Windows CE 6.0 and Embedded Compact 7

For

BCT RE2

User Guide

Document Reference: Windows CE User Guide for RE2

Document Issue: 1.03

Contents

Introduction

The content of this document provides all the necessary information required to get started with application development under Windows CE 6.0 for the RE2 platform. It covers:

- An overview of the Windows CE 6.0 Boot Process
- Peripheral support included in Windows CE 6.0
- How to install the tools necessary to develop applications that run under Windows CE 6.0
- How to start developing applications
- How to use the Hardware API functions supported under RE2

Windows CE 6.0 initialisation and booting overview

The RE2 boot process begins with the execution of a Windows CE boot loader. The boot loader is configurable over USB, and performs the following initialisation steps:

- Setup initial processor registers
- Test for configuration mode or normal Windows CE boot
- Setup display and show a custom splash screen
- Locate a Windows CE 6.0 image
- Boot Windows CE 6.0 Image

The boot loader can be used for updating Windows CE images, Splash screens and even the boot loader itself. The boot loader is also used to enable or disable peripherals, and configure the required LCD panel / DVI monitor connected to a BCT RE2. The RE2 supports booting from either onboard NAND flash, SD Card or over Ethernet using Windows KITL. Again the boot source is selectable using the boot loader configuration utility. For full details on configuring the boot loader over USB using the desktop configuration utility please see the document, "RE2 Single Board Computer User Guide".

Windows CE 6.0 follows the standard boot process except drivers are configured to dynamically load dependent on their configuration in the boot loader. The generic Windows CE image for RE2 supports the hive based registry, and uses the onboard NAND flash as the medium for storing the hives. This allows the OS to persist registry settings through a cold boot.

Booting from NAND

The Windows CE firmware for RE2 can be updated using the boot loader configuration utility or from within the Windows CE OS. Being able to update firmware from within the Windows CE OS provides a method of remotely updating operating system firmware from a customer application.

To provide a method of recovering from a failed remote update the following features are implemented:

- When Windows CE firmware is updated within the OS it is stored in NAND but in a physically different location to firmware updated using the boot loader configuration utility.
- The boot loader features an option allowing the boot of an operating system to be protected by the watchdog.
- The watchdog API provides a function allowing customer software to signal the successful boot of the operating system.

These features when combined allow the operating system firmware to be updated remotely with fallback protection in the case of a failed update. The following table indicates which OS firmware the boot loader would use based on previous boot success and firmware availability. Unless watchdog protected boot is enabled, the boot loader will always use OS loaded firmware in preference of boot loader programmed firmware.

Last boot of operating system was signalled as successful from customer software	**Operating System updated firmware Present**	**Operating system to boot**
Yes	No	Boot loader programmed firmware
No	No	Boot loader programmed firmware
Yes	Yes	OS programmed firmware
No	Yes	Boot loader programmed firmware

For details on updating Windows CE firmware from within Windows CE please see the RE2 API section of this document.

Note: If the boot loader configuration utility is used to update the Windows CE firmware any firmware that has been updated using the RE2 API will be erased.

Windows CE 6.0 Peripheral Support

The optional generic Windows CE 6.0 image included with an RE2 features support for the following on-board peripherals.

USB Host

The BCT RE2 features support for an OHCI and EHCI (USB2) compatible USB host. USB class support for HID and Mass storage devices is included in the image.

USB Device

In Windows CE the USB device port is implemented as a Microsoft ActiveSync device. Using ActiveSync 4.5 or greater, it is possible to debug and deploy applications using Visual Studio, as well as view the internal RE2 files system in an explorer style interface.

GPIO

The Windows CE GPIO driver supports up to a maximum of 12 separate pins, all configurable as either inputs or outputs.

Real Time Clock

The BCT RE2 includes a battery backed real time clock. This allows the system time to be remembered through a cold boot. Calls to either SetSystemTime() or SetLocalTime() automatically cause the new time to be saved into the battery backed clock.

Serial Ports

Two RS232 ports and one RS422 / 485 port are exposed as standard COM ports in Windows CE. Please see the following table for details of how each physical port is mapped in Windows CE.

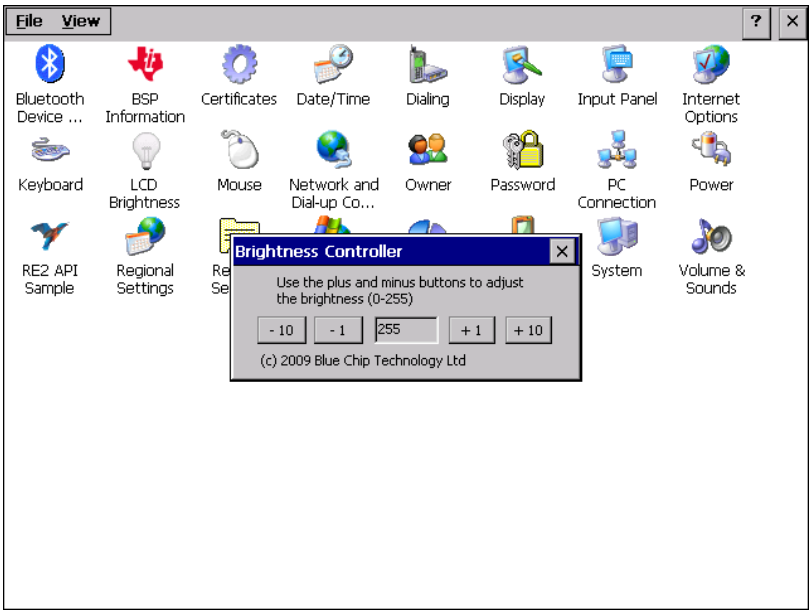
Header	Signal Type	Control Lines	Windows CE COM port
P11 RS422 / 485	RS422 / 485	No	COM1
P11 RS232	RS232	No	COM2 (When not in kernel debugging mode)
P10	RS232	Yes	COM3

COM2 has a dual purpose in Windows CE. It can be configured as either a Windows CE standard COM port available to applications or as a kernel debug port useful during OS low level development. When configured for kernel debug, COM2 is unavailable for application development and is configured for 115200 baud, 8 data bits, 1 stop bit, and no parity. Please see, “RE2 Single Board Computer User Guide” for details on configuring this port using the configuration utility.

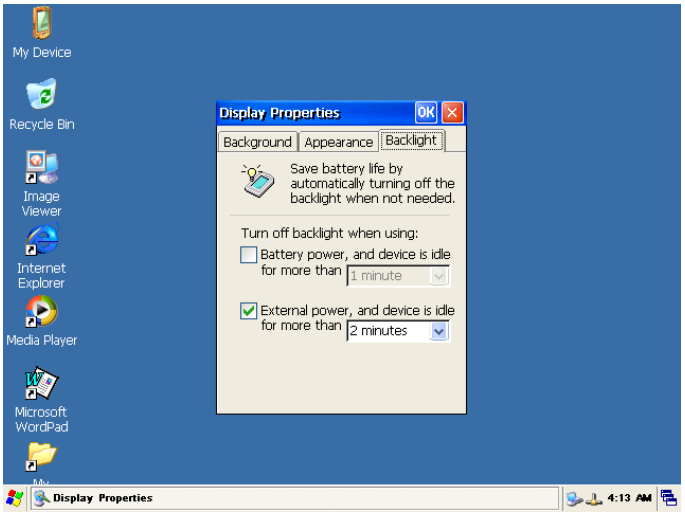
The transmit line of the RS422/485 interface is software controllable to be enabled or disabled by using the DTR control line. When DTR is enabled the TX line is enabled. When DTR is disabled the TX line is disabled.

Backlight control

A sample brightness control application is included in the Windows CE image to allow the brightness to be easily changed using the control panel in Windows Explorer. The sample application is included as source with the Windows CE SDK to demonstrate how to change the brightness using a custom application. To try the sample LCD Brightness application, navigate to the control panel and double click on 'LCD Brightness'.



The backlight is also configurable in the "Display Properties" dialog to allow the screen to be automatically dimmed after a set amount of time. This feature is useful for power saving when the device is not in use. Only the external power idle mode is implemented. In the below screen shot the device is configured to automatically dim the backlight after 2 minutes of inactivity.

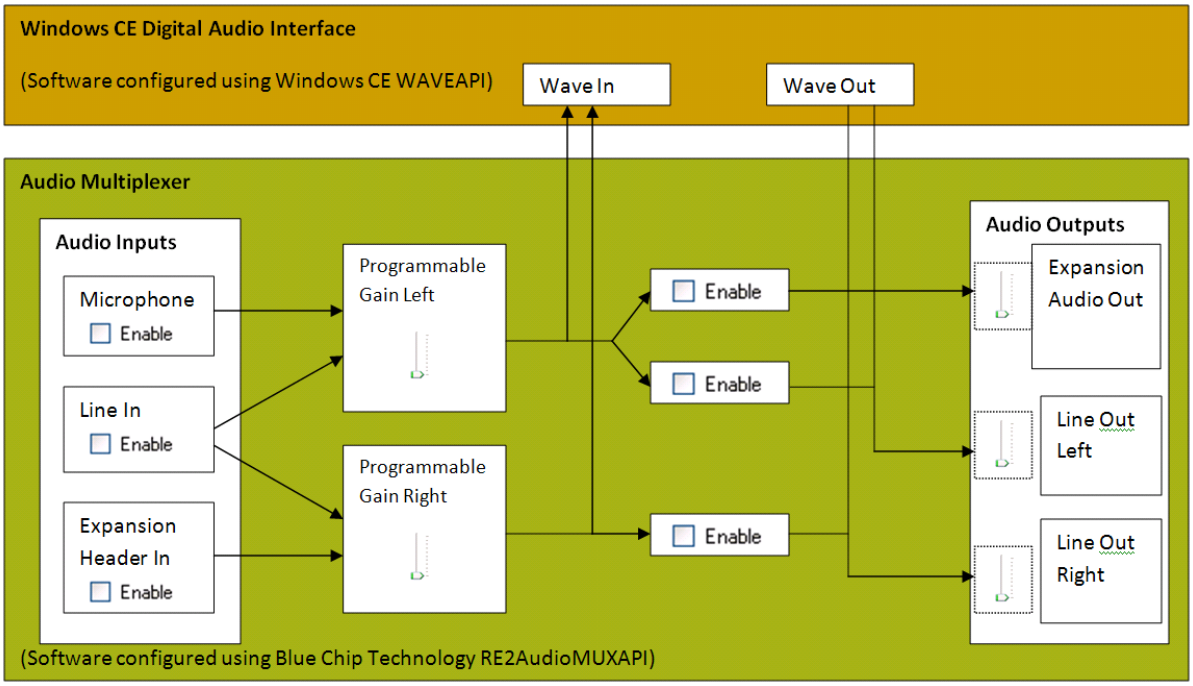


SD Card

RE2 includes a Micro SD card interface which conforms to specification version 2.0 and supports cards up to 32GB in size.

Audio

RE2 features multiple audio inputs and outputs with different routing options. The diagram below shows how audio can be routed around the RE2. Audio routings can be configured using a basic API provided with the Windows CE SDK. Please see the API definitions section of this document for details.



Bluetooth

The RE2 features an on-board Bluetooth device. Windows CE includes Bluetooth profile support for:
Bluetooth HID devices
(SYSGEN_BTH_HID_MOUSE / SYSGEN_BTH_HID_KEYBOARD),
Bluetooth HS/HF and Audio Gateway Service (SYSGEN_BTH_AG),
Bluetooth LAP and Configuration Utility (SYSGEN_BTH_GATEWAY)
Bluetooth PAN (SYSGEN_BTH_PAN).
The standard Windows CE API's should be used for communication with Bluetooth devices. Power control of the Bluetooth module is provided by a basic API included with the Windows CE SDK. Please see the API definitions section of this document for details.

