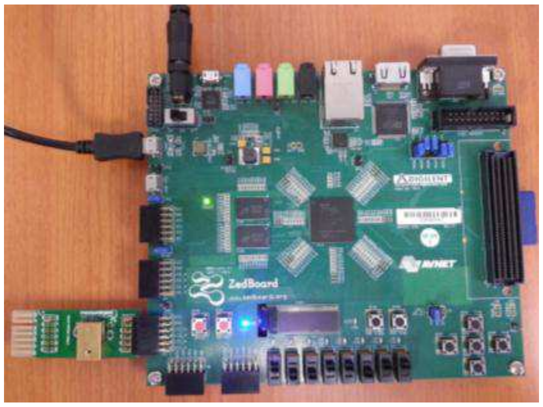
4. Connect the MS5637 digital pressure sensor to the "JC" Digilent Pmod™ port of the ZedBoard (1), ensure that jumpers JP7, JP8, JP9, JP10, and JP11 are configured such that the ZedBoard will boot from the SD card on start-up (2), and connect the power adapter to the barrel jack on the ZedBoard (3). Finally connect the micro-USB cable to the micro-USB port of the ZedBoard that is labeled "UART" (4). The USB cable will facilitate UART transmissions for the console application.





ZedBoard and Digilent Pmod™ are trademarks.

5. Turn on the power to the board with the switch next to the barrel jack. When the board powers up, the ZedBoard will illuminate a green power LED. After close to 30 seconds, the FPGA will be successfully programmed by the boot image on the SD card and a blue "Done" LED will illuminate on the ZedBoard. Your hardware should appear as shown below. If the board was powered on before this step, turn the power off and repeat this step.



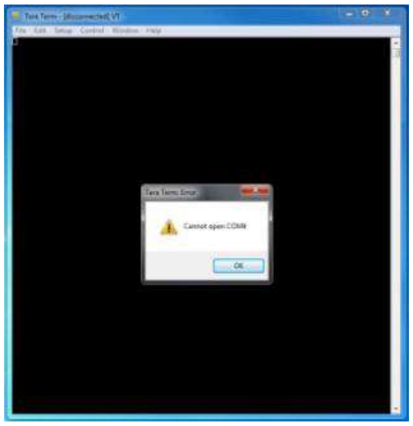
Launching the Console Application

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

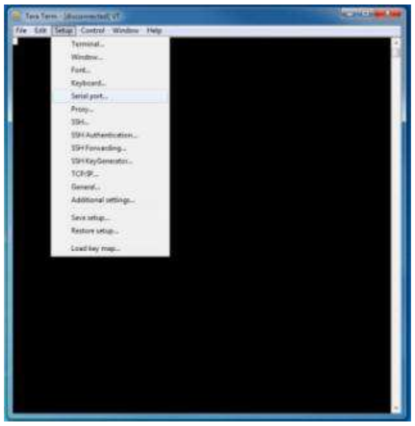
1. Upon power-on, the console application should already be running. It will be necessary to open a terminal and configure a serial connection to interact with the console application. Do this by opening tera term (which can be downloaded from http://en.sourceforge.jp/projects/ttssh2/releases/) or a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

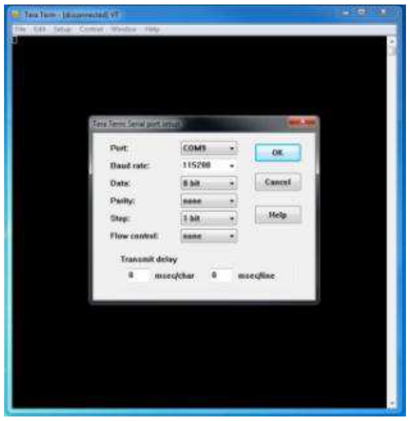
2. Tera term may display an error when it starts up if it tries to connect to a COM port where no device is present. It is safe to ignore this warning, so click OK. Next, open the "Setup" menu and click the "Serial Port…" option.

