Toradex
Swiss. Embedded. Computing.

Capacitive Multi-Touch Solution

Multi-Touch Hardware Adaption Source Code for PolyTouch™ / EPxxxxM06

Revision History

Date	Doc. Rev.	Program Version	Changes
10-Mar-14	Rev. 1.0		Initial Version
17-Apr-14	Rev. 1.1		Review
9-Mai-14	Rev. 1.2		Changed Title, Header. Add description of the touch structure

Contents

1. Introduction 3

2. General Functionality 3

3. General Functionality of the Hardware Adaption 4

4. Description of the Source Code 4

4.1. IDE 4

4.2. Specific project files 5

5. Source Code 5

5.1. Start Sequence 5

5.2. Main Loop 6

5.3. Loop to read out the touch position 6

6. Touch event type used by DeviceIoControl () 7

6.1. Header File "unfd_multitchdrv.h" 7

6.2. Touch event type 7

6.3. Type TCHINPUT (CETOUCHINPUT) 7

Reference Documents

For detailed technical information, please refer to the documents listed below.

[1] **Capacitive Multi-Touch Solution, General Functionality**
This document can be found on our website
http://developer.toradex.com/knowledge-base/capacitive-multi-touch-solution
see "Documents, General Functionality", (Toradex_MultiTch_Solution.pdf)

[2] **Capacitive Multi-Touch Solution, Unified Multi-Touch Driver**
Description of the Unified Driver for EPxxxxM06
http://developer.toradex.com/knowledge-base/capacitive-multi-touch-solution
see "Documents, Unified Multi-Touch Driver" , (Toradex_UnfdMutiTchDrv.pdf)

[3] **CETOUCHINPUT (Compact 2013)**
Microsoft Developer Network, Touch event structure CETOUCHINPUT.
http://msdn.microsoft.com/en-us/library/gg157266.aspx
See "Touch Driver Structure", "CETOUCHINPUT"

1. Introduction

This document describes the source code of the Hardware Adaption EPxxxxM06 example.