WiFi

The RE2 features on-board WiFi. The Windows CE Wireless Zero Configuration Service should be used for configuring setting up wireless networks. Power control of the WiFi module is provided by a basic API included with the Windows CE SDK. Please see the API definitions section of this document for details.

VIP Camera Module

The Video Input driver is based on the TVP5146 chip featured on the VIP that acquires video signals from various multiplexed sources then sends the video stream to the CPU over the ISP parallel bus.

The Video Input driver is DirectShow compliant and supports the following acquisition sources:

- Composite 1
- Composite 2
- Composite 3
- S-Video
- Component
-

The driver exposes the following formats:

PIN	Color format	Resolution	Bits Per Pixel	Frames per Second
CAPTURE/ PREVIEW	UYVY	720x576	16 bpp	25 fps

As the camera driver is implemented as a standard DirectShow Camera it can be accessed using the DirectShow API documented at:
http://msdn.microsoft.com/en-us/library/aa919874.aspx

A sample application is distributed with the Windows CE SDK for RE2 which demonstrates how to preview and capture video with the VIP camera module. Running the VIP1.exe application without parameters shows a Camera preview of "Composite 1". Running the VIP1.exe application with various command parameters makes it possible to capture video, change the video input and also change the frame rate of video capture. At a command prompt type, "vip1.exe /?" for a list of available parameters. The table below shows some sample commands and their purpose.

Command	Purpose
vip1.exe	Displays a camera preview of Composite 1
vip1.exe /venc h264 /file capture.asf	Displays a preview of Composite 1 and captures the video data to a file called capture.asf encoded as H264 in the root of the file system.
vip1.exe /venc h264 /file capture.asf /nand	Displays a preview of Composite 1 and captures the video data to a file called capture.asf encoded as H264 in the nand flash (hard disk).

vip1.exe /venc h264 /file capture.asf /vin comp3	Displays a preview of Composite 3 and captures the video data to a file called capture.asf encoded as H264 in the root of the file system.
vip1.exe /fskip 1	Displays a camera preview of Composite 1 skipping every other frame. This will produce a frame rate of 12 fps.

Other Peripherals

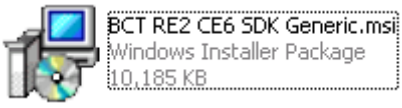
The Windows CE 6.0 has support for, 10/100 Ethernet, stylus touch screen, both of which are implemented as standard OS components.

Watchdog and I2C support is also provided in the form of API's.

Development tool installation

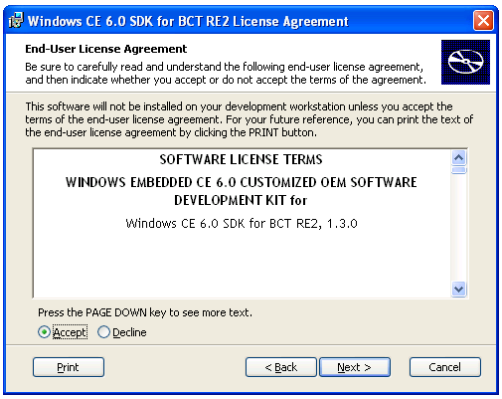
Application development targeting Windows CE 6 for RE2 requires Microsoft Visual Studio 2005 SP1, Microsoft Active sync 4.5 or greater, and the RE2 software development kit. **The version of Visual Studio 2005 chosen must support smart device development.** Ensure that Visual studio is fully installed along with active sync before following the steps below to install the BCT RE2 SDK.

1. Launch the RE2 SDK installer file from the support CD

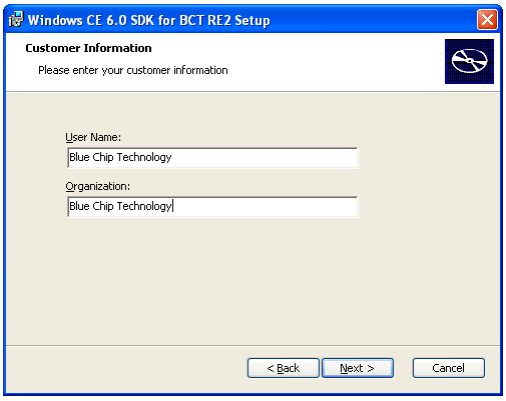


2. Click next

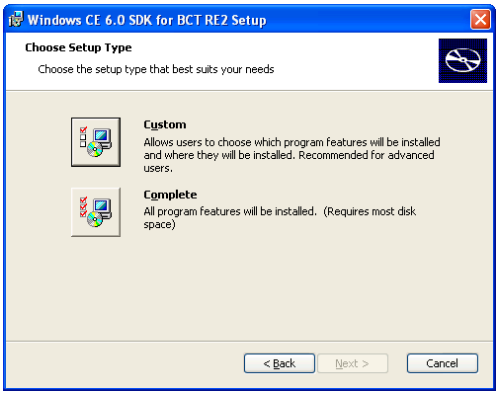
1. Accept the licence agreement and click next



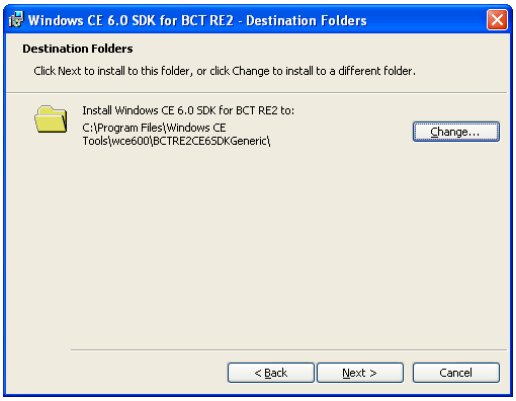
2. Enter user and company name information and click next



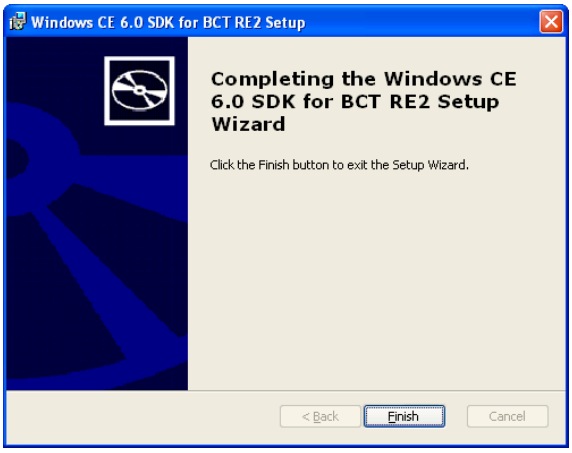
3. Choose complete installation



4. Click next



5. Click install



6. After the installation completes click the “Finish” button
7. The installation of the BCT RE2 SDK is now complete.

By default the RE2 SDK installs to location:
C:\Program Files\Windows CE Tools\wce600\BCTRE2CE6SDKGeneric.
In this location the following folders will be copied.

Folder	**Description**
Include	This folder holds all the header files required to build an application for the RE2 platform
Lib	This folder holds all the library files required to build an application for the RE2 platform
Sample_applications	This folder holds some sample applications that can be used as references while creating applications for RE2. The examples demonstrate how to interface to the RE2 hardware libraries.

Sample Applications

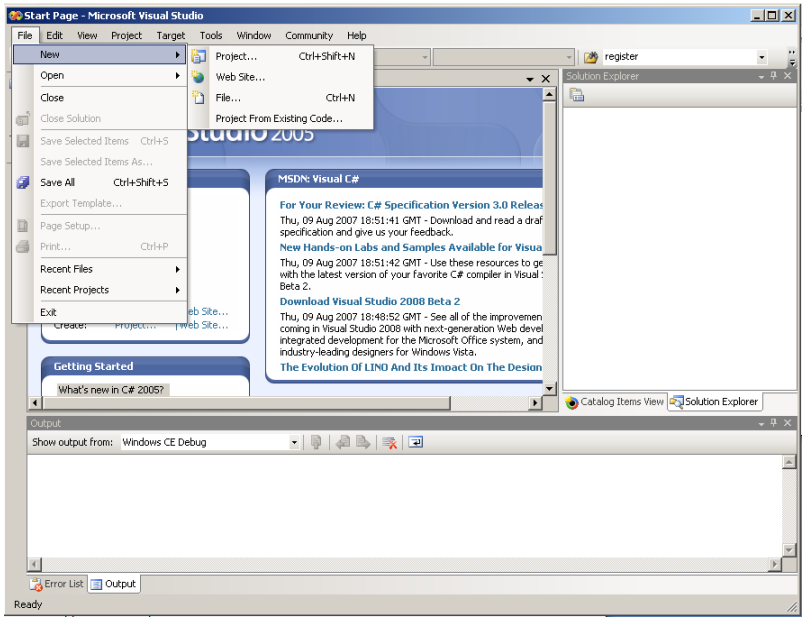
The Windows CE 6.0 SDK for RE2 includes six sample applications that demonstrate the use of RE2 specific API's. The sample applications are detailed below.

Application	Description
BrightnessController	This sample can be used for evaluating the brightness control capability of the RE2 platform. A binary of this sample is included in the Windows CE 6 image and can be accessed from the control panel.
GPIOSample	This sample can be used for evaluating the general purpose input/outputs of the RE2 platform. This application makes use of the GPIOAPI.dll API library.
WatchdogSample	This sample demonstrates how to operate the RE2 watchdog using the watchdog API.
Re2audiomuxsampleapp	This sample demonstrates how to adjust volume levels and change audio routes on the RE2 platform.
Re2apisampleapp	This sample application demonstrates how reset the RE2, how to enable the WiFi and Bluetooth modules, and how to update the system firmware from with the Windows CE operating system.
Cm1sampleapplication	This sample application demonstrates how to operate the optional CM1 expansion board. The CM1 features a GSM/GPRS module, as well as an accelerometer. To use this sample application ensure that CM1 module in enabled in the boot loader. Please see the document, "RE2 Single Board Computer User Guide" for details.
VIP1SampleApplication	This sample application demonstrates how to preview and capture live video using the BCT VIP module. At the command prompt on the target device type "VIP1.exe /?" for details on using the sample application.

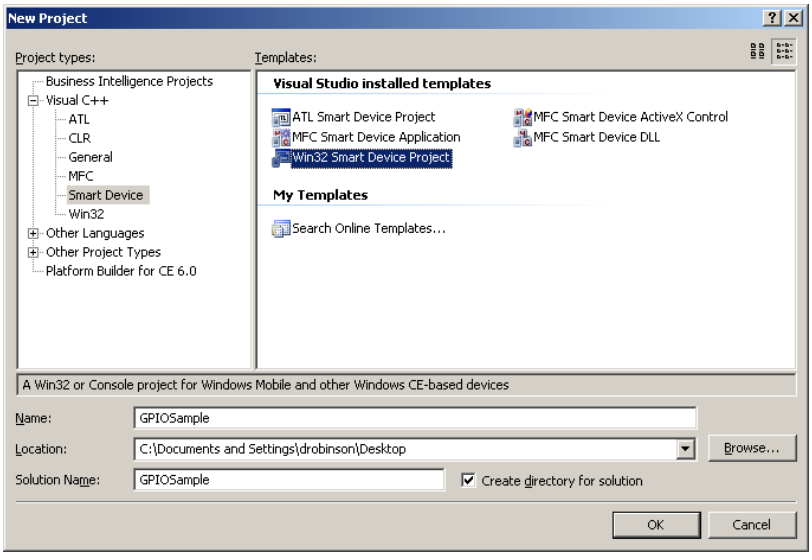
Software development

This section describes how to create a native RE2 Windows CE 6.0 application using the SDK and deploy the application to the RE2 device using Microsoft ActiveSync over USB. The sample application created will demonstrate how to use the RE2 GPIOapi to manipulate the GPIO bits.