3. Now select the appropriate COM port that your ZedBoard 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.


Tera Term - [disconnected] VT
File Edit Setup Control Window Help
Tera Term: Error
OK

Cannot open COM

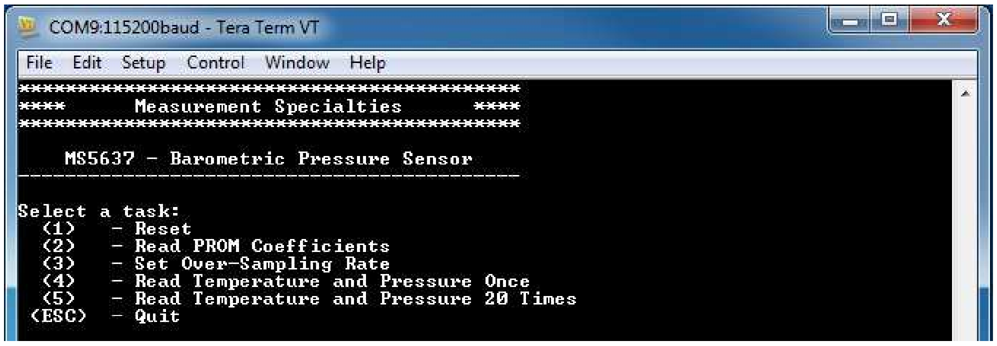



Tera Term - [disconnected] VT
File Edit Setup Control Window Help
Tera Term: Serial port setup
OK
Cancel
Help

Port: COM9
Baud rate: 115200
Data: 8 bit
Parity: none
Stop: 1 bit
Flow control: none
Transmit delay
0 msec/char 0 msec/line

ZedBoard is a trademark.

4. You should now have a live connection open to the console application running on the ZedBoard. Press enter and the console application will display the main menu from which you can perform several tasks on the MS5637 digital pressure sensor.


COM9:115200baud - Tera Term VT
File Edit Setup Control Window Help

```
*******************************************
****      Measurement Specialties     ****
*******************************************

    MS5637 - Barometric Pressure Sensor
-------------------------------------------

Select a task:
  (1)    - Reset
  (2)    - Read PROM Coefficients
  (3)    - Set Over-Sampling Rate
  (4)    - Read Temperature and Pressure Once
  (5)    - Read Temperature and Pressure 20 Times
 (ESC)   - Quit
```

Running the Console Application

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

a. After startup, it is a good idea to reset the sensor. This puts it in a known state. Do this by selecting (1) in the console application.

b. Each sensor is tested at the factory to determine the variation of the sensor due to fabrication. Calibration coefficients are stored in the device at that time for later use in calculating the correct output. These coefficient values must be read from the device and stored in software variables before temperature and pressure measurements can be taken. Do this by selecting (2) in the console application.

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

c. The console application option (3) displays a menu that allows the user to select from the six possible over-sampling rates of the sensor.

d. The console application option (4) reads both the temperature and pressure values and displays each of them once.

e. The console application option (5) reads the temperature and pressure 20 times each at approximately two measurement pairs per second and displays them to the screen in real time.

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_MS5637_Main.c: Console Application for Testing the MS5637
 *
 * 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 "sleep.h"
#include "ms5637.h"

#define               XPAR_AXI_IIC_JC_BASEADDR        XPAR_IIC_0_BASEADDR

void ms5637_main_menu(void);

int main()
{

    char key_input;
    char prom_read_flag=0;
    int i;
    ms5637_status stat;
    float temperature;
    float pressure;

    //Initialize the UART
    init_platform();

    // Set the AXI address of the IIC core
    ms5637_init(XPAR_AXI_IIC_JC_BASEADDR);

    // Display the main menu
    ms5637_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

            // Send the reset command to the MS5637
            printf("\n");
            printf("Resetting MS5637...\n");
            stat = ms5637_reset();

            // Display the status returned from the reset operation
            printf("MS5637 Reset Complete with status: ");
            if(stat==ms5637_status_ok)
                printf("Ok.\n");
            if(stat==ms5637_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);
            ms5637_main_menu();

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

            // Read the PROM coefficients from the MS5637
            printf("\n");
            printf("Reading PROM Coefficients...\n");
            stat = ms5637_read_prom();

            // Display status returned from read_prom operation
            // and display prom values if successful
            printf("Read PROM Complete with status: ");
            if(stat==ms5637_status_ok){
                prom_read_flag=1;
                printf("Ok.\n");
                printf("\n");
                printf("_________________________________________________\n");
                printf("|   PROM Addr   |Coeff (Base 10)|  Coeff (Hex)  |\n");
                printf("|---------------+---------------+---------------|\n");
                for(i=0;i<7;i++){
                    printf("|\t%d\t|     %5d\t|     0x%4X\t|\n",i,(unsigned int)ms5637_prom_coeffs[i],(unsigned int)ms5637_prom_coeffs[i]);
                }
            }else if(stat==ms5637_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);
            ms5637_main_menu();

        }else if(key_input == '3'){

           // Display resolution selection menu
            printf("\n");
            printf("Select a resolution (over-sampling rate):\n");
            printf("  (1)   - 256\n");
            printf("  (2)   - 512\n");
            printf("  (3)   - 1024\n");
            printf("  (4)   - 2048\n");
            printf("  (5)   - 4096\n");
            printf("  (6)   - 8192\n");

