

ECER Technology
TiltCONTROL SDK DOCUMENTATION

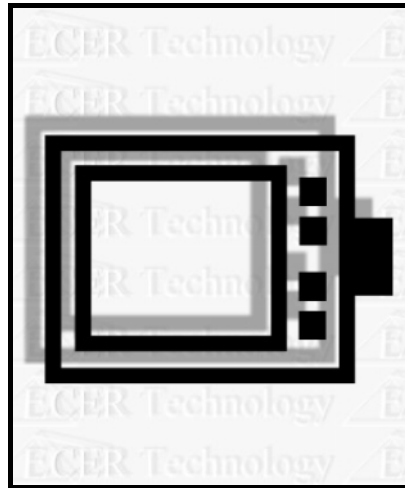


Table Of Contents

Introduction	3
Getting started	4
Setting up the development environment	4
Adding the TiltCONTROL SDK to your own project	4
Integrating the TiltCONTROL into your application	5
Using the WM_TILTDATA message to get the TiltCONTROL readings	5
Example using WM_TILTDATA messages	7
Polling the TiltCONTROL directly to get the TiltCONTROL readings	8
Example polling the TiltCONTROL directly	10
Functions	11
IntializeTiltControl	11
DestroyTiltControl	11
GetCurrentTiltControlData	12
GetCurrentAcceleration	13
GetCurrentAcceleration	14
GetTiltControlStatus	14
ConvertTiltDataMessageToTiltSensorData	15
GetStringRepresentationOfStatus	16
CalculateAngleOfRotation	17
Messages	18
WM_TILTDATA message	18
Structures	19
TILTSENSORDATA	19
TILTCONTROLSTATE	19
Distributing the TiltCONTROL Drivers with your application	20

Introduction

The purpose of this document is to give an overview of the TiltCONTROL SDK and how it is used to integrate the TiltCONTROL device into Third Party applications.

Getting started

Setting up the development environment

The TiltCONTROL SDK has been created with, and is designed to work with Microsoft eMbedded Visual C/C++ 3 or 4.

There are sample projects contained in the Samples directory that demonstrate how to use the SDK. Simply open up the project files (.dsp) in Microsoft eMbedded Visual C/C++ 4 to get started.

Read on to find out how to specifically integrate the TiltCONTROL into your own eMbedded Visual C/C++ project.

Adding the TiltCONTROL SDK to your own project

The TiltCONTROL SDK consists of a static library, a dynamically linked library and several C header files, which define the functions available in the SDK, these are

```
TiltCONTROLDD.h  
TiltControlMessages.h  
SpecialFunctions.h  
TiltControlDefinitions.h  
  
TiltCONTROLDD.lib  
  
TiltCONTROLDD.dll
```

and these can be found in the install directory for this SDK, which by default is C:\TiltCONTROL SDK.

To include the SDK into your project, simply `#include` the header files that contain the functions you want access to, into and source files that will be using the library and link TiltCONTROLDD.lib into your project.

NOTE: To specify additional libraries to be linked by your application simply select from the Project menu in eMbedded Visual C++, the Settings item. Then select the Link Tab and under the Object/Library modules label simply add the name and location of the TiltCONTROLDD.lib.

Make sure that the DLL TiltCONTROL.dll is located in the Windows directory of the target Pocket PC device. If you have the TiltCONTROL drivers installed on the device then the DLL will already be there.

Once you **#include** the header files, link the static library and made sure the DLL is on the target device, you will be ready to start developing your application which integrates the TiltCONTROL into your application.

Integrating the TiltCONTROL into your application

There are two ways to integrate the TiltCONTROL into an application:

- The application can either listen for the WM_TILTDATA messages, which are sent to the foremost window every 100ms, to get the latest TiltCONTROL readings.
- Or the application can poll the TiltCONTROL directly for its current readings.

Using the WM_TILTDATA message to get the TiltCONTROL readings

The following section will explain how to get the current TiltCONTROL readings from the WM_TILTDATA message.

The WM_TILTDATA message contains the current TiltCONTROL readings.

The WM_TILTDATA message is sent approximately every 100ms to the foremost window, by the TiltCONTROL driver. It is useful to use this method, when you are using the TiltCONTROL to control an application that is the foreground window, but do not need the readings if the application is not the foreground window.