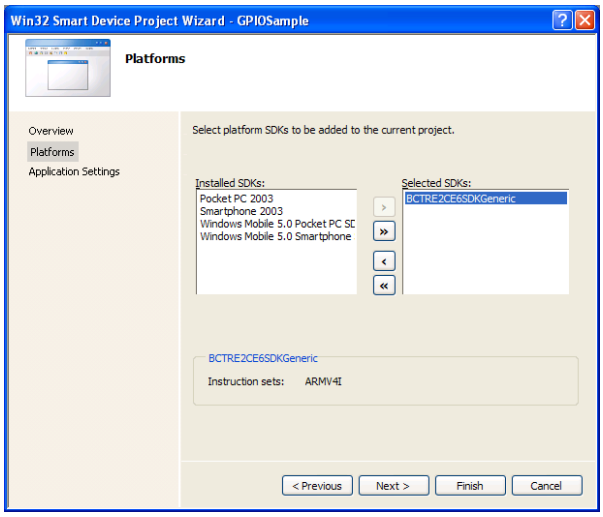
1. Open Visual Studio 2005 or 2008. Click on File ->New ->Project to begin a new project.



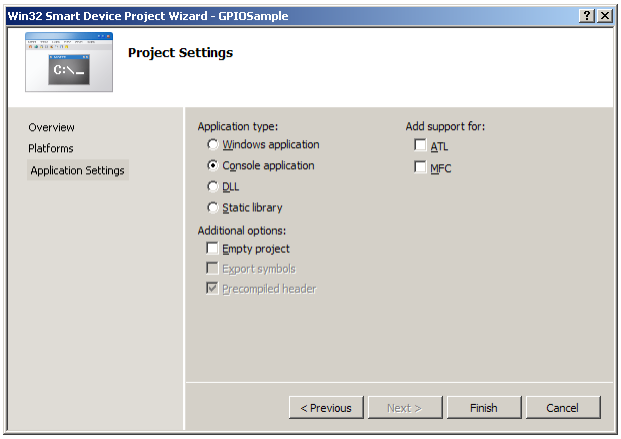
2. Under the “Visual C++” language click “smart device”. Select “Win32 Smart Device Project” and give the project the name “GPIOSample”. Click OK



3. The smart device project wizard should now start. Click next to begin. The RE2 SDK installed in the previous section should now be populated in the “Installed SDKs” list. Arrange the list boxes so that “BCTRE2CE6SDKGeneric” is the only SDK in the “Selected SDK’s” list. Click Next.



Select console application from the “Application type” selection box and click finish.

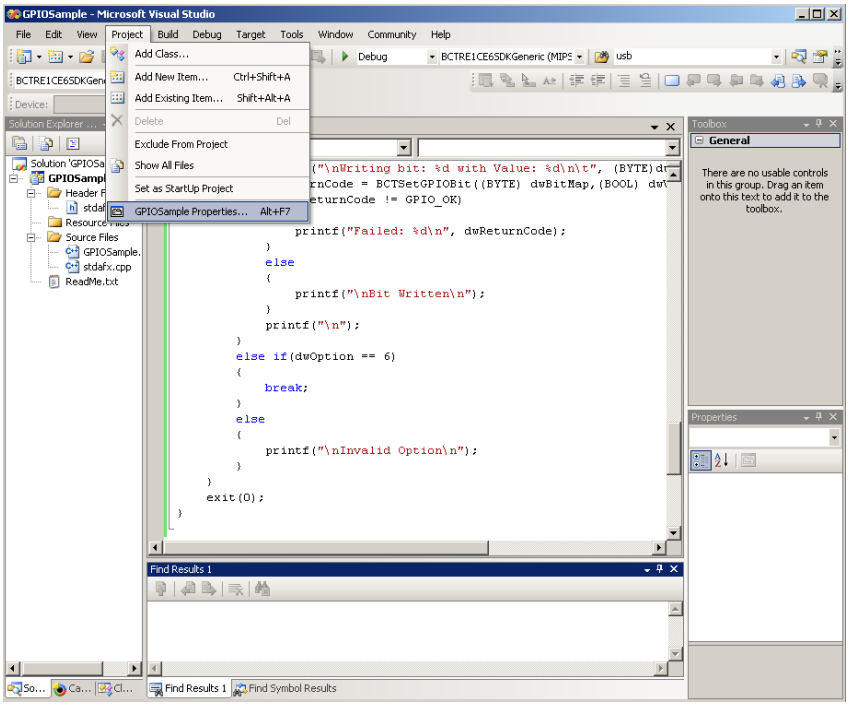


4. Modify the GPIOSample.cpp file to include the following code:

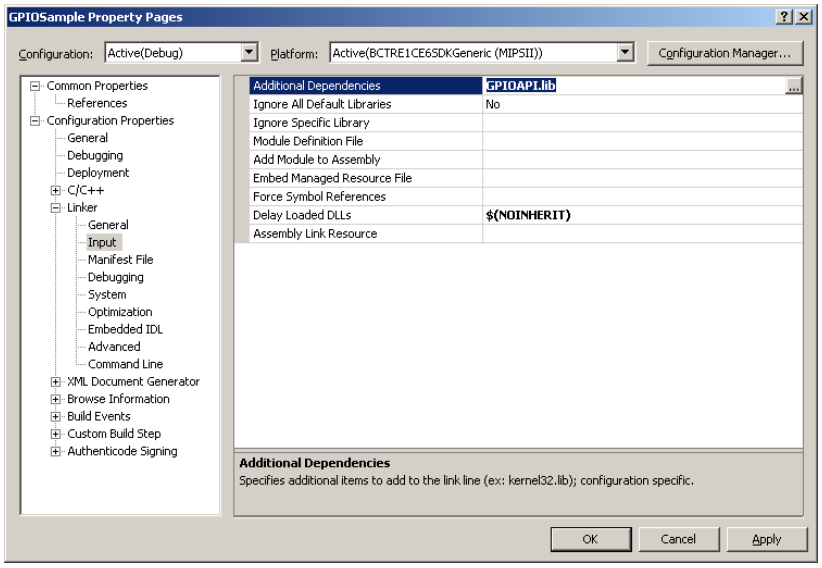
```
#include "stdafx.h"
#include <windows.h>
#include <commctrl.h>

#include <RE2GPIO.h>

int _tmain(int argc, TCHAR *argv[], TCHAR *envp[])
{
        DWORD dwReturnCode;
        DWORD dwOption = 0;
        DWORD dwValue;
        DWORD dwBitMap;
        WORD  wValue;
        BOOL  iValue;

        printf("BCT RE2 GPIO sample application V1.00\n");

        while(1)
        {
                fflush(stdin);
                printf("\n\t1) Read PORT\n");
                printf("\t2) Write a WORD to PORT\n");
                printf("\t3) Set Pin directions\n");
                printf("\t4) Get bit\n");
                printf("\t5) Set bit\n");
                printf("\t6) Exit\n");
                printf("\t\tPlease enter an Option(1-6)");
                scanf_s("%d", &dwOption);
                fflush(stdin);
                if(dwOption == 1)
                {
                        printf("\n\nReading Port....\n\t");

                        dwReturnCode = BCTReadGPIOPort(&wValue);
                        if(dwReturnCode != GPIO_OK)
                        {
                                printf("Failed to read byte with error code: %d\n",
dwReturnCode);
                        }
                        else
                        {
                                printf("Read value %.4xh\n", wValue);
                        }
                        printf("\n");
                }
                else if(dwOption == 2)
                {
                        printf("\n\nPlease enter the byte to write (HEX): ");
                        scanf_s("%x", &dwValue);
                        printf("\n\nWriting Port: %.4xh\n\t", (WORD) dwValue);
                        dwReturnCode = BCTWriteGPIOPort((WORD) dwValue);
                        if(dwReturnCode != GPIO_OK)
                        {
                                printf("Failed to write byte with error code: %d\n",
dwReturnCode);
                        }
                        else
                        {
                                printf("Byte written\n");
                        }
                        printf("\n");
                }
                else if(dwOption == 3)
                {
                        printf("\n\nPlease enter a bitmap for pin Directions (HEX): ");
                        scanf_s("%x", &dwBitMap);
                        printf("\n\nWriting Port directions: %.2xh\n\t", (WORD) dwBitMap);
                        dwReturnCode = BCTSetGPIOPinDirection((WORD) dwBitMap);
                        if(dwReturnCode != GPIO_OK)
                        {
                                printf("Failed to set pin directions with error code: %d\n",
dwReturnCode);
                        }
                        else
```

```
                        {
                                printf("Bit directions written\n");
                        }
                        printf("\n");
                }
                else if(dwOption == 4)
                {
                        printf("\n\nPlease enter which bit value to read (0-11): ");
                        scanf_s("%d", &dwBitMap);
                        printf("\n\nReading bit: %d\n\t", (BYTE)dwBitMap);
                        dwReturnCode = BCTGetGPIOBit((BYTE) dwBitMap, &iValue);
                        if(dwReturnCode != GPIO_OK)
                        {
                                printf("Failed to read bit with error code: %d\n",
dwReturnCode);
                        }
                        else
                        {
                                printf("Read bit value: %d\n", iValue);
                        }
                        printf("\n");
                }
                else if(dwOption == 5)
                {
                        printf("\n\nPlease enter which bit value to write (0-11): ");
                        scanf_s("%d", &dwBitMap);
                        printf("\n\nPlease enter 1 to set or 0 to clear: ");
                        scanf_s("%d", &dwValue);

                        if(dwValue > 1)
                        {
                                dwValue = 1;
                        }
                        if(dwValue < 0)
                        {
                                dwValue = 0;
                        }

                        printf("\nWriting bit: %d with Value: %d\n\t", (BYTE)dwBitMap,(BOOL)
dwValue);
                        dwReturnCode = BCTSetGPIOBit((BYTE) dwBitMap,(BOOL) dwValue);
                        if(dwReturnCode != GPIO_OK)
                        {
                                printf("Failed: %d\n", dwReturnCode);
                        }
                        else
                        {
                                printf("\nBit Written\n");
                        }
                        printf("\n");
                }
                else if(dwOption == 6)
                {
                        break;
                }
                else
                {
                        printf("\nInvalid Option\n");
                }
        }
        exit(0);
}
```

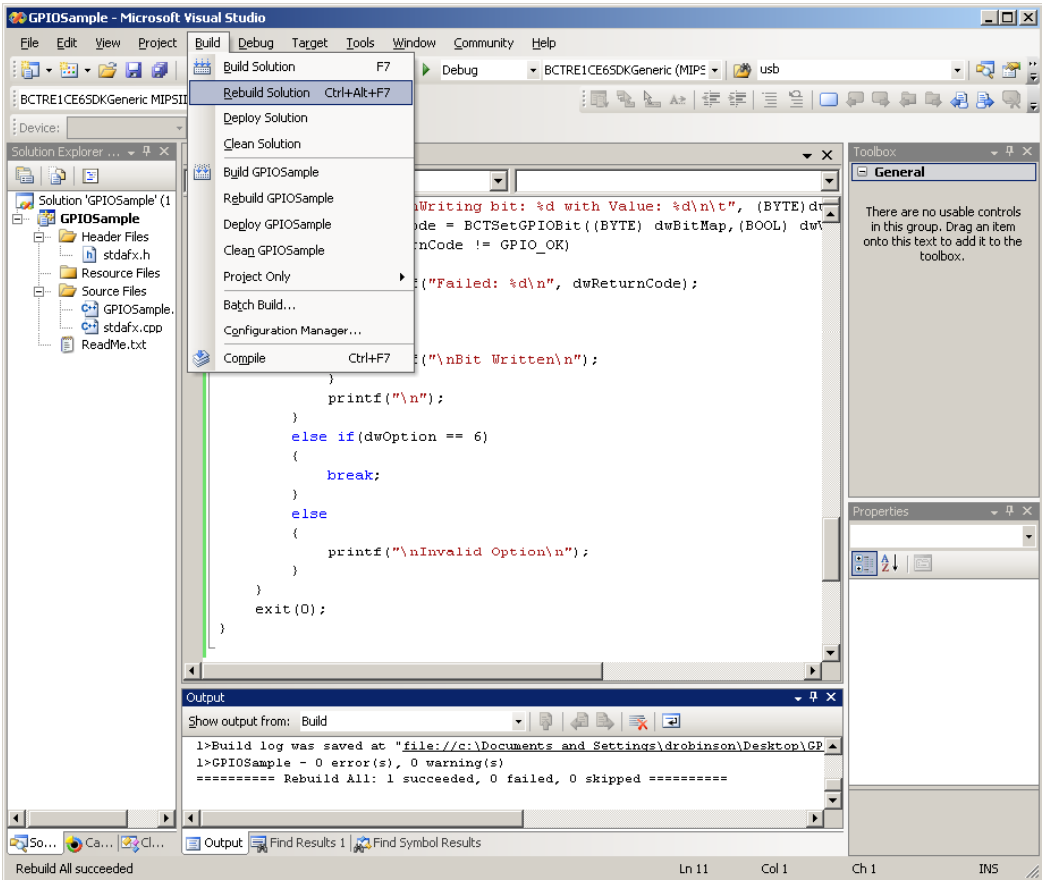
5. As this application is using functions exported by the GPIOAPI library we need to link this project to the file “GPIOAPI.lib”. From the “Project” menu click properties.