            // Get keyboard input ignoring keypresses that are not '1' or '2' or '3' or '4' or '5' or '6'
            read(1, (char*)&key_input, 1);
            while(key_input!='1' && key_input!='2' && key_input!='3' && key_input!='4' && key_input!='5' && key_input!='6'){
                read(1, (char*)&key_input, 1);
            }

            if(key_input == '1'){           // If the '1' key is pressed
                // Set OSR to 256
                ms5637_resolution = ms5637_resolution_osr_256;
                printf("\nSet MS5637 Over-Sampling Rate to 256\n");
            }else if(key_input == '2'){     // If the '2' key is pressed
                // Set OSR to 512
                ms5637_resolution = ms5637_resolution_osr_512;
                printf("\nSet MS5637 Over-Sampling Rate to 512\n");
            }else if(key_input == '3'){     // If the '3' key is pressed
                // Set OSR to 1024
                ms5637_resolution = ms5637_resolution_osr_1024;
                printf("\nSet MS5637 Over-Sampling Rate to 1024\n");
            }else if(key_input == '4'){     // If the '4' key is pressed
                // Set OSR to 2048
                ms5637_resolution = ms5637_resolution_osr_2048;
                printf("\nSet MS5637 Over-Sampling Rate to 2048\n");
            }else if(key_input == '5'){     // If the '5' key is pressed
                // Set OSR to 4096
                ms5637_resolution = ms5637_resolution_osr_4096;
                printf("\nSet MS5637 Over-Sampling Rate to 4096\n");
            }else if(key_input == '6'){     // If the '6' key is pressed
                // Set OSR to 8192
                ms5637_resolution = ms5637_resolution_osr_8192;
                printf("\nSet MS5637 Over-Sampling Rate to 8192\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);
            ms5637_main_menu();

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

            if(prom_read_flag==0){          // PROM was not yet read--cannot read temperature and pressure yet
                printf("PROM Coefficients have not yet been read. Cannot complete temperature/pressure read.\n");
            }else{                          // PROM has been read--continue on to read temperature and pressure

                                           // Read one temperature value and one pressure value
                                           printf("\n");
                                           printf("Reading a Temperature Value and a Pressure Value...\n");
                                           stat = ms5637_read_temperature_and_pressure(&temperature, &pressure);
```

```
                                        // Display the status returned from the read_temperature_and_pressure
                                        // operation and display the temperature and pressure if successful
                                        printf("Temperature and Pressure Read Complete with status: ");
                                        if(stat==ms5637_status_ok){
                                                   printf("Ok.\n");
                                                   printf("Temperature : %5.2f%cC, \tPressure : %6.2fhPa",temperature,248,pressure);
                                        }else if(stat==ms5637_status_i2c_transfer_error){
                                                   printf("Transfer Error.");
                                        }
                                        printf("\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);
            ms5637_main_menu();

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

           if(prom_read_flag==0){           // PROM was not yet read--cannot read temperature and pressure yet
                printf("PROM Coefficients have not yet been read. Cannot complete temperature/pressure read.\n");
            }else{                           // PROM has been read--continue on to read temperature and pressure

                                        // Read 20 temperature values at ~2 per second
                                        printf("\n");
                                        printf("Reading 20 Temperature and Pressure Value Pairs...\n");
                                        for(i=0;i<20;i++){
                                                   stat = ms5637_read_temperature_and_pressure(&temperature, &pressure);
                                                   if(stat==ms5637_status_ok){
                                                              printf("%2d: Temperature : %5.2f%cC, \tPressure :
%6.2fhPa",i+1,temperature,248,pressure);
                                                   }else if(stat==ms5637_status_i2c_transfer_error){
                                                              printf("%2d: Transfer Error.", i+1);
                                                   }
                                                   printf("\n");
                                                   usleep( (500-MS5637_CONV_DELAY_OSR_8192)*1000 );
                                        }

            }

            // 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);
            ms5637_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
            ms5637_main_menu();

        }

    }

    return 0;

}

void ms5637_main_menu(void){

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

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

    printf("\n");
    printf("    MS5637 - Barometric Pressure Sensor    \n");
    printf("--------------------------------------------\n");

    printf("\n");
    printf("Select a task:\n");
    printf("  (1)   - Reset\n");
    printf("  (2)   - Read PROM Coefficients\n");
    printf("  (3)   - Set Over-Sampling Rate\n");
    printf("  (4)   - Read Temperature and Pressure Once\n");
    printf("  (5)   - Read Temperature and Pressure 20 Times\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