As long as the TiltCONTROL driver is installed on the target device and the TiltCONTROL is plugged in and the TiltCONTROL is not disabled, the TiltCONTROL driver will send a WM_TILTDATA message approximately every 100ms to the foremost window. All that your application has to do is receive these messages and process them.

The WM_TILTDATA message takes the form

WM_TILTDATA

```
wHorizontalTiltData = LOWORD(wParam);
wVerticalTiltData   = HIWORD(wParam);
nTiltControlStatus = (int) lParam;
```

wHorizontalTiltData represents the left to right tilt of the TiltCONTROL.
wVerticalTiltData represents the up and down tilt of the TiltCONTROL.
nTiltControlStatus represents the current status of the TiltCONTROL, and can be one of the following values

```
TILTCONTROL_NOT_INITIALISED = 0
TILTCONTROL_SYNCHRONISING = 1
TILTCONTROL_INITIALISED_AND_CONNECTED = 2
TILTCONTROL_NOT_CONNECTED = 3
TILTCONTROL_DESTROYED = 0
```

The actual horizontal and vertical angle of tilt can be calculated using the following formula:

```
Horizontal Angle
  of tilt = ((wHorizontalTiltData / 65535) * 2 - 1) * 90;

Vertical Angle
  of tilt = ((wVerticalTiltData / 65535) * 2 - 1) * 90;
```

It is not necessary to know how to calculate the angle of tilt, as you can call the function `ConvertTiltDataMessageToTiltSensorData()` contained in `TiltControlMessages.h` to convert the message data to the more usable data contained in a `TILTSENSORDATA` structure.

`ConvertTiltDataMessageToTiltSensorData(WPARAM wParam, TILTSENSORDATA *TiltSensorData)` takes the `wParam` received within the `WM_TILTDATA` message as its first parameter and returns the current TiltCONTROL data in a `TILTSENSORDATA` structure via the second parameter.

The `TILTSENSORDATA` structure is defined in `TiltControlDefinitions.h` and is defined as

```
typedef struct
{
    double dHorizontalAccerleration;    // The left to right acceleration
                                        // in G's (m/s/s)
    double dVerticalAccerleration;     // The up and down acceleration in
                                        // G's (m/s/s)
    double dHorizontalAngleOfTilt;     // The left to right angle of tilt
                                        // in degrees
    double dVerticalAngleOfTilt;       // The up and down angle of tilt in
                                        // degrees
}TILTSENSORDATA;
```

You need to include the following items into your application to use the TiltCONTROL with the `TILTDATA` message:

- Make sure that the TiltCONTROL drivers are installed on the target device.
- `#include TiltControlMessages.h` into your source files
- Link the static library `TiltCONTROLDD.lib` into your project.
- Add a message handler for the `WM_TILTDATA` message (defined in `TiltControlMessages.h`)
- Call `ConvertTiltDataMessageToTiltSensorData()` to extract the TiltCONTROL data from the `WM_TILTDATA` message.
- Process the TiltCONTROL data that will be returned via a `TILTSENSORDATA` structure (defined in `TiltControlDefinitions.h`)

Example using WM_TILTDATA messages

Below is a sample piece of code that shows how to integrate with the TiltCONTROL using the WM_TILTDATA messages to get the current TiltCONTROL readings. It uses a Dialog Procedure to handle the WM_TILTDATA messages and display them to a dialog.

```

BOOL CALLBACK TiltCONTROLHandlerDlgProc (HWND hWnd, UINT wParam, WPARAM wParam,
                                         LPARAM lParam) {

    TILTSENSORDATA TiltSensorData;
    TCHAR szHorizontalAngle[512];
    TCHAR szVerticalAngle[512];

    switch (wParam)
    {
    case WM_TILTDATA:
        ConvertTiltDataMessageToTiltSensorData(wParam, &TiltSensorData);

        wsprintf(szHorizontalAngle, L"%.2f", TiltSensorData.dHorizontalAngleOfTilt);
        wsprintf(szVerticalAngle, L"%.2f", TiltSensorData.dVerticalAngleOfTilt);

        SetDlgItemText(hWnd, IDD_HORIZONTALANGLE, szHorizontalAngle);
        SetDlgItemText(hWnd, IDD_VERTICALANGLE, szVerticalAngle);

        SetDlgItemText(hWnd, IDD_TILTCONTROLSTATUS,
            GetStringRepresentationOfStatus(LOWORD(lParam)));
        break;
    }
}

```

There is a sample included in this SDK distribution called Messages that demonstrates how to integrate the TiltCONTROL into your application using the WM_TILTDATA message, which can be found in the Samples directory of the SDK install directory.