6. Under “Configuration Properties -> Linker -> Input”, add “GPIOAPI.lib” to the “Additional dependencies”, and click OK.

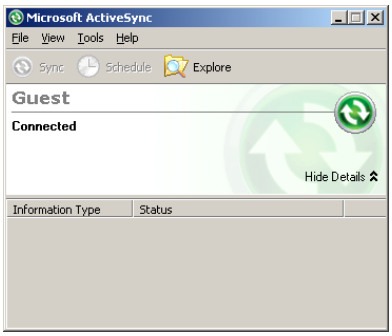


7. We are now ready to compile and build the sample application. From the “Build” menu click on “Rebuild Solution”. If the compile and build was successful the output window should state “1 succeeded, 0 failed”.

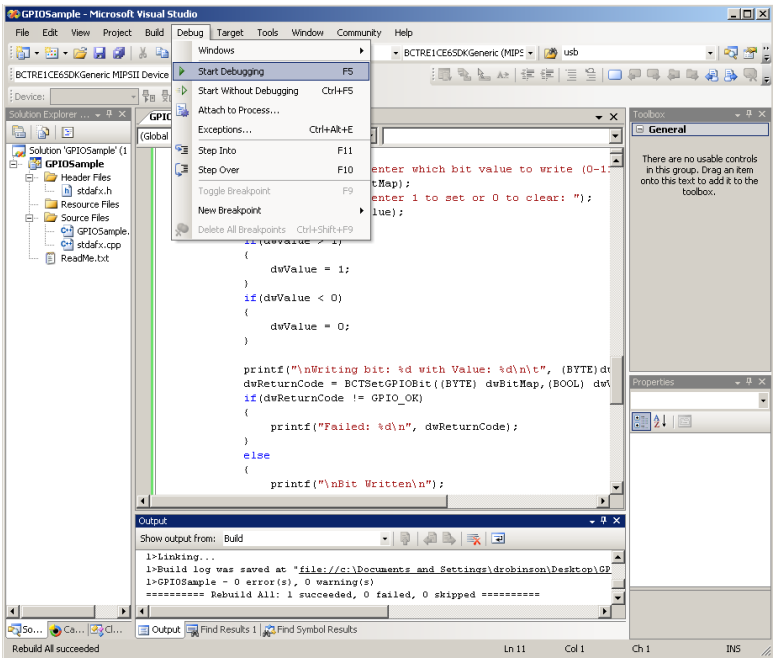


8. Visual Studio 2005 SP1 supports deploying applications automatically to the target device and debugging applications remotely. This requires an ActiveSync connection. Using a USB A/B cable, attach the development machine to the RE2 device port and ensure that the RE2 is turned on.

9. Ensure Microsoft ActiveSync is connected.



10. We can now deploy our application remotely from Visual Studio. From the “Debug” menu click on “Start Debugging”. Visual studio should now download the application to the target and run it.

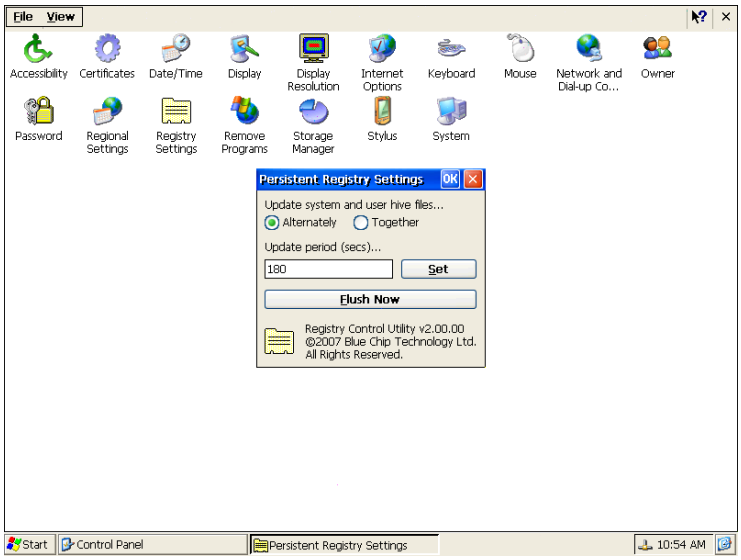


If deployment fails ensure that the USB A/B cable is attached and ActiveSync is connected.

System and Development tools

Registry Settings

Windows CE 6.0 for RE2 comes with hive based registry support. This allows registry settings to be persisted through a cold boot. The Registry Settings utility, accessible from the system control panel can be used to set how often the volatile registry is backed up to solid state media, and also perform manual commits. It is possible for a custom application to manage the persisting the hive registry using the Windows API function "RegFlushKey()". **In the event that a registry change makes the system unusable, a factory reset will force the registry to be restored to its default state on next boot.** Refer to the RE2 user manual for details on how to achive this.

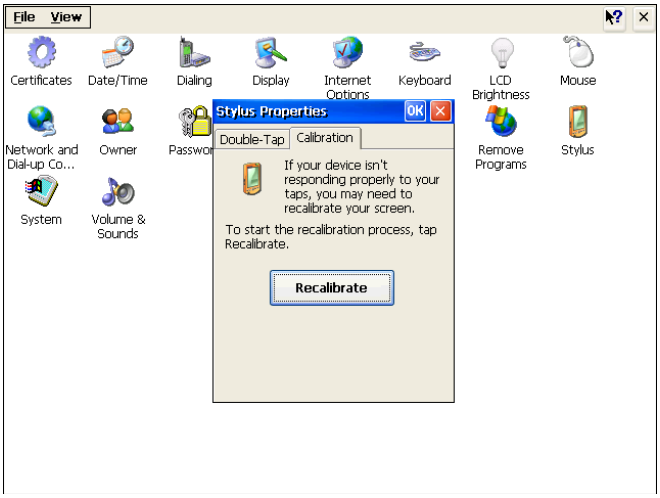


Regedit

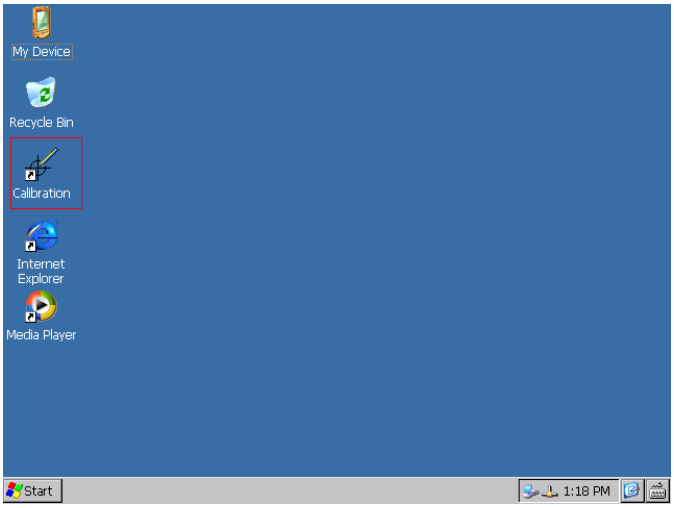
Windows CE 6.0 for RE2 comes with a built in registry editor in the style of the standard Windows registry editor. To access it load "regedit" from either the command prompt or Run menu.

Touch Screen Calibration

Windows CE 6 for RE2 features support for two touch screen controllers. One controller is built onto the RE2 board and can be calibrated using the built in calibration utility. To access it open the “Stylus Properties” window from the system control panel.

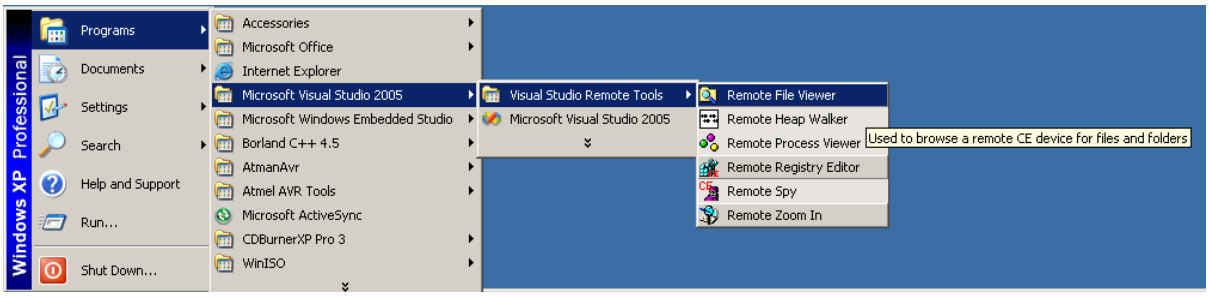


If using the RE2 platform in conjunction with a PM5 personality module a separate calibration utility is provided and can be access from the Windows CE desktop.



Visual Studio 2005 Remote Tools

Visual 2005 includes remote tools that can be used for managing Windows CE images and debugging Windows CE applications. All the remote tools require an ActiveSync connection. The remote tools must be run from the start menu rather than within Visual Studio at location: Start->Programs->Microsoft Visual Studio 2005->Visual Studio Remote Tools.



The table below details the remote tools available and their purpose:

Remote Tool	**Purpose**
Remote File Viewer	Used to browse a remote CE device for files and folders. The same can be achieved using the Explore option in ActiveSync.
Remote Registry Editor	Used to remotely view and edit a Windows CE registry
Remote Heap Walker	Used to remotely view the memory allocation (heap) on a CE device
Remote Spy	Used to remotely view Windows/Messages on a CE device
Remote Process Viewer	Used to remotely view processes running on a Windows CE device
Remote Zoom In	Used to retrieve a current snap shot of a CE device desktop

Automatically starting applications at Windows CE boot

Basic Method

The Windows CE image for RE2 will automatically run any executables that are placed in the "hard disk\startup" or "storage card\startup" folders.

