Software delivery

In this environment, UPDD is supplied as a number of separate components. Software sent via email will be held in the file ZIP file TBUPDDCE.ZIP. This avoids the rejection by many mail servers of .exe files. Touch-Base utilises virus detection software on all of our systems but recipients of the software should pass the files through their own virus checking software before proceeding with installation.

As of October 2006 we are only shipping UPDD CE V4 unless otherwise requested to supply V3 and these note refer only to the V4 driver.

Overview

For OEMs requiring a touch screen, or other pointer interface on Windows CE devices, the Touch-Base Universal Pointer Device Driver suite of software includes a Windows CE 2.xx, 3.x, 4.x/.NET, 5,x and 6.x embedded driver. This driver utilises the same code base as the Windows NT/9x/2K/XP UPDD product and so has all facilities found in UPDD on those systems, except for minor differences to accommodate variance in the Windows CE implementation. The UPDD application program interface is supported on Windows CE allowing 3rd party utilities to be developed.

Hardware Interfaces

Serial should work in all target platforms.

PS/2 has been tested in X86 only.

USB have been tested on X86, Strongarm (Intel XScale PXA-255) and MIPS platforms. For other processors we would need to be supplied a target system to modify the UPDD USB interface to work with the processor.

Other interfaces such as ISA etc could be added if required.

See important Port Interface issues below.

Processor support

The driver has been tested with X86, StrongArm and MIPS processors running CE 3.0 .NET 4.1, 4.2, 5.0 and 6.0. Our CE.NET driver has been built in a CE.NET 4.2 development environment. We can build drivers for other processors, as supported by the Microsoft CE Platform Builder, on request. The CE 5.0 and 6.0 drivers are incompatible with previous versions of Windows CE so the appropriate driver for your target platform should be used.

At the time of writing, the processors supported by Win CE 2.x/3.x/4.x (Net)/5.x/6.x are:

- MIPS
- StrongArm (also supports ARM processors)
- Power PC (Win CE 3.0/2.xx only)
- Hitachi SH3 (CE 4.x and earlier) / SH4
- x86

A complete list of supported hardware can be found at:

http://msdn.microsoft.com/en-us/embedded/aa714536.aspx

Any processor can be supported as long as a Board Support Package is available for the target processor.
Read
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcehardware5/html/wce50conSupportedBoardSupportPackages.asp for more information on BSP's

Target hardware may have to be supplied for testing if any problems are experienced with the driver.

Embedding UPDD for Win CE

It is assumed that the developer is using Platform Builder 3.0 or greater and has created a Win CE image for the target hardware. This image should be created with the MAXALL configuration under CE 3.0 or an appropriate platform configuration under CE.NET.

To embed UPDD in a Windows CE image follow the following steps.

1. Copy the files (tbcalib.exe, tbupddce.dll, tbupddceusb.dll (if using a USB touch controller) and tbcalibdll.dll (if it exists) from the supplied zip file to the project folder in which your project files are stored.
2. Edit the configuration file project.bib, adding the following line in the MODULES section:
   tbupddce.dll $(_FLATRELEASEDIR)¥tbupddce.dll NK SH or NK SK for CE 6.0
3. If USB support is required edit the configuration file project.bib, adding the following line in the MODULES section:
   tbupddceusb.dll $(_FLATRELEASEDIR)¥tbupddceusb.dll NK SH or NK SK for CE 6.0
4. If manual calibration facilities are required, edit the configuration file project.bib, adding the following line in the MODULES section:
   tbcalib.exe $(_FLATRELEASEDIR)¥tbcalib.exe NK
   tbcalibdll.dll $(_FLATRELEASEDIR)¥tbcalibdll.dll NK (if using CE 6.0)
5. Edit the configuration file project.reg, adding the contents of the supplied file tbupddce.reg and any other .reg files supplied as part of the UPDD CE package.
6. If USB support is required edit the configuration file project.reg, manually adding the following entries:
   [HKEY_LOCAL_MACHINE¥Drivers¥USB¥LoadClients¥VID_PID¥Default¥Default¥Tbupddceusb]

"DLL"="Tbupddceusb.dll"
"Order"=dword:1
"Index"=dword:1
"Prefix"="USB"
with the VID and PID tokens being replaced with the appropriate values in base 10 (decimal). The VID and PID values can be found in the supplied tbupddce.reg in hexadecimal format.

“Product Id”=dword:0000HHHH

“Vendor Id”=dword:0000HHHH

Important note 1 - The VID and PID values MUST match the VID and PID of the controller in use. If the supplied files do not match the controllers VID and PID, manually edit the values to match. If necessary, we can supply a Win 2000 utility to display the VID and PID when the controller is plugged in. e.g. Controller has hex VID = **1234** and PID = **11** values. Hex 1234 = decimal **4660** and Hex **11** = decimal **17**.