Polling the TiltCONTROL directly to get the TiltCONTROL readings

The following section will explain how to get the current TiltCONTROL readings by polling the TiltCONTROL directly.

The TiltCONTROL SDK consists of the following additional functions:

```
InitializeTiltControl
DestroyTiltControl
GetCurrentTiltControlData
GetCurrentAcceleration
GetCurrentAngle
GetTiltControlStatus
```

Each one of these functions is described in the following sections.

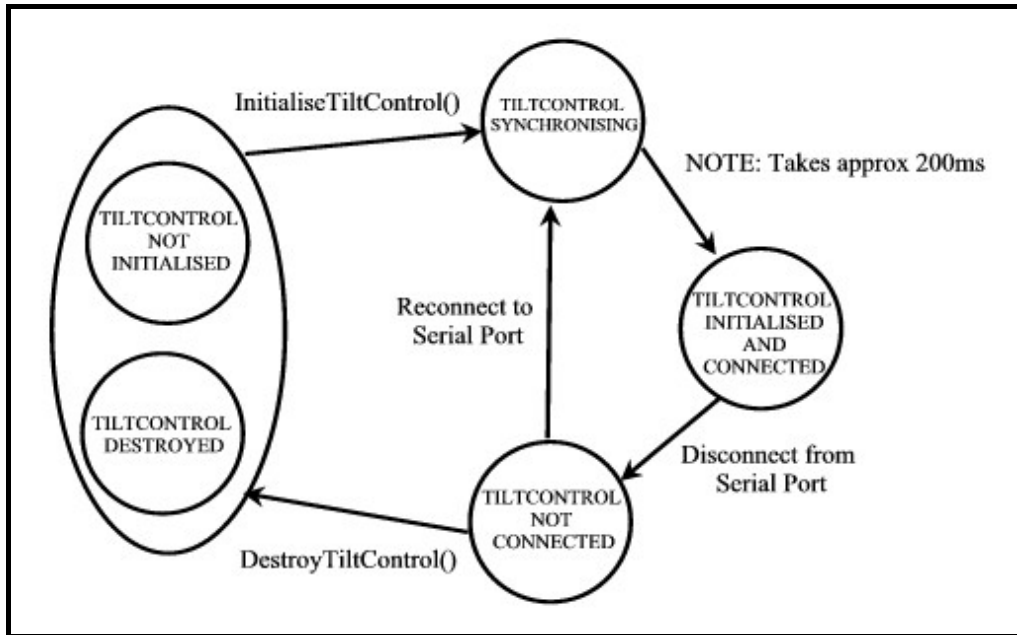
But first the basic usage of the functions will be explained.

The TiltCONTROL can be in one of 5 (actually 4 because two are essentially the same) states at any one time. The current state of the TiltCONTROL is returned as a return value from most of the functions, which can be used to determine if the function was successful, and if not, why the function failed. The TiltCONTROL states and their descriptions are shown in Table 0-1. The flow of states that the TiltCONTROL will follow is shown in the state diagram in Figure 0-1.

Table 0-1 Possible TiltCONTROL States, as defined in the enumerated type TILTCONTROLSTATE in TiltCONTROLDefinitions.h

Return Code	Integer Value	Description
TILTCONTROL_NOT_INITIALISED	0	State of the TiltCONTROL if <code>InitialiseTiltControl()</code> has never been called successfully.
TILTCONTROL_SYNCHRONISING	1	State of the TiltCONTROL while the software is trying to synchronise with the data stream coming from the TiltCONTROL. NOTE: The TiltCONTROL will only be in this state for approximately 200ms.
TILTCONTROL_INITIALISED_AND_CONNECTED	2	State of the TiltCONTROL if the TiltCONTROL is connected to the serial port and the software has synchronised with the data stream. In this state all of the functions will be successful. This is the return value that determines that a function was successful.
TILTCONTROL_NOT_CONNECTED	3	State of the TiltCONTROL if <code>InitialiseTiltControl()</code> has never been called successfully but the TiltCONTROL is not connected to the serial port.
TILTCONTROL_DESTROYED	0	State of the TiltCONTROL if <code>DestroyTiltControl()</code> has been called.