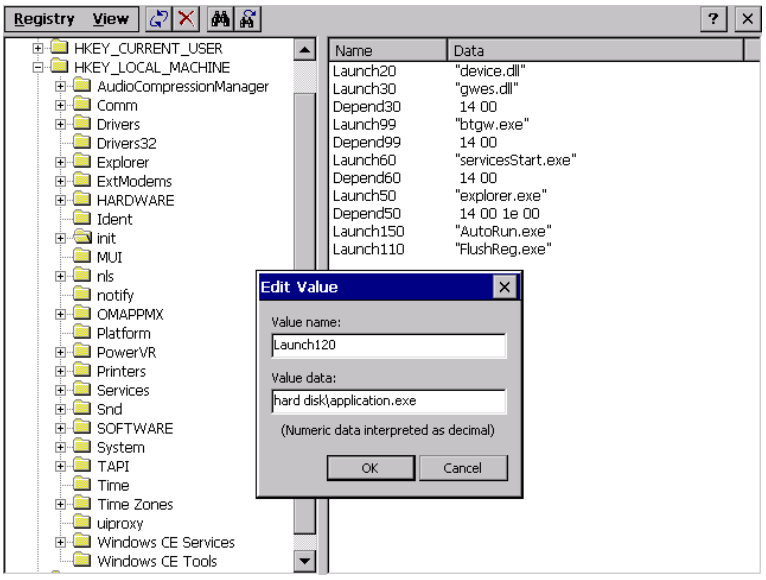
1. Create a folder called "startup" in either the "Hard Disk" or "Storage Card" folder.
2. Copy the exe files that should run at start up into one of the created folders.

Note: Only files with the extension "exe" will be run at start up, and the order that exe files are executed cannot be guaranteed.

Advanced Method

1. Using either the local or remote registry editor navigate to "HKEY_LOCAL_MACHINE\Init" on the device.
2. Add a launch LaunchXX String value with the value data being the path to the executable to load at startup.

E.g. to automatically load "application.exe" at start up from the "Hard Disk" folder the regedit.exe edit value window would look like below.



3. Flush the registry using the Registry Setting control panel applet accessible from the control panel

For further details on the use of LaunchXX value please visit: http://msdn.microsoft.com/en-us/library/ms901773.aspx

Tip: Windows explorer.exe can be prevented from loading, by locating the "explorer.exe" launch value and deleting it. In the screen shot above "Launch50" would be deleted.

RE2 Hardware API Libraries

SMBUS API

The SMBUSAPI is provided to give developers a simple mechanism for accessing devices attached the RE2 SMBUS compatible bus. The four SMBUSAPI functions provided are detailed over the next pages.

BCTSmbusWriteByte

Sends a command, and writes a byte of data to a device on the SMBUS.

```
DWORD WINAPI BCTSmbusWriteByte (BYTE bDeviceAddress, BYTE bCommand, BYTE bData);
```

Parameters

bDeviceAddress
[in] The slave address on the SMBUS to send the command to
bCommand
[in] The SMBUS command identifier
bData
[in] A byte of data to pass in with the command. For commands that do not require any data be passed in, set this value to 0x00

Return Value

If the function succeeds, the return value is `SMBUS_OK`.

If the function fails, the return value is a nonzero error code defined in SMBUS.h.

Remarks

As the SMBUS architecture is a two wire interface it operates on a "first come first served" bases. For this reason the driver also operates in the same way and limits access to its functions to one process at a time. If the SMBUS is accessed while already in use the error code `SMBUS_DRIVER_LOCKED_BY_OTHER_PROCESS` will be returned and is normal. The application should wait for an undefined period before retrying.

Requirements

Header	Declared in SMBUS.h
Library	Use SMBUSAPI.lib.
DLL	Requires SMBUSAPI.dll.

BCTSmbusReadByte

Sends a command, and reads a byte of data from a device on the SMBUS.

```
DWORD WINAPI BCTSmbusReadByte (BYTE bDeviceAddress, BYTE bCommand, PBYTE pbData);
```

Parameters

bDeviceAddress
[in] The slave address on the SMBUS to send the command to
bCommand
[in] The SMBUS command identifier
pbData
[out] A pointer to an 8 bit value to hold the data returned

Return Value

If the function succeeds, the return value is `SMBUS_OK`.

If the function fails, the return value is a nonzero error code defined in SMBUS.h.

Remarks

As the SMBUS architecture is a two wire interface it operates on a "first come first served" bases. For this reason the driver also operates in the same way and limits access to its functions to one process at a time. If the SMBUS is accessed while already in use the error code `SMBUS_DRIVER_LOCKED_BY_OTHER_PROCESS` will be returned and is normal. The application should wait for an undefined period before retrying.

Requirements

Header	Declared in SMBUS.h
Library	Use SMBUSAPI.lib.
DLL	Requires SMBUSAPI.dll.

BCTSmbusBufferedWrite

Sends a command, and writes up to 16 bytes of data.

```
DWORD WINAPI BCTSmbusBufferedWrite(BYTE bDeviceAddress, BYTE bCommand, BYTE
bdata[], BYTE bBytesToWrite);
```

Parameters

bDeviceAddress
[in] The slave address on the SMBUS to send the command to
bCommand
[in] The SMBUS command identifier
bdata
[in] An pointer to an array of bytes to write
bBytesToWrite [in] The number of bytes to write

Return Value

If the function succeeds, the return value is `SMBUS_OK`.

If the function fails, the return value is a nonzero error code defined in SMBUS.h.

Remarks

As the SMBUS architecture is a two wire interface it operates on a "first come first served" bases. For this reason the driver also operates in the same way and limits access to its functions to one process at a time. If the SMBUS is accessed while already in use the error code `SMBUS_DRIVER_LOCKED_BY_OTHER_PROCESS` will be returned and is normal. The application should wait for an undefined period before retrying. This function supports sending a maximum of 16 bytes at a time. This function can also be used for SMBUS quick writes, by setting the bBytesToWrite to 0. This will cause the function to send the command without a data phase.

Requirements

Header	Declared in SMBUS.h
Library	Use SMBUSAPI.lib.
DLL	Requires SMBUSAPI.dll.

BCTSmbusBufferedRead

Sends a command, and reads up to 16 bytes of data.

```
DWORD WINAPI BCTSmbusBufferedRead(BYTE bDeviceAddress, BYTE bCommand, BYTE bdata[],
BYTE bBytesToRead);
```

Parameters

bDeviceAddress
[in] The slave address on the SMBUS to send the command to
bCommand
[in] The SMBUS command identifier
bData
[out] A pointer to an array of bytes to read into
bBytesToRead
[in] The number of bytes to read.

Return Value

If the function succeeds, the return value is `SMBUS_OK`.

If the function fails, the return value is a nonzero error code defined in SMBUS.h.

Remarks

As the SMBUS architecture is a two wire interface it operates on a "first come first served" bases. For this reason the driver also operates in the same way and limits access to its functions to one process at a time. If the SMBUS is accessed while already in use the error code `SMBUS_DRIVER_LOCKED_BY_OTHER_PROCESS` will be returned and is normal. The application should wait for an undefined period before retrying. This function supports reading a maximum of 16 bytes at a time.

Requirements

Header	Declared in SMBUS.h
Library	Use SMBUSAPI.lib.
DLL	Requires SMBUSAPI.dll.

LCD Brightness API

The LCD brightness API library allows the brightness of compatible LCD's to be changed. The library exports two functions which are detailed below.

BCTSetLCDBrightness

Sets the LCD brightness to the value specified

```
DWORD WINAPI BCTSetLCDBrightness(BYTE bBrightness);
```

Parameters

bBrightness
[in] The brightness value to write

Return Value

If the function succeeds, the return value is `BACKLIGHT_OK`.

If the function fails, the return value is a nonzero error code defined in BCTLCDBrightnessAPI.h.

Remarks

When bBrightness is set to 0 the LCD will be at its dimmest.

Requirements

Header	Declared in BCTLCDBrightnessAPI.h
Library	BCTLCDBrightnessAPI.lib
DLL	BCTLCDBrightnessAPI.dll

BCTGetLCDBrightness

Retrieves the current LCD brightness.

```
DWORD WINAPI BCTGetLCDBrightness(PBYTE bBrightness);
```

Parameters

bBrightness
[out] A pointer to a byte that will hold the current LCD brightness

Return Value

If the function succeeds, the return value is `BACKLIGHT_OK`.

If the function fails, the return value is a nonzero error code defined in BCTLCDBrightnessAPI.h.

Remarks

When bBrightness is set to 0 the LCD will be at its dimmest.

Requirements

Header	Declared in BCTLCDBrightnessAPI.h
Library	BCTLCDBrightnessAPI.lib
DLL	BCTLCDBrightnessAPI.dll

GPIO API

The GPIO API library provides access to the 12 available GPIO pins on the RE2 platform. The library exports five functions which are detailed below.

BCTSetGPIOPinDirection

Sets the directions of GPIO bits to either input or output.

```
DWORD WINAPI BCTSetGPIOPinDirection(WORD wVal);
```

Parameters

wVal
[in] A bitmap of the required pin directions. Bit set = input. Bit cleared = output.

E.g. Passing a value of 0x05 into the function would set bits 0 and 2 to inputs and other bits to outputs

Return Value

If the function succeeds, the return value is `GPIO_OK`.

If the function fails, the return value is a nonzero error code defined in re2gpio.h.

Remarks

Bits 12 – 15 of wVal are ignored.

Requirements

Header	Declared in RE2GPIO.h
Library	Use GPIOapi.lib
DLL	Requires GPIOapi.dll

BCTReadGPIOPort

Reads the current state of the GPIO port

```
DWORD WINAPI BCTReadGPIOPort (PWORD pwVal);
```

Parameters

pwVal
[out] A pointer to an 16 bit value that will hold the value of the GPIO port.

Return Value

If the function succeeds, the return value is `GPIO_OK`.

If the function fails, the return value is a nonzero error code defined in re2gpio.h.

Remarks

Bits 12 – 15 of pwVal should be ignored.

Requirements

Header	Declared in RE2GPIO.h
Library	Use GPIOapi.lib.
DLL	Requires GPIOapi.dll.

BCTWriteGPIOPort

Writes to the GPIO port

```
DWORD WINAPI BCTWriteGPIOPort (WORD wVal);
```

Parameters

wVal
[in] The word that gets written to the GPIO port.

Return Value

If the function succeeds, the return value is `GPIO_OK`.

If the function fails, the return value is a nonzero error code defined in re2gpio.h.

Remarks

Bits 12 – 15 of wVal are ignored.

Requirements

Header	Declared in RE2GPIO.h
Library	Use GPIOapi.lib.
DLL	Requires GPIOapi.dll.

BCTSetGPIOBit

Sets an individual bit to a value specified

```
DWORD WINAPI BCTSetGPIOBit (WORD wBitNumber, BOOL iVal);
```

Parameters

wBitNumber
[in] The bit that should be written. Acceptable values 0-11
iVal
[in] The value to be written to the bit. TRUE = Set, FALSE = Clear

Return Value

If the function succeeds, the return value is `GPIO_OK`.

If the function fails, the return value is a nonzero error code defined in re2gpio.h.

Remarks

Requirements

Header	Declared in RE2GPIO.h
Library	Use GPIOapi.lib.
DLL	Requires GPIOapi.dll.

BCTGetGPIOBit

Gets the value of an individual bit

```
DWORD WINAPI BCTGetGPIOBit (WORD wBitNumber, PBOOL piVal);
```

Parameters

wBitNumber
[in] The bit that should be read. Acceptable values 0-11
iVal
[in] A pointer to a BOOL that will hold the state of the pin. TRUE = Set, FALSE = Clear

Return Value

If the function succeeds, the return value is `GPIO_OK`.

If the function fails, the return value is a nonzero error code defined in re2gpio.h.

Remarks

Requirements

Header	Declared in RE2GPIO.h
Library	Use GPIOapi.lib.
DLL	Requires GPIOapi.dll.

Watchdog API

The Watchdog API allows the system watchdog to be used to cause a system reset in the event of an unresponsive application. The library exports five functions which are detailed below.