Based on the above controller the settings would be as follows:

project.reg

[HKEY_LOCAL_MACHINE¥Drivers¥USB¥LoadClients¥**4660**_**17**¥Default¥Default¥Tbupddceusb]

tbupddce.reg

“Product Id”=dword:00000**11**

“Vendor Id”=dword:0000**1234**

7. If EEPROM calibration data retrieval is required configure EEPROM retrieval to be invoked at startup. See EEPROM notes below.
8. Make any required software changes to the system components. See “Port interface issues” below.
9. Rebuild the image using the Platform Builder.

Calibration issues

The touch screen interface requires that calibration data be used to map screen touches to the corresponding Windows display area. This data is held in the registry or the touch controller's EEprom. UPDD for Windows CE supports EEPROM based calibration storage where the device itself has the appropriate support and where support has been added in UPDD. Not all EEprom enabled controllers are supported by UPDD for EEprom calibration storage as support has been added on a per requirement basis. Should you require EEprom support, please contact Touch-Base to discuss further. Unless EEprom calibration is preconfigured in the supplied UPDD CE driver, calibration storage will be in the registry.

If calibration data id held in the registry and the registry is not held in non volatile memory then the registry is reset to its default values whenever the device is reset. This raises some calibration issues that must be considered, these are especially relevant to x86 implementations where the device is reset upon suspend or power off.

UPDD calibration information can be determined in one of 3 ways:

Auto-calibration. The calibration information is calculated based on the maximum theoretical range of values (from the number of bits in the touch data packet) assuming that the available touch area is exactly the same size as the visible desktop area.

Pre-calibration. The calibration data is determined for a device – or class of device, and stored in the embedded configuration.

Manual Calibration. The user executes the calibration program and touches a series of displayed points on the screen. Data recorded in this way may be lost when the device is reset (if the registry is not held in persistent memory) unless the data is stored in the touch controller's EEPROM. An OEM using manual calibration, where the calibration data is lost over a reboot, needs to decide on a strategy for initiating the manual calibration. E.g. executing the calibration program at start-up, or placing an icon on the desktop. These and other options are implemented via the platform configuration.

Calibration Settings

The supplied registry settings will usually be set to indicate auto-calibration. A CE image developer might wish to alter these default settings, for example to provide a pre-calibrated device. This is achieved by changing the settings copied to project.reg. The settings that affect calibration are stored under:

HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{…}¥1 – Where {…} is a GUID that identifies the package.

The specific values used are: –

Number Of Calibration Points

RefX0, RefX1, RefY0, RefY1 *

CalX0, CalX1, CalY0, CalY1*

InvertX, InvertY, SwapXY

*If "Number Of Calibration Points" is greater than 2 then there will be correspondingly more of these values.

The above values may be changed in the following ways.

- **To force auto-calibration**

  Number of Calibration Points=2

  CalX0=CalX1=CalY0=CalY1=0

  InvertX,InvertY,SwapXY=? (These values will have to be determined by experiment for a specific device. The valid values are 0 and 1 (for true / false)). Tip – running the calibration program with 3 or more calibration points will automatically set these values which can then be noted in the project.reg file.

- **To define pre-calibration data**

  The easiest way to determine the pre-calibration data is to run the calibration program on the device and perform a manual calibration. The calibration values, detailed above, are noted from the registry on the target device and manually entered in to project.reg. NB when executing tbcalib.exe the mode of calibration is taken from a different registry location, see "manual calibration" below.

- **To prepare for manual calibration:**

  A CE image developer might wish to alter aspects of the manual calibration, such as the number of points or the location of the points. In doing this it is important to bear in mind that the values outlined above are used for the active data (i.e. the previous calibration), and the data used for the next manual calibration are located at: -

  HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{guid}¥1¥Calibration Styles¥0

- **Determining calibration reference points:**

  If defining pre-calibration data, or preparing for a manual calibration with a non-default number of points, the reference values must be set. The reference values represent the location of the calibration points based on a grid 0-65535 in the x and y planes, with the origin at the top left. So for a three-point calibration with points

  At top left middle and bottom right the values would be: -

  RefX0=0
  RefY0=0
  RefX1=32768
  RefY1=32768
  RefX2=65535
  RefY2=65535

- **EEPROM storage**

  If the controller in use supports EEPROM storage **and** it is supported by UPDD and EEprom calibration storage is required then registry entry

  HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{guid}¥1¥EEPROM Calibration

  should be enabled.

  Important note: If calibration data is stored in the controller then it is necessary to automatically call the calibration program at system startup to retrieve the calibration data passing the eeprom retrieval parameter, i.e. **tbcalib eeprom**. See Calibration documentation for further details.

Calibration beeps

A system beep will be made as each calibration point is accepted if the calibration beep setting is enabled

HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{…}¥Calibration Beeps

Calibration Procedure

Calibration can be invoked by calling the calibration program, **TBcalib**. This program can be invoked to calibrate the main video calibration, calibrate toolbars and retrieve stored calibration data from EEPROM and set driver settings. For more information see the separate Calibration documentation.