Figure 0-1 TiltCONTROL state flow



Example polling the TiltCONTROL directly

Below is a sample piece of code that shows how to integrate with the TiltCONTROL using the polling method.

```
#include "TiltCONTROLDD.h"

HANDLE hTiltCONTROL;
TILTSENSORDATA TiltSensorData;
int rc;

hTiltCONTROL = InitializeTiltControl();
if(hTiltCONTROL == CANNOT_CONNECT_TO_SERIAL_PORT)
{
    // Display error message indicating that the serial port is already in use
    // by another application
    return;
}

// Wait for the TiltCONTROL to be connected and initialised
while(GetTiltControlStatus(hTiltCONTROL) != TILTCONTROL_INITIALISED_AND_CONNECTED);

// Do what you want to do with the TiltCONTROL until you want to destroy it
while(1)
{
    // If the TiltCONTROL is still connected and initialised
    if(GetCurrentTiltControlData(hTiltControl, &TiltSensorData, 8) ==
        TILTCONTROL_INITIALISED_AND_CONNECTED)
    {
        // Do something with data in the TiltSensorData structure
    }
    else
    {
        // Display an error or do nothing because the TiltCONTROL is not connected or is
        // still synchronising
    }

    Sleep(50);
}

// Once finished don't forget to Destroy the TiltCONTROL
DestroyTiltControl(hTiltCONTROL);
```

There is a full example in the Samples directory called Direct, that comes with this SDK that demonstrates how to use the TiltCONTROL functions.

Functions

InitializeTiltControl

Prototype	<code>HANDLE InitializeTiltControl();</code>
Description	InitializeTiltControl will open the serial port for the TiltCONTROL and initialize a connection with the TiltCONTROL device, if the serial port is not in use by another application.
Inputs	None.
Outputs	Handle to the TiltCONTROL is returned.
Return Values	<code>CANNOT_CONNECT_TO_SERIAL_PORT</code> if unable to connect to the serial port. Otherwise this function will always return a handle to the serial port used to connect to the TiltCONTROL, even if the TiltCONTROL is not connected to the serial port.
Example Usage	<pre> hTiltCONTROL = InitialiseTiltControl(); if(hTiltCONTROL == CANNOT_CONNECT_TO_SERIAL_PORT) { // Display error message indicating that the serial port is already in // use by another application return; } </pre>
Notes	Always remember to call <code>DestroyTiltControl</code> once finished with the TiltCONTROL.

DestroyTiltControl

Prototype	<code>HANDLE DestroyTiltControl(HANDLE hTiltControl);</code>
Description	<code>DestroyTiltControl</code> will destroy the connection to the TiltCONTROL with the handle <code>hTiltControl</code> .
Inputs	hTiltControl: Handle to TiltCONTROL.
Outputs	Destroyed Handle.
Return Values	Should be NULL if successful.
Example Usage	<code>DestroyTiltControl(hTiltCONTROL);</code>
Notes	Always remember to call <code>DestroyTiltControl</code> once finished with the TiltCONTROL.

GetCurrentTiltControlData

Prototype	int GetCurrentTiltControlData(HANDLE hTiltControl, TILTSENSORDATA * TiltSensorData, nDigitalFilterLength);	
Description	GetCurrentTiltControlData will return both the horizontal and vertical axis sensor readings via the TiltSensorData structure.	
Inputs	hTiltControl:	Handle to TiltCONTROL.
	*TiltSensorData:	Pointer to a TILTSENSORDATA structure.
	nDigitalFilterLength:	The length of the digital filter applied to the raw data received from the accelerometer in the TiltCONTROL. The larger nDigitalFilterLength is, the more stable the sensor readings will be. The smaller nDigitalFilterLength is the more responsive the the sensor readings will be.
Outputs	Pointer to a TILTSENSORDATA structure with results.	
Return Values	The current status of the TiltCONTROL. TILTCONTROL_INITIALISED_AND_CONNECTED if successful The current TiltControl Status, other than TILTCONTROL_INITIALISED_AND_CONNECTED if unsuccessful	
Example Usage	GetCurrentTiltControlData(hTiltControl, &TiltSensorData, 8);	
Notes		