BCTWatchdogStatus

Retrieves the current status of the RE2 watchdog.

```
DWORD WINAPI BCTWatchDogStatus (BOOL * bEnabled);
```

Parameters

bEnabled
[out] Boolean value indicating if the watchdog is currently enabled.

Return Value

If the function succeeds, the return value is `WATCHDOG_OK`.

If the function fails, the return value is a nonzero error code defined in watchdog.h.

Remarks

This function can be used to detect if a watchdog protected boot was enabled, and thus determine if the watchdog should be disabled for refreshed after boot.

Requirements

Header	Declared in Watchdog.h
Library	Use watchdog.lib
DLL	Requires watchdog.dll

BCTEnableWatchdog

Enables the RE2 watchdog to timeout in the time specified

```
DWORD WINAPI BCTEnableWatchDog (BYTE bTickInterval, BYTE bTimeout);
```

Parameters

bTickInterval [in] The duration between watchdog ticks. 0x01 = 4ms, 0x02 = 1sec, 0x03 = 1min.

bTimeout [in] The amount of bTickInterval before a timeout is triggered. Must be greater than 0.

Return Value

If the function succeeds, the return value is `WATCHDOG_OK`.

If the function fails, the return value is a nonzero error code defined in watchdog.h.

Remarks

Requirements

Header	Declared in Watchdog.h
Library	Use watchdog.lib
DLL	Requires watchdog.dll

BCTDisableWatchdog

Disables the RE2 watchdog.

```
DWORD WINAPI BCTDisableWatchDog (VOID);
```

Parameters

Return Value

If the function succeeds, the return value is `WATCHDOG_OK`.

If the function fails, the return value is a nonzero error code defined in watchdog.h.

Remarks

Requirements

Header	Declared in Watchdog.h
Library	Use watchdog.lib.
DLL	Requires watchdog.dll.

BCTRefreshWatchdog

Resets the watchdog counter to the timeout value.

```
DWORD WINAPI BCTRefreshWatchDog (VOID);
```

Parameters

Return Value

If the function succeeds, the return value is `WATCHDOG_OK`.

If the function fails, the return value is a nonzero error code defined in watchdog.h.

Remarks

Requirements

Header	Declared in Watchdog.h & azfavr.h
Library	Use watchdog.lib.
DLL	Requires watchdog.dll.

BCTSignalSuccessfulBoot

Signals to the firmware that the RE2 has booted successfully.

```
DWORD WINAPI BCTSignalSuccessfulBoot(VOID);
```

Parameters

Return Value

If the function succeeds, the return value is `WATCHDOG_OK`.

If the function fails, the return value is a nonzero error code defined in watchdog.h.

Remarks

This function is only useful if watchdog protected boot is enabled.

Requirements

Header	Declared in Watchdog.h
Library	Use watchdog.lib.
DLL	Requires watchdog.dll.

RE2 API

The RE2 API provides five functions as detailed below

BCTEnableWIFIBT

Allows the WiFi and Bluetooth modules to be turned on and off independently of each other.

```
DWORD WINAPI BCTEnableWIFIBT(BOOL bEnableWIFI, BOOL bEnableBT);
```

Parameters

bEnableWIFI [in] TRUE = WiFi enabled

bEnableBT [in] TRUE = BT enabled

Return Value

If the function succeeds, the return value is `RE2API_OK`.

If the function fails, the return value is a nonzero error code defined in re2api.h.

Remarks

Requirements

Header	Declared in re2api .h
Library	Use RE2API.lib.
DLL	Requires RE2API .dll

BCTWIFIBTStatus

Retrieves the current status of the WiFi and Bluetooth modules

```
DWORD WINAPI BCTWIFIBTStatus(BOOL * WifiStatus, BOOL * BTStatus);
```

Parameters

WifiStatus [out] TRUE = WiFi enabled

BTStatus [out] TRUE = BT enabled

Return Value

If the function succeeds, the return value is `RE2API_OK`.

If the function fails, the return value is a nonzero error code defined in re2api.h.

Remarks

Requirements

Header	Declared in re2api .h
Library	Use RE2API.lib.
DLL	Requires RE2API .dll

BCTResetSystem

Causes RE2 to reset

```
DWORD WINAPI BCTResetSystem();
```

Parameters

Return Value

If the function succeeds, the return value is `RE2API_OK`.

If the function fails, the return value is a nonzero error code defined in re2api.h.

Remarks

Requirements

Header	Declared in re2api .h
Library	Use RE2API.lib.
DLL	Requires RE2API .dll

BCTEraseUserOSPartition

Erases the Windows CE updatable firmware partition

```
DWORD WINAPI BCTEraseUserOSPartition();
```

Parameters

Return Value

If the function succeeds, the return value is `RE2API_OK`.

If the function fails, the return value is a nonzero error code defined in re2api.h.

Remarks

See the RE2APISampleApp in the Windows CE SDK for an example of how to use this function.

Requirements

Header	Declared in re2api .h
Library	Use RE2API.lib.
DLL	Requires RE2API .dll

BCTProgramUserOSImage

Programs the Windows CE updatable firmware partition with the image specified

```
DWORD WINAPI BCTProgramUserOSPartition(LPWSTR ImageFile);
```

Parameters

ImageFile [in] path to the operating system image.

Return Value

If the function succeeds, the return value is `RE2API_OK`.

If the function fails, the return value is a nonzero error code defined in re2api.h.

Remarks

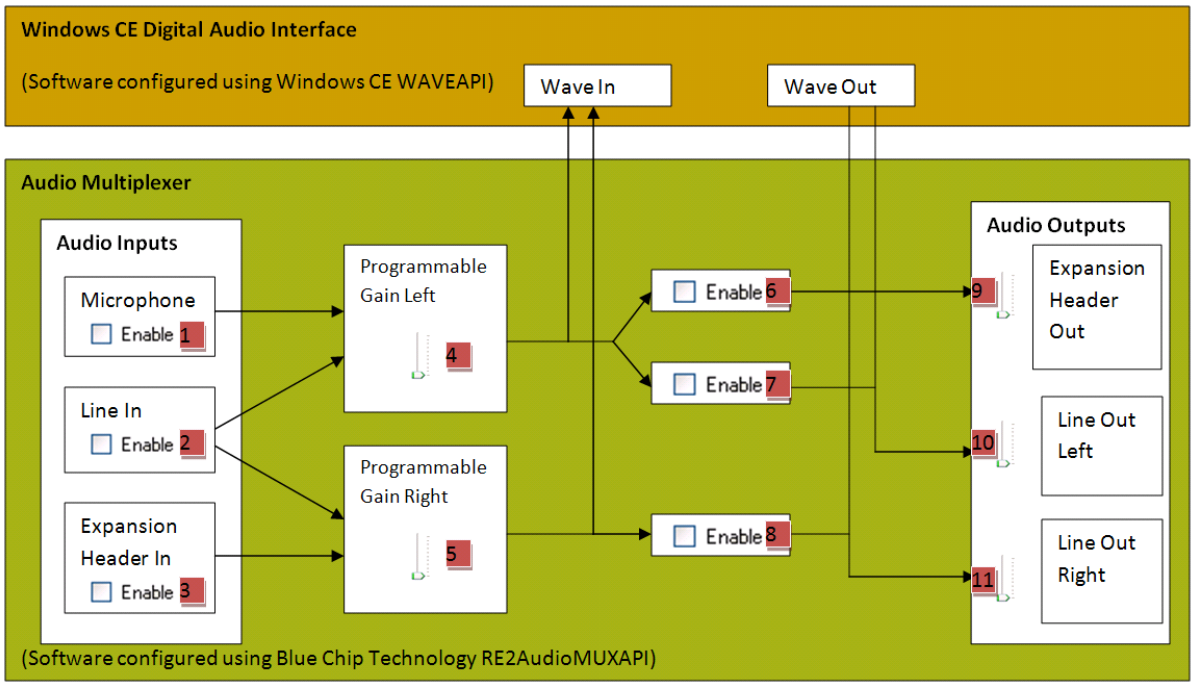
See the RE2APISampleApp in the Windows CE SDK for an example of how to use this function. When programming Windows CE firmware use the BCTSignalSuccessfulBoot function to ensure that the firmware tries to boot the new firmware on the next boot.

Requirements

Header	Declared in re2api .h
Library	Use RE2API.lib.
DLL	Requires RE2API .dll

Audio Multiplexer API

RE2 features multiple audio inputs and outputs with different routing options. The following functions define how each aspect of the audio multiplexer can be operated. Each function corresponds to a numbered aspect of the below diagram.



See Table – Audio Multiplexer API Functions below for individual function information

Parameters

See Table - Audio Multiplexer API functions

Return Value

If the function succeeds, the return value is `AUDIO_MUX_OK`.

If the function fails, the return value is a nonzero error code defined in RE2AudioMUXAPI.h

Remarks

See the Re2AudioMuxSampleApp in the Windows CE SDK for an example of how to use this API.

Requirements

Header	Declared in RE2AudioMUXAPI.h
Library	Use RE2AudioMUXAPI .lib.
DLL	Requires RE2AudioMUXAPI.dll

Mux ID	Function prototype	Function Description	Parameter Information
1	`DWORD WINAPI MicInEnable(BOOL bEnable);`	Enables / Disables Microphone input	TRUE = MIC Enabled
1	`DWORD WINAPI MicInStatus(BOOL * bEnable);`	Gets the enable status of the Microphone input	TRUE = MIC Enabled
2	`DWORD WINAPI LineInEnable(BOOL bEnable);`	Enables / Disables Line In input	TRUE = Line In Enabled
2	`DWORD WINAPI LineInStatus(BOOL * bEnable);`	Gets the enable status of the Line In input	TRUE = Line In Enabled
3	`DWORD WINAPI ExpansionInEnable(BOOL bEnable);`	Enables / Disables Expansion In input	TRUE = Expansion Enabled
3	`DWORD WINAPI ExpansionInStatus(BOOL * bEnable);`	Gets the enable status of the Expansion input	TRUE = Expansion Enabled
4	`DWORD WINAPI SetLeftPGAVolume(BYTE volume);`	Sets the gain of the left PGA	Gain between 0-127. 0 = Muted
4	`DWORD WINAPI GetLeftPGAVolume(BYTE * volume);`	Gets the current gain of the left PGA	Gain between 0-127. 0 = Muted
5	`DWORD WINAPI SetRightPGAVolume(BYTE volume);`	Sets the gain of the right PGA	Gain between 0-127. 0 = Muted
5	`DWORD WINAPI GetRightPGAVolume(BYTE * volume);`	Gets the current gain of the right PGA	Gain between 0-127. 0 = Muted
6	`DWORD WINAPI LeftPGAToExpansionOut(BOOL bEnable);`	Enables / Disables the audio route from the left PGA to the Expansion output	TRUE = Route Enabled
6	`DWORD WINAPI LeftPGAToExpansionOutStatus(BOOL * bEnable);`	Gets the left PGA to the Expansion output route status	TRUE = Route Enabled
7	`DWORD WINAPI LeftPGAToLineOutLeft(BOOL bEnable);`	Enables / Disables the audio route from the left PGA to the left Line Out	TRUE = Route Enabled
7	`DWORD WINAPI LeftPGAToLineOutLeftStatus(BOOL * bEnable);`	Gets the left PGA to the left Line Out route status	TRUE = Route Enabled
8	`DWORD WINAPI RightPGAToLineOutRight(BOOL bEnable);`	Enables / Disables the audio route from the right PGA to the right Line Out	TRUE = Route Enabled
8	`DWORD WINAPI RightPGAToLineOutRightStatus(BOOL * bEnable);`	Gets the right PGA to the right Line Out route status	TRUE = Route Enabled
9	`DWORD WINAPI SetExpansionOutVolume(BYTE bVolume);`	Sets the Expansion audio output amplification	Volume between 0-10. 0 = Muted
9	`DWORD WINAPI GetExpansionOutVolume(BYTE * bVolume);`	Gets the current Expansion audio output amplification	Volume between 0-10. 0 = Muted
10	`DWORD WINAPI SetLineOutVolumeLeft(BYTE bVolume);`	Sets the left Line Out audio output amplification	Volume between 0-10. 0 = Muted