2. General Functionality

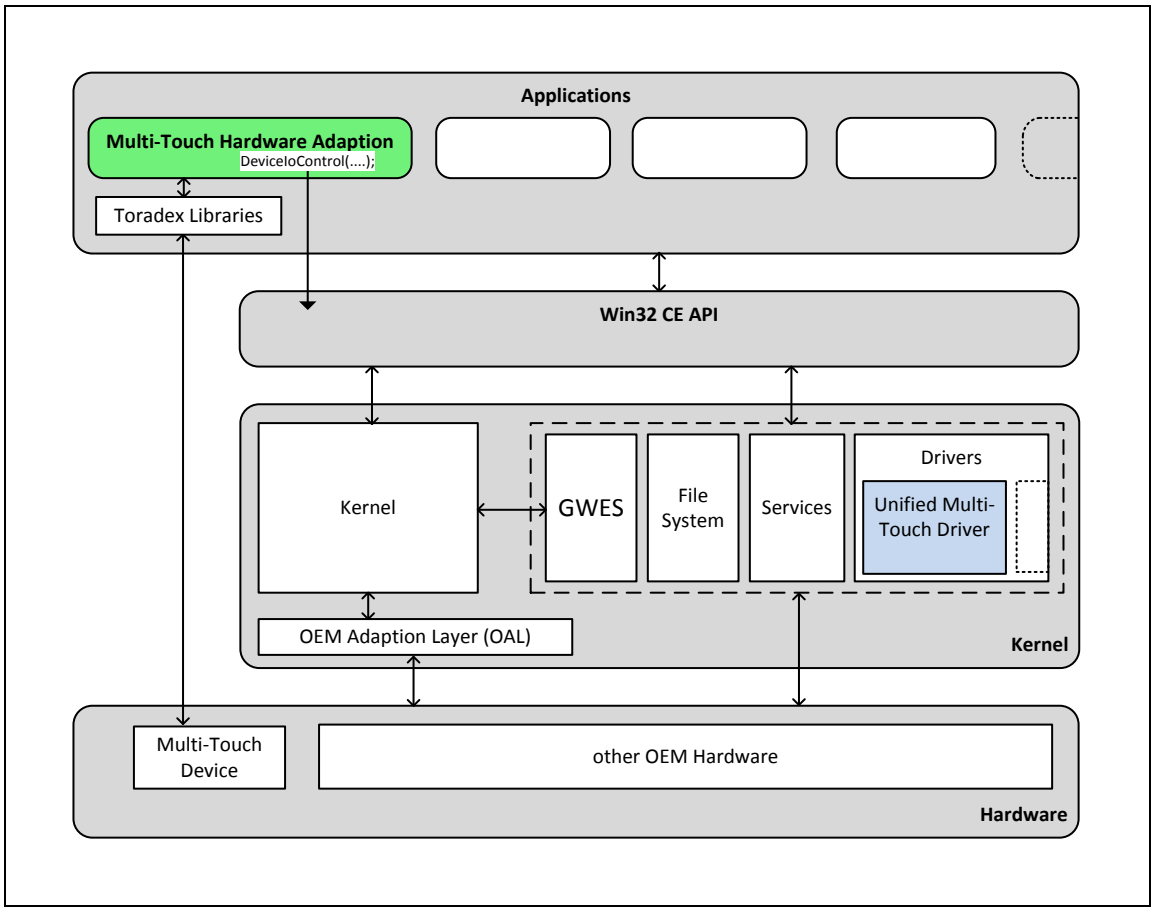

Figure 1: Overview of Capacitive Multi-Touch Solution

The overview of the whole "Capacitive Multi-Touch Solution" is described in "Capacitive Multi-Touch Solution, General Functionality" (see [1]).

3. General Functionality of the Hardware Adaption

The “Multi-Touch Hardware Adaption” runs as an application. After booting, the application sets up the I/O signals (e.g. reset, interrupt etc.), establishes a communication with the “Multi-Touch Device” and sends the necessary telegrams to initialize the touch device.

Additionally, the communication channel to the “Unified Multi-Touch Driver” is open.

After that, the application waits for a touch event from the touch device. If this happens, the application gets the status and the touch position(s) from the touch device and converts it to the “TCHINPUT” data block and sends it as an IOCTL call to the “Unified Multi-Touch Driver”.

How often the touch events are reported depends on the application (polling), or the “Multi-Touch Device” itself, or a combination of both.

Another feature of this application allows it to test the communication to the touch device and start a recovery sequence if necessary.

To store some settings in the registry gives the possibility to write a more generic “Multi-Touch Hardware Adaption”.

4. Description of the Source Code

This chapter gives some additional hints to the comments in the source code.

The source code can be downloaded from

http://developer.toradex.com/product-selector/capacitive-multi-touch-solution

as a zip-file (Src_HwAdapt_EPxxxxM06_rx.zip)

see:

‘Source Code Example “Hardware Adaption”’

4.1. IDE

The example is a Visual Studio 2008 Project with installed Toradex SDK:

“Windows Embedded Compact 7 SDK”

It can be found at:

http://developer.toradex.com/software-resources/arm-family/windows-ce/development-tools

The following Toradex libraries are used:

intlib.lib

kernelcallbacklib.lib

commctrl.lib

i2clib.lib

coredll.lib

These libraries can be found at:

http://developer.toradex.com/knowledge-base/windows-ce-libraries-and-code-samples

Probably the search path for the library h-files and lib-files must be adapted in the project.

4.2. Specific project files

The Hardware Adaption project contains the following specific project files:

HwAdapt.c	// contains the main loop
TchContr_EP0700.c	// contains all function for the handling of the
TchContr_EP0700.h	// EPxxxx touch controller
unfd_multitchdrv.h	// contains the common data type "TCHINPUT"

5. Source Code

The following chapters describe the source code of the file "HwAdapt.c". The called functions are defined in "TchContr_EP0700.c"

5.1. Start Sequence

After starting the application, the registry settings are read.

```
// Read the and setup the configuration
vI2C_ReadConfiguration();
```

Then the I2C communication and the GPIO for reset and interrupt are setup.

```
// Init the IC interface on the Colibri (makes no sense to retry)
if (FALSE==bI2C_Init())
```