GetCurrentAcceleration

Prototype	<code>int GetCurrentAcceleration(HANDLE hTiltControl, double *dCurrentAcceleration, bool bIsHorizontal, nDigitalFilterLength);</code>	
Description	GetCurrentAcceleration will return the Horizontal or Vertical axis sensor reading via the CurrentAcceleration variable. If <code>bIsHorizontal == true</code> then the horizontal axis sensor reading will be returned else the vertical axis sensor reading will be returned.	
Inputs	hTiltControl:	Handle to TiltCONTROL
	*dCurrentAcceleration:	Pointer to the dCurrentAcceleration variable
	bIsHorizontal:	Set true if you require the horizontal axis sensor reading
	nDigitalFilterLength:	The length of the digital filter applied to the raw data received from the accelerometer in the TiltCONTROL. The larger nDigitalFilterLength is, the more stable the sensor readings will be. The smaller nDigitalFilterLength is the more responsive the the sensor readings will be.
Outputs	Pointer to the dCurrentAcceleration variable with result.	
Return Values	The current status of the TiltCONTROL. TILTCONTROL_INITIALISED_AND_CONNECTED if successful The current TiltControl Status, other than TILTCONTROL_INITIALISED_AND_CONNECTED if unsuccessful	
Example Usage	<code>GetCurrentAcceleration(hTiltControl, &dAcceleration, true, 8)</code>	
Notes		

GetCurrentAcceleration

Prototype	<code>int GetCurrentAngle(HANDLE hTiltControl, double * dCurrentAngle, bool bIsHorizontal, nDigitalFilterLength);</code>	
Description	GetCurrentAngle will return the Horizontal or Vertical angle of tilt via the dCurrentAngle variable. If bIsHorizontal == true then the horizontal tilt reading will be returned else the vertical tilt reading will be returned.	
Inputs	hTiltControl:	Handle to TiltCONTROL
	*dCurrentAngle:	Pointer to the dCurrentAngle variable
	bIsHorizontal:	Set true if you require the horizontal tilt
	nDigitalFilterLength:	The length of the digital filter applied to the raw data received from the accelerometer in the TiltCONTROL. The larger nDigitalFilterLength is, the more stable the sensor readings will be. The smaller nDigitalFilterLength is the more responsive the the sensor readings will be.
Outputs	Pointer to the dCurrentAngle variable with result.	
Return Values	The current status of the TiltCONTROL. TILTCONTROL_INITIALISED_AND_CONNECTED if successful The current TiltControl Status, other than TILTCONTROL_INITIALISED_AND_CONNECTED if unsuccessful	
Example Usage	<code>GetCurrentAngle(hTiltControl, &dCurrentAngle, true, 8)</code>	
Notes		

GetTiltControlStatus

Prototype	<code>int GetTiltControlStatus(HANDLE hTiltControl);</code>	
Description	GetTiltControlStatus will return the status of the current TiltCONTROL connection	
Inputs	hTiltControl:	Handle to TiltCONTROL
Outputs	None.	
Return Values	The status of the current TiltCONTROL connection. This could be one of the following values TILTCONTROL_NOT_INITIALISED TILTCONTROL_SYNCHRONISING TILTCONTROL_INITIALISED_AND_CONNECTED TILTCONTROL_NOT_CONNECTED TILTCONTROL_DESTROYED	
Example Usage	<code>GetTiltControlStatus(hTiltCONTROL);</code>	
Notes		