Toolbars

A toolbar is an area on the touch screen that acts independently from the main calibrated video area. A toolbar can simply be used to mask off areas of the calibrated video area or they can be used to trigger an event. Toolbar utilisation is described in the Toolbar document.

Driver settings

The supplied registry settings will usually be set to the default settings for the controller in use. In Win CE these settings are set manually in the tbupddce.reg file prior to generating the CE build. We trust the entries in the .reg structure are self-explanatory and are found in the registry branch HKEY_LOCAL_MACHINE ¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{…}¥1

Given that all settings for an embedded system are defined in the CE Platform Builder and the image then embedded on the target image it has been decided that there is no need for a UPDD CE GUI as the same Windows GUI (UPDD Console) can be used to define the required settings under Windows which are then placed in the UPDD CE configuration file. In most cases the supplied configuration files will contain the required UPDD settings. Values can be manually edited in the tbupddce.reg file or changed on a Windows system using the UPDD Console.

If values are changed on a Windows system, export the UPDD registry entry (e.g. by using regedit's export option) and embed these setting in the tbupddce.reg file.

Export the complete hive

HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥TBUPDD

and take the contents of the exported file and replace all occurrences of

[HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥

with

[HKEY_LOCAL_MACHINE¥Drivers¥BuiltIn¥

If only updating individual branches, e.g. HKEY_LOCAL_MACHINE¥SYSTEM¥CurrentControlSet¥Services¥TBUPDD¥Parameters¥{guid}¥N¥Toolbars, take care to check that the guid value in the edited item matches that in the earlier entries in tbupddce.reg (it may differ from the exported value).

Sound

The sound setting requests that a sound be generated ‘on contact’ that is when Pen down is generated. Given that Windows CE does not support access to the system speaker in UPDD version 4 we play the Windows "beep" when the sound should be made. With this technique there is no control over the length or pitch of the sound.

HKLM¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{…}¥1¥Sound

HKLM¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{…}¥1¥Sound Duration – not used

HKLM¥Drivers¥BuiltIn¥TBUPDD¥Parameters¥{…}¥1¥Sound Pitch – not used

Mouse Emulation

The mouse emulation modes dictate the manner in which the touch screen emulates pointer movement and mouse click sequences and are defined across a number of settings:

event alt mode **n** = Secondary Click Mode name (e.g.TouchDown Right)

event bind **n** = Bind name (e.g DFLT)

event mode **n** =Primary Click Mode name (e.g. TouchDown Left)

event name **n** = Event Name (e.g. Default)

where **n** is the touch event number.

Please refer to the Mouse Emulation documentation for further details.

Port interface issues – important

Serial

Touch-screens may be connected to a CE device via a standard serial (COM). A CE image builder should bear in mind that the default CE image generated by platform builder might well make assumptions regarding the usage of such ports. E.g. debug output will be sent to the first physical COM port, preventing its use. By default, CE creates two com port instances, Com 0 and Com 1. Com 0 is used as the debug port and relates to the physical port com 1. Com 1 is therefore the first port that can be used by the touch screen that actually relates to the physical Com 2 port.

Many customers have been unable to get their touch screens working with a default CE build until they have plugged the touch screen into com2 or changed the BIOS so that serial port is referenced as Com2, although the references in the CE build refer to com 1 !!!!

In some circumstances the CE builder will need to amend the CE configuration to alter the default serial port behavior. If you are not familiar with this procedure we have technical bulletin that covers this subject.

USB

The CE image must be amended to support the USB host controller (this is the system's USB host controller and NOT the USB touch controller). Consult the manufacturer's documentation and or Platform Builder help for details of how to achieve this with the particular model of hardware in use. A 3rd party driver might be required for the host controller, although this has not the case for the hardware we have tested so far. To ensure the CE system's USB host controller is functioning use a HID mouse prior to testing the USB touch controller.

PS/2

Windows CE version 5 implements a new composite PS/2 keyboard and mouse driver. In its attempt to initialise the PS/2 mouse some PS/2 touch controllers can be initialised incorrectly. This can be solved by executing '**tbcalib /reinit**' during Windows startup.

We have seen systems where it is not possible to use a PS/2 keyboard when using a PS/2 touchscreen using the alternative PS/2 port. At the end of calibration the PS/2 keyboard is no longer functioning. We have not investigated this further at this time.

Software requirements

In order to build a Win CE binary image the appropriate MS Platform Builder relating to the CE environment is required. E.g. MS Platform Builder 3.0 for Win CE 3.0 builds or Window CE.NET Platform for Windows CE.NET etc.

Hardware requirements

A target Win CE device is required.

Alternatively x86 based images can be run on a standard PC using the LOADCEPC utility.

Important note. If you wish to use a Serial connection to the Windows CE device, please note that there is a consideration that is not always made clear in the Microsoft documentation. A null modem cable is required, but this differs from a standard null modem cable in that the RI pin is connected straight through. Without this connection it will be impossible to make a serial connection from the NT host to the Win CE device

Revision 1.15, 5th Apr 2008