Then the communication to the Unified Multi-Touch Driver is established.

```
// Get a handle to the driver. Which causes the driver's XXX_Open function to be called
hDS = CreateFile( TEXT("TCH1:"), GENERIC_READ|GENERIC_WRITE,
FILE_SHARE_READ|FILE_SHARE_WRITE, NULL, OPEN_EXISTING, 0, 0);
```

5.2. Main Loop

The main loop contains the initialization of the touch controller.

```
if (FALSE==bInitEPxxxxM06())          // reset and start touch controller
```

If this is ok, then the program enters in another loop (see chapter 5.3) which reads out the status and touch position(s).

```
// wait for messages
while(1==1)
```

Otherwise the touch controller will be reset, and after a delay, the initialization of the touch controller will happen again.

```
vDeInitEPxxxxM06();                   // stop (reset/powerdown) the Touch Controller
Sleep(400);                           // wait before an new try starts
```

This is done for ever

5.3. Loop to read out the touch position

This loop waits until an interrupt from the touch controller occurs or a timeout happens.

```
if (FALSE==bWaitTouchDataINT())
```

If a timeout occurs, then the communication is tested.

```
// timeout, test the communication to the touch controller
if (FALSE==bCommunicationTestEPxxxxM06())
```

Otherwise the status and the touch position(s) are read out of the touch controller.

```
// get the touch events
vTouchPanelGetPoint(&CeTchInp.TchInp[0],&CeTchInp.iCnt);
```

If the touch event is valid, then it sends an IOCLT code “IOCTL_SET_TOUCH_EVENT” to the Unified Multi-Touch Driver.

```
// send the touch events to the unified touch driver
DeviceIoControl( hDS, IOCTL_SET_TOUCH_EVENT, &CeTchInp, sizeof(CeTchInp), NULL, 0,
&BytesHandled, NULL );
```

Any error causes the exit of these loops, and the main loop resets and initializes the touch controller again.

6. Touch event type used by DeviceIoControl ()

6.1. Header File “unfd_multitchdrv.h”

The header file "tchddi.h" is part of the BSP and not available for application normally. Because of the lack of this header file the types CETOUCHINPUT and PCETOUCHINPUT are missing.

The header file “unfd_multitchdrv.h” contains a copy of these two types. To avoid type conflicts TCHINPUT, PTCHINPUT are defined and used instead CETOUCHINPUT and PCETOUCHINPUT.

Please note: TCHINPUT,PTCHINPUT must be identical to TCHINPUT,PCETOUCHINPUT.

6.2. Touch event type

To send data with function DeviceIoControl() all data must be packed in one structure.

```
typedef struct
{
        TCHINPUT        TchInp[MAXTOUCHES];
        INT             iCnt;
}CETCHINP_IOCTL;
```

TchInp[MAXTOUCHES];	An array of touch data of the type TCHINPUT (CETOUCHINPUT)
iCnt	Actually count of touches saved in the TchInp[].

6.3. Type TCHINPUT (CETOUCHINPUT)

The following chapter describes the minimum data which must be defined for a touch event. More information can be found at [3].

typedef struct {

LONG x;	Specifies the x coordinate of the touch point in 4ths of a pixel.
LONG y;	Specifies the y coordinate of the touch point in 4ths of a pixel.
HANDLE hSource;	Touch point identifier set to zero, it is set by the Unified Multi-Touch Driver before the touch data is send to the MDD.
DWORD dwID;	Maintained for a contact from the time it goes down until the time it goes up. Must be a value bigger the zero and different from other contacts (fingers).

DWORD dwFlags; A set of bit flags that specify various aspects of touch point press / release and motion. Valid flags are:

TOUCHEVENTF_DOWN | TOUCHEVENTF_INRANGE	A finger contacts the touch
TOUCHEVENTF_MOVE | TOUCHEVENTF_INRANGE	The finger moves touch
TOUCHEVENTF_UP	The finger is lift up

All other data must be zero or set according to the specification of Microsoft

DWORD dwMask;

DWORD dwTime;

DWORD cxContact;

DWORD cyContact;

DWORD dwPropertyOffset;

DWORD cbProperty;

} TCHINPUT, *PTCHINPUT;