10	`DWORD WINAPI GetLineOutVolumeLeft(BYTE * bVolume);`	Gets the current Left Line Out audio output amplification	Volume between 0-10. 0 = Muted
11	`DWORD WINAPI SetLineOutVolumeRight(BYTE bVolume);`	Sets the right Line Out audio output amplification	Volume between 0-10. 0 = Muted
11	`DWORD WINAPI GetLineOutVolumeRight(BYTE * bVolume);`	Gets the current right Line Out audio output amplification	Volume between 0-10. 0 = Muted

Table - Audio Multiplexer API functions

Appendix A – Windows CE components included in the generic Windows CE 6 image for RE2

SYSGEN_ASYNCMAC, SYSGEN_AS_BASE, SYSGEN_ATL, SYSGEN_AUDIO, SYSGEN_AUDIO_ACM, SYSGEN_AUDIO_STDWAVEFILES, SYSGEN_AUTH, SYSGEN_AUTH_KERBEROS, SYSGEN_AUTH_NTLM, SYSGEN_AUTH_SCHANNEL, SYSGEN_AUTORAS, SYSGEN_AYGSHELL, SYSGEN_BATTERY, SYSGEN_BTH_A2DP, SYSGEN_BTH_AG, SYSGEN_BTH_AUDIO, SYSGEN_BTH_AVDTP, SYSGEN_BTH_GATEWAY, SYSGEN_BTH_HID_KEYBOARD, SYSGEN_BTH_HID_MOUSE, SYSGEN_BTH_PAN, SYSGEN_BTH_USB_ONLY, SYSGEN_CACHEFILT, SYSGEN_CEDDK, SYSGEN_CEPLAYER, SYSGEN_CERTS, SYSGEN_CMD, SYSGEN_COMMCTRL, SYSGEN_COMMDLG, SYSGEN_CONNMC, SYSGEN_CONSOLE, SYSGEN_CORELOC, SYSGEN_CORESTRA, SYSGEN_CPP_EH_AND_RTTI, SYSGEN_CREDMAN, SYSGEN_CRYPTO, SYSGEN_CTLPNL, SYSGEN_CURSOR, SYSGEN_DCOM, SYSGEN_DCOM_STG, SYSGEN_DDRAW, SYSGEN_DEVICE, SYSGEN_DEVLOAD, SYSGEN_DHCPSRV, SYSGEN_DISPLAY, SYSGEN_DNSAPI, SYSGEN_DOTNETV2_SUPPORT, SYSGEN_DOTNETV35, SYSGEN_DOTNETV35_SUPPORT, SYSGEN_DSHOW, SYSGEN_DSHOW_ACMWRAP, SYSGEN_DSHOW_AVI, SYSGEN_DSHOW_CAPTURE, SYSGEN_DSHOW_DISPLAY, SYSGEN_DSHOW_DMO, SYSGEN_DSHOW_ERRORS, SYSGEN_DSHOW_GENERICSTREAMER, SYSGEN_DSHOW_ICM, SYSGEN_DSHOW_IMAADPCM, SYSGEN_DSHOW_MP3, SYSGEN_DSHOW_MPEGA, SYSGEN_DSHOW_MPEGSPLITTER, SYSGEN_DSHOW_MPEGV, SYSGEN_DSHOW_MSADPCM, SYSGEN_DSHOW_MSRLE, SYSGEN_DSHOW_VIDREND, SYSGEN_DSHOW_WAV, SYSGEN_DSHOW_WAVEOUT, SYSGEN_DSHOW_WMA, SYSGEN_DSHOW_WMA_VOICE, SYSGEN_DSHOW_WMP, SYSGEN_DSHOW_WMT, SYSGEN_DSHOW_WMT_ASXV1, SYSGEN_DSHOW_WMT_ASXV2, SYSGEN_DSHOW_WMT_ASXV3, SYSGEN_DSHOW_WMT_HTTP, SYSGEN_DSHOW_WMT_LOCAL, SYSGEN_DSHOW_WMT_MMS, SYSGEN_DSHOW_WMT_MULTI, SYSGEN_DSHOW_WMT_NSC, SYSGEN_DSHOW_WMV, SYSGEN_EAP, SYSGEN_ETHERNET, SYSGEN_ETH_80211, SYSGEN_EXFAT, SYSGEN_FATFS, SYSGEN_FIBER, SYSGEN_FMTMSG, SYSGEN_FONTS_COUR_1_30, SYSGEN_FONTS_TAHOMA_1_07, SYSGEN_FONTS_WINGDING, SYSGEN_FSDBASE, SYSGEN_FSPASSWORD, SYSGEN_FSRAMROM, SYSGEN_FSREGHIVE, SYSGEN_FSREPLBIT, SYSGEN_FULL_CRT, SYSGEN_GDI_ALPHABLEND, SYSGEN_GESTUREANIMATION, SYSGEN_GRADFILL, SYSGEN_IABASE, SYSGEN_IE, SYSGEN_IESAMPLE, SYSGEN_IESAMPLE_EXR, SYSGEN_IE_FLASHLITE, SYSGEN_IE_JSCRIPT, SYSGEN_IE_TILEENGINE, SYSGEN_IE_VBSCRIPT, SYSGEN_IMAGING, SYSGEN_IMAGING_BMP_DECODE, SYSGEN_IMAGING_BMP_ENCODE, SYSGEN_IMAGING_GIF_DECODE, SYSGEN_IMAGING_GIF_ENCODE, SYSGEN_IMAGING_ICO_DECODE, SYSGEN_IMAGING_JPG_DECODE, SYSGEN_IMAGING_JPG_ENCODE, SYSGEN_IMAGING_PNG_DECODE, SYSGEN_IMAGING_PNG_ENCODE, SYSGEN_IMAGING_TIFF_DECODE, SYSGEN_IMAGING_TIFF_ENCODE, SYSGEN_IMM, SYSGEN_INETCPL, SYSGEN_IPHLPAPI, SYSGEN_LOCALAUDIO, SYSGEN_MEDIAAPPS_WMPOCX, SYSGEN_MINGDI, SYSGEN_MINGWES, SYSGEN_MININPUT, SYSGEN_MINWMGR, SYSGEN_MLANG, SYSGEN_MODEM, SYSGEN_MSGQUEUE, SYSGEN_MSHTML, SYSGEN_MSIM, SYSGEN_MSPART, SYSGEN_MSXML_DOM, SYSGEN_MSXML_XQL, SYSGEN_NDIS, SYSGEN_NDISUIO, SYSGEN_NETAPI32, SYSGEN_NETUTILS, SYSGEN_NKCOMPR, SYSGEN_NKMAPFILE, SYSGEN_NLED, SYSGEN_NOTIFY, SYSGEN_OBEX_CLIENT, SYSGEN_OBEX_FILEBROWSER, SYSGEN_OBEX_SERVER, SYSGEN_OLE, SYSGEN_OLE_GUIDS, SYSGEN_OLE_STG, SYSGEN_PHYSICSENGINE, SYSGEN_PM, SYSGEN_POWERVR, SYSGEN_PPP, SYSGEN_PPP_SERVER, SYSGEN_PRINTING, SYSGEN_PVR_OPENGLES1, SYSGEN_PVR_OPENGLES2, SYSGEN_PVR_OPENVG, SYSGEN_RDP, SYSGEN_REDIR, SYSGEN_RELFSD, SYSGEN_SDBUS, SYSGEN_SD_MEMORY, SYSGEN_SERDEV, SYSGEN_SERVICES, SYSGEN_SHDOCVW, SYSGEN_SHELL, SYSGEN_SMARTCARD, SYSGEN_SOAPTK_CLIENT, SYSGEN_SOFTKB, SYSGEN_SQLCOMPACT, SYSGEN_SQLCOMPACT_MP, SYSGEN_SQLCOMPACT_SQLMP, SYSGEN_STANDARDSHELL, SYSGEN_STDIO, SYSGEN_STDIOA, SYSGEN_STOREMGR, SYSGEN_STOREMGR_CPL, SYSGEN_STREAMAUDIO, SYSGEN_STREAMAV, SYSGEN_STRSAFE, SYSGEN_TAPI, SYSGEN_TCPIP, SYSGEN_TFAT, SYSGEN_TIMESVC_DST, SYSGEN_TOOLHELP, SYSGEN_TOUCH, SYSGEN_TOUCHGESTURE, SYSGEN_UIPROXY, SYSGEN_UNIMODEM, SYSGEN_URLMON, SYSGEN_USB, SYSGEN_USBFN, SYSGEN_USBFN_ETHERNET, SYSGEN_USBFN_SERIAL, SYSGEN_USB_HID, SYSGEN_USB_HID_CLIENTS, SYSGEN_USB_HID_KEYBOARD, SYSGEN_USB_HID_MOUSE, SYSGEN_USB_STORAGE, SYSGEN_VEM, SYSGEN_WCELOAD, SYSGEN_WCETK, SYSGEN_WININET, SYSGEN_WINSOCK, SYSGEN_XAML_RUNTIME

Appendix B – Windows CE components included in the generic Windows CE 7 image for RE2