ConvertTiltDataMessageToTiltSensorData

Prototype	<code>int ConvertTiltDataMessageToTiltSensorData(WPARAM wParam, TILTSENSORDATA * TiltSensorData);</code>	
Description	ConvertTiltDataMessageToTiltSensorData will convert the WPARAM from the WM_TILTDATA message, from the message format to the format of the TILTSENSORDATA structure.	
Inputs	wparam:	WPARAM from the WM_TILTDATA message.
Outputs	TiltSensorData:	TILTSENSORDATA containing the converted data.
Return Values	NULL if unsuccessful.	
Example Usage	<pre> BOOL CALLBACK TiltCONTROLHandlerDlgProc (HWND hWnd, UINT wParam, LPARAM lParam) { TILTSENSORDATA TiltSensorData; TCHAR szHorizontalAngle[512]; TCHAR szVerticalAngle[512]; switch (wParam) { case WM_TILTDATA: ConvertTiltDataMessageToTiltSensorData(wParam, &TiltSensorData); wsprintf(szHorizontalAngle, L"%.2f", TiltSensorData.dHorizontalAngleOfTilt); wsprintf(szVerticalAngle, L"%.2f", TiltSensorData.dVerticalAngleOfTilt); SetDlgItemText(hWnd, IDD_HORIZONTALANGLE, szHorizontalAngle); SetDlgItemText(hWnd, IDD_VERTICALANGLE, szVerticalAngle); SetDlgItemText(hWnd, IDD_TILTCONTROLSTATUS, GetStringRepresentationOfStatus(LOWORD(lParam))); break; } } </pre>	
Notes		

GetStringRepresentationOfStatus

Prototype	TCHAR * GetStringRepresentationOfStatus (int nStatus);
Description	GetStringRepresentationOfStatus will return the string representation of the status nStatus.
Inputs	nStatus: Status constant.
Outputs	
Return Values	The string representation of the status.
Example Usage	<pre> BOOL CALLBACK TiltCONTROLHandlerDlgProc (HWND hWnd, UINT wParam, WPARAM wParam, LPARAM lParam) { TILTSENSORDATA TiltSensorData; TCHAR szHorizontalAngle[512]; TCHAR szVerticalAngle[512]; switch (wParam) { case WM_TILTDATA: ConvertTiltDataMessageToTiltSensorData(wParam, &TiltSensorData); wsprintf(szHorizontalAngle, L"%2f", TiltSensorData.dHorizontalAngleOfTilt); wsprintf(szVerticalAngle, L"%2f", TiltSensorData.dVerticalAngleOfTilt); SetDlgItemText(hWnd, IDD_HORIZONTALANGLE, szHorizontalAngle); SetDlgItemText(hWnd, IDD_VERTICALANGLE, szVerticalAngle); SetDlgItemText(hWnd, IDD_TILTCONTROLSTATUS, GetStringRepresentationOfStatus(LOWORD(lParam))); break; } } </pre>
Notes	

CalculateAngleOfRotation

Prototype	double CalculateAngleOfRotation(double dHorizontalAngle, double dVerticalAngle)	
Description	<p>CalculateAngleOfRotation(double dHorizontalAngle, double dVerticalAngle) will return the calculated angle of rotation of the TiltCONTROL based on the Horizontal and Vertical angle of tilt of the device. If the PDA that the TiltCONTROL is connected to is held with the right side parallel to the ground then the Angle of rotation will be 0 degrees. If it is tilted to the left 90 degrees so that it is upright then the Angle of Rotation will be 270 degrees. If it is tilted to the left 90 degrees from the upright position the Angle of Rotation will be 180 degrees and so on. The figure below demonstrates the concept.</p> <div data-bbox="507 770 1307 1075" data-label="Image"> <p>The diagram shows four PDA devices in different orientations, each labeled with an angle: 0 Deg, 270 Deg, 180 Deg, and 90 Deg. The 0 Deg device is held with its right side parallel to the ground. The 270 Deg device is upright. The 180 Deg device is upside down. The 90 Deg device is upside down and rotated.</p> </div>	
Inputs	dHorizontalAngle:	Horizontal Angle of tilt
	dVerticalAngle:	Vertical Angle of tilt
Outputs		
Return Values	Angle Of Rotation	
Example Usage	<pre> BOOL CALLBACK TiltCONTROLHandlerDlgProc (HWND hWnd, UINT wParam, WPARAM wParam, LPARAM lParam) { TILTSENSORDATA TiltSensorData; TCHAR szHorizontalAngle[512]; TCHAR szVerticalAngle[512]; switch (wParam) { case WM_TILTDATA: ConvertTiltDataMessageToTiltSensorData(wParam, &TiltSensorData); wsprintf(szHorizontalAngle, L"%.2f", TiltSensorData.dHorizontalAngleOfTilt); wsprintf(szVerticalAngle, L"%.2f", TiltSensorData.dVerticalAngleOfTilt); SetDlgItemText(hWnd, IDD_HORIZONTALANGLE, szHorizontalAngle); SetDlgItemText(hWnd, IDD_VERTICALANGLE, szVerticalAngle); SetDlgItemText(hWnd, IDD_TILTCONTROLSTATUS, dAngleOfRotation = CalculateAngleOfRotation(TiltSensorData.dHorizontalAngleOfTilt, TiltSensorData.dVerticalAngleOfTilt); break; } } </pre>	
Notes		