SYSGEN_3GPP_DEMUX=1, SYSGEN_AAC_DECODE_FILTER=1, SYSGEN_AAC_PARSER=1,
SYSGEN_ACM_GSM610=1, SYSGEN_ASYNCMAC=1, SYSGEN_AS_BASE=1, SYSGEN_ATL=1,
SYSGEN_AUDIO=1, SYSGEN_AUDIO_ACM=1, SYSGEN_AUDIO_STDWAVEFILES=1, SYSGEN_AUTH=1,
SYSGEN_AUTH_KERBEROS=1, SYSGEN_AUTH_NTLM=1, SYSGEN_AUTH_SCHANNEL=1,
SYSGEN_AUTORAS=1, SYSGEN_AYGSHELL=1, SYSGEN_BATTERY=1, SYSGEN_BTH=1, Sysgen_bth_a2dp=1,
SYSGEN_BTH_AG=1, SYSGEN_BTH_AUDIO=1, SYSGEN_BTH_HID_KEYBOARD=1,
SYSGEN_BTH_HID_MOUSE=1, SYSGEN_BTH_PAN=1, SYSGEN_CACHEFILT=1, SYSGEN_CCPSELECT=1,
SYSGEN_CEDDK=1, SYSGEN_CERTS=1, SYSGEN_CMD=1, SYSGEN_CMEM=1, SYSGEN_CNG_CORE=1,
SYSGEN_COMMCTRL=1, SYSGEN_COMMCTRL_ANIMATE=1, SYSGEN_COMMCTRL_LINK=1,
SYSGEN_COMMDLG=1, SYSGEN_CONFIGMGR=1, SYSGEN_CONNMC=1, SYSGEN_CONNMGR2=1,
SYSGEN_CONSOLE=1, SYSGEN_CORELOC=1, SYSGEN_CORESTRA=1, SYSGEN_CPP_EH_AND_RTTI=1,
SYSGEN_CREDMAN=1, SYSGEN_CRYPTO=1, SYSGEN_CS=1, SYSGEN_CTLPNL=1, SYSGEN_CTLPNL2=1,
SYSGEN_CURSOR=1, SYSGEN_CXPORT=1, SYSGEN_DCOM=1, SYSGEN_DCOM_STG=1, SYSGEN_DDRAW=1,
SYSGEN_DEVICE=1, SYSGEN_DHCPSRV=1, SYSGEN_DISPLAY=1, SYSGEN_DMSAPI=1, SYSGEN_DMSRV=1,
SYSGEN_DNSAPI=1, SYSGEN_DOTNETV35=1, SYSGEN_DOTNETV35_SUPPORT=1, SYSGEN_DSHOW=1,
SYSGEN_DSHOW_ACMWRAP=1, SYSGEN_DSHOW_AVI=1, SYSGEN_DSHOW_CAPTURE=1,
SYSGEN_DSHOW_COLOR=1, SYSGEN_DSHOW_DISPLAY=1, SYSGEN_DSHOW_DMO=1,
SYSGEN_DSHOW_ERRORS=1, SYSGEN_DSHOW_HTTPSTREAMER=1, SYSGEN_DSHOW_ICM=1,
SYSGEN_DSHOW_IMAADPCM=1, SYSGEN_DSHOW_IMAGEDECODER=1,
SYSGEN_DSHOW_LOCALSTREAMER=1, SYSGEN_DSHOW_MIDIPARSER=1,
SYSGEN_DSHOW_MIDISYNTH=1, SYSGEN_DSHOW_MP3=1, SYSGEN_DSHOW_MPEG2DEMUX=1,
SYSGEN_DSHOW_MPEGA=1, SYSGEN_DSHOW_MPEGSPLITTER=1, SYSGEN_DSHOW_MPEGV=1,
SYSGEN_DSHOW_MSADPCM=1, SYSGEN_DSHOW_MSG711=1, SYSGEN_DSHOW_MSGSM610=1,
SYSGEN_DSHOW_MSRLE=1, SYSGEN_DSHOW_URLRDR=1, SYSGEN_DSHOW_VMR=1,
SYSGEN_DSHOW_WAV=1, SYSGEN_DSHOW_WAVEOUT=1, SYSGEN_DSHOW_WMA=1,
SYSGEN_DSHOW_WMA_VOICE=1, SYSGEN_DSHOW_WMP=1, SYSGEN_DSHOW_WMT=1,
SYSGEN_DSHOW_WMT_ASXV1=1, SYSGEN_DSHOW_WMT_ASXV2=1, SYSGEN_DSHOW_WMT_ASXV3=1,
SYSGEN_DSHOW_WMT_DRMOCX=1, SYSGEN_DSHOW_WMT_HTTP=1, SYSGEN_DSHOW_WMT_LOCAL=1,
SYSGEN_DSHOW_WMT_MMS=1, SYSGEN_DSHOW_WMT_MULTI=1, SYSGEN_DSHOW_WMT_NSC=1,
SYSGEN_DSHOW_WMT_WMDRM10PD=1, SYSGEN_DSHOW_WMV=1, SYSGEN_DSHOW_XDRMREMOTE=1,
SYSGEN_DSPLINK=1, SYSGEN_DSTSVC=1, SYSGEN_EAP=1, SYSGEN_EAPHOST=1, SYSGEN_EDB=1,
SYSGEN_ETHERNET=1, SYSGEN_ETH_80211_NWIFI=1, SYSGEN_EXFAT=1, SYSGEN_FATFS=1,
SYSGEN_FIBER=1, SYSGEN_FMTMSG=1, SYSGEN_FONTS_COUR_1_30=1, SYSGEN_FONTS_SYMBOL=1,
SYSGEN_FONTS_TAHOMA=1, SYSGEN_FONTS_TIMES=1, SYSGEN_FONTS_WINGDING=1,
SYSGEN_FSDBASE=1, SYSGEN_FSPASSWORD=1, SYSGEN_FSRAMROM=1, SYSGEN_FSREGHIVE=1,
SYSGEN_FSREPLBIT=1, SYSGEN_FULL_CRT=1, SYSGEN_GDI_ALPHABLEND=1,
SYSGEN_GESTUREANIMATION=1, SYSGEN_GRADFILL=1, SYSGEN_H264_DECODE_FILTER=1,
SYSGEN_H264_ENCODE_FILTER=1, SYSGEN_HTTPD=1, SYSGEN_IABASE=1, SYSGEN_IE7_COORDCONV=1,
SYSGEN_IE7_FLASH10_1=1, SYSGEN_IE7_GESTURESUPPORT=1, SYSGEN_IE7_IEEXR=1,
SYSGEN_IE7_IEFRAME=1, SYSGEN_IE7_IMAGESOURCE=1, SYSGEN_IE7_IMGCACHE=1,
SYSGEN_IE7_IMGUTIL=1, SYSGEN_IE7_INETCPL_EXR=1, SYSGEN_IE7_JSCRIPT=1, SYSGEN_IE7_MLANG=1,
SYSGEN_IE7_MOBILEWORKINGMODE=1, SYSGEN_IE7_MSLS=1, SYSGEN_IE7_PANELDETECTION=1,
SYSGEN_IE7_PNGFILT=1, SYSGEN_IE7_SHLWAPI=1, SYSGEN_IE7_URLMON=1, SYSGEN_IE7_VBSCRIPT=1,
SYSGEN_IE7_WININET=1, SYSGEN_IMAGING=1, SYSGEN_IMAGING_BMP_DECODE=1,
SYSGEN_IMAGING_BMP_ENCODE=1, SYSGEN_IMAGING_GIF_DECODE=1,
SYSGEN_IMAGING_GIF_ENCODE=1, SYSGEN_IMAGING_ICO_DECODE=1,
SYSGEN_IMAGING_JPG_DECODE=1, SYSGEN_IMAGING_JPG_ENCODE=1,
SYSGEN_IMAGING_PNG_DECODE=1, SYSGEN_IMAGING_PNG_ENCODE=1,
SYSGEN_IMAGING_TIFF_DECODE=1, SYSGEN_IMAGING_TIFF_ENCODE=1, SYSGEN_IMM=1,
SYSGEN_INPUTSCOPES=1, SYSGEN_IPHLPAPI=1, SYSGEN_JSCRIPT=1, SYSGEN_LAP_PSWD=1,
SYSGEN_LASS=1, SYSGEN_LOCALAUDIO=1, SYSGEN_MEDIAAPPS_MEDIALIBRARY=1,
SYSGEN_MEDIAAPPS_MEDIARENDERER=1, SYSGEN_MEDIAAPPS_WIC=1,
SYSGEN_MEDIAAPPS_WMPOCX=1, SYSGEN_MEDIARENDERER_MEDIAPLAYER=1, SYSGEN_MINGDI=1,
SYSGEN_MINGWES=1, SYSGEN_MINICOM=1, SYSGEN_MININPUT=1, SYSGEN_MINWMGR=1,

SYSGEN_MODEM=1, SYSGEN_MPEG2_DECODE_FILTER=1, SYSGEN_MPEG4_DECODE_FILTER=1,
SYSGEN_MPEG4_ENCODE_FILTER=1, SYSGEN_MSGQUEUE=1, SYSGEN_MSIM=1, SYSGEN_MSPART=1,
SYSGEN_MSXML_DOM=1, SYSGEN_MSXML_MIMEVIEWER=1, SYSGEN_MSXML_MINI=1,
SYSGEN_MSXML_XQL=1, SYSGEN_MSXML_XSLT=1, SYSGEN_MULTIUI=1, SYSGEN_MUSIC_PLAYER=1,
SYSGEN_NDIS=1, SYSGEN_NDISUIO=1, SYSGEN_NETUTILS=1, SYSGEN_NETWORK_POLICY=1,
SYSGEN_NKCOMPR=1, SYSGEN_NKMAPFILE=1, SYSGEN_NKTZINIT=1, SYSGEN_NLED=1,
SYSGEN_NOTIFY=1, SYSGEN_OBEX_CLIENT=1, SYSGEN_OBEX_FILEBROWSER=1,
SYSGEN_OBEX_SERVER=1, SYSGEN_OLE=1, SYSGEN_OLE_GUIDS=1, SYSGEN_OLE_STG=1,
SYSGEN_OSSVCS=1, SYSGEN_PHOTO_VIEWER=1, SYSGEN_PHYSICSENGINE=1, SYSGEN_PM=1,
SYSGEN_POWERVR=1, SYSGEN_PPP=1, SYSGEN_PRINTING=1, SYSGEN_PVR_SGXCOREREV_121=1,
SYSGEN_PWORD=1, SYSGEN_RDP=1, SYSGEN_RDP_AUDIO=1, SYSGEN_RDP_CLIPBOARD=1,
SYSGEN_RDP_DRIVE=1, SYSGEN_RDP_LICINFO=1, SYSGEN_RDP_REMOTEFX=1, SYSGEN_RDP_UI=1,
SYSGEN_REDIR=1, SYSGEN_RELFSD=1, SYSGEN_RPCRT4=1, SYSGEN_SDBUS=1, SYSGEN_SD_MEMORY=1,
SYSGEN_SECUREWIPE=1, SYSGEN_SERDEV=1, SYSGEN_SERVICES=1, SYSGEN_SHELL=1,
SYSGEN_SMARTCARD=1, SYSGEN_SNDSCHEMECPL=1, SYSGEN_SOAPTK_CLIENT=1, SYSGEN_SOFTKB=1,
SYSGEN_SQLCOMPACT=1, SYSGEN_SQLCOMPACTMP=1, SYSGEN_STANDARDSHELL=1,
SYSGEN_STATE_NOTIFICATIONS=1, SYSGEN_STDIO=1, SYSGEN_STDIOA=1, SYSGEN_STOREMGR=1,
SYSGEN_STOREMGR_CPL=1, SYSGEN_STREAMAUDIO=1, SYSGEN_STREAMAV=1, SYSGEN_STRSAFE=1,
SYSGEN_SYSCSPS=1, SYSGEN_SYSCSPS_SNDSCHEMECSP=1, SYSGEN_TAPI=1, SYSGEN_TCPIP=1,
SYSGEN_TFAT=1, SYSGEN_TIMEZONES=1, SYSGEN_TIMM=1, SYSGEN_TOOLHELP=1, SYSGEN_TOUCH=1,
SYSGEN_TOUCHGESTURE=1, SYSGEN_TSKSCHCSP=1, SYSGEN_UIPROXY=1, SYSGEN_UNIMODEM=1,
SYSGEN_UPNP=1, SYSGEN_UPNP_AV=1, SYSGEN_UPNP_AV_CTRL=1, SYSGEN_UPNP_AV_DEVICE=1,
SYSGEN_UPNP_CTRL=1, SYSGEN_UPNP_DEVICE=1, SYSGEN_USB=1, SYSGEN_USBFN=1,
SYSGEN_USBFN_ETHERNET=1, SYSGEN_USBFN_SERIAL=1, SYSGEN_USB_HID=1,
SYSGEN_USB_HID_CLIENTS=1, SYSGEN_USB_HID_KEYBOARD=1, SYSGEN_USB_HID_MOUSE=1,
SYSGEN_USB_PRINTER=1, SYSGEN_USB_STORAGE=1, SYSGEN_VBSCRIPT=1, SYSGEN_VEM=1,
SYSGEN_VIDEO_PLAYER=1, SYSGEN_WCELOAD=1, SYSGEN_WIFICPL=1, SYSGEN_WININET=1,
SYSGEN_WINSOCK=1, SYSGEN_XAMLIM=1, SYSGEN_XAML_RUNTIME=1, SYSGEN_XMLLITE=1

Appendix C – Known Functional and Feature Limitations

At the time of writing there was certain known functional and feature limitations which are listed below. We are constantly working to improve the level of Windows CE support for our products so please contact us if you need something on this list as me may already have it available.

Issue	Description	Resolution Possible	Date Planned
1	VIP1 module is only supported by RM2+HB2 running Windows Embedded compact 7.	Yes	TBA
2	Windows CE 7 currently has no WiFi support	Yes	TBA