Messages

WM_TILTDATA message

The WM_TILTDATA message takes the form

```
WM_TILTDATA
wHorizontalTiltData = LOWORD(wParam);
wVerticalTiltData   = HIWORD(wParam);
nTiltControlStatus = (int) lParam;
```

wHorizontalTiltData represents the left to right tilt of the TiltCONTROL.
wVerticalTiltData represents the up and down tilt of the TiltCONTROL.
nTiltControlStatus represents the current status of the TiltCONTROL, and can be one of the following values

```
TILTCONTROL_NOT_INITIALISED = 0
TILTCONTROL_SYNCHRONISING = 1
TILTCONTROL_INITIALISED_AND_CONNECTED = 2
TILTCONTROL_NOT_CONNECTED = 3
TILTCONTROL_DESTROYED = 0
```

The actual horizontal and vertical angle of tilt can be calculated using the following formula:

```
Horizontal Angle
    of tilt = ((wHorizontalTiltData / 65535) * 2 - 1) * 90;

Vertical Angle
    of tilt = ((wVerticalTiltData / 65535) * 2 - 1) * 90;
```

It is not necessary to know how to calculate the angle of tilt, as you can call the function `ConvertTiltDataMessageToTiltSensorData()` contained in `TiltControlMessages.h` to convert the message data to the more usable data contained in the `TILTSENSORDATA` structure.

`ConvertTiltDataMessageToTiltSensorData(WPARAM wParam, TILTSENSORDATA * TiltSensorData)` takes the `wParam` received within the `WM_TILTDATA` message as its first parameter and returns the current TiltCONTROL data in a `TILTSENSORDATA` structure via the second parameter.

The `TILTSENSORDATA` structure is defined in `TiltControlDefinitions.h` and is defined as

```
typedef struct
{
    double dHorizontalAccerleration; // The left to right acceleration
                                        // in G's (m/s/s)
    double dVerticalAccerleration; // The up and down acceleration in
                                        // G's (m/s/s)
    double dHorizontalAngleOfTilt; // The left to right angle of tilt
                                        // in degrees
    double dVerticalAngleOfTilt; // The up and down angle of tilt in
                                        // degrees
}TILTSENSORDATA;
```

Structures

TILTSENSORDATA

The TILTSENSORDATA structure is defined in TiltControlDefinitions.h and is defined as

```
typedef struct
{
    double dHorizontalAccerleration;    // The left to right acceleration
                                        // in G's (m/s/s)
    double dVerticalAccerleration;     // The up and down acceleration in
                                        // G's (m/s/s)
    double dHorizontalAngleOfTilt;     // The left to right angle of tilt
                                        // in degrees
    double dVerticalAngleOfTilt;       // The up and down angle of tilt in
                                        // degrees
}TILTSENSORDATA;
```

TILTCONTROLSTATE

The enumerated type TILTCONTROLSTATE represents the possible states of the TiltCONTROL. The TiltCONTROL can only be in one of the following states at any one time, except for the states TILTCONTROL_NOT_INITIALISED and TILTCONTROL_DESTROYED. These states have exactly the same properties thus have the same value to represent them.

```
typedef enum
{
    TILTCONTROL_NOT_INITIALISED = 0,
    TILTCONTROL_SYNCHRONISING,
    TILTCONTROL_INITIALISED_AND_CONNECTED,
    TILTCONTROL_NOT_CONNECTED,
    TILTCONTROL_DESTROYED = 0
}TILTCONTROLSTATE;
```

Distributing the TiltCONTROL Drivers with your application

The TiltCONTROL Drivers can be distributed with your application by installing the TiltCONTROL.PPC.CAB on the target Pocket PC, which can be downloaded from the downloads section at www.ecertech.com/downloads.aspx.