E2ISP DLL

Using the DLL in your applications

LAST UPDATED : 13 October 2004

Embedded Results

Table of Contents

Embedded Results

1 Introduction

The E2ISP DLL allows the E2 user to add ISP functionality to their own applications. This document explains how this can be achieved with a number of different programming languages.

1.1 Device Support

24C00	24C01	24WC01	24C02	24WC02
24C04	24WC04	24C08	24WC08	24C16
24WC16	24C32	24WC32	24C64	24WC64
24C128	24WC128	24C256	24WC256	24C512
24C1024				
25C010	25010	25C01	25C020	25020
25C02	25C040	25C04	25040	25C080
25C08	25C160	25C16	25C320	25C32
25C640	25C64	25C128	25C256	25256
93C06	93C46	93C56	93C57	93C66
93C76	93C86			

1.2 Driver Installation

You need to be logged on as a user with Administrator privileges to install the software under Windows NT/2000/XP.

1.3 Programming Language Support

The DLL should work in any Windows programming language that can call DLL files using the stdcall method, whether dynamically linked or statically linked.

The E2ISP DLL has support files for Borland Delphi, Microsoft Visual C++, Borland C++ Builder and Microsoft Visual Basic.

Support for other programs is available, provided they accept ANSI C.

2

Header Support Files

2.1 Borland Delphi

To static link the dll for use with Borland Delphi you will need to add the e2isp.pas file to your uses section. This is all that is required for the dll to be usable within Delphi.

The files to do this are available in the dlldefinitionfiles\Delphi directory.

2.2 Borland C++ Builder

To static link the dll for use with Borland C++ Builder you will need to include the e2isp.h file in any cpp file that you call a dll function from.

You will also require the .lib file linked into the project. This will then allow the dll to be accessed by the project.

The files to do this are available in the dlldefinitionfiles\C++Builder directory.

2.3 Microsoft Visual C++

To static link the dll for use with Microsoft Visual C++ you will need to include the e2isp.h into the headers section of the project and link the .lib file. The files to do this are available in the dlldefinitionfiles\VisualC++ directory.

2.4 Microsoft Visual Basic

The dll can be accessed from Visual basic using the defines declared in the file provided.

The files to do this are available in the dlldefinitionfiles\VisualBasic directory.

2.5 Labview and other programs

Most programs that can use stdcall
and ANSI C can be used with the DLL.

3

Exported Functions

All functions return a non-zero value on failure.

E2_IdentifyFile(FileName : LPTSTR) : Integer;

This function returns the type of hex file that is being used, the return value can be one of the following.

-2	File empty or binary
-1	File not found
1	(*.hex) Intel Intellex 8/MDS
2	(*.hex) Intel MCS-86
3	(*.hex) Intel Hex-32
4	(*.s*) Motorola EXORciser
5	(*.s*) Motorola EXORmacs
6	(*.s*) Motorola 32-bit
7	Extended Tektronix
8	Tektronix
9	(*.bin) Binary file

E2_SetDevice(DeviceName : LPTSTR) : LongWord;

This function sets the device name that you require, a non-zero return value indicates device not found.

E2_SetMode(AMode : Byte) : LongWord;

This function sets the programming mode that will be used for programming.

1	COM	serial dongle programming
2	LPT	parallel dongle programming

E2_SetPort(APort : Byte) : LongWord;

This function sets the actual port that will be used for programming.

1	LPT1	COM1
2	LPT2	COM2
3	LPT3	COM3
4	LPT4	COM4

This function returns the number 17 if the port is not available.

Embedded Results

3

Exported Functions

E2_GetDevice(DeviceNum : LongWord; DeviceName : LPTSTR) : LongWord;

This function returns the device name in the Device name variable. The DeviceNum is a number from 0 to n, n depending on the number of devices that are supported.

E2_GetDeviceCount : LongWord;

This function returns the number of devices that are supported by the programmer

E2_EraseAllDevice:LongWord;

This function will bulk erase the device. *This will only work on the 93C devices*

E2_ReadEEPROM(FileName:LPTSTR; FileType:Byte):LongWord;

This function reads the eeprom to the given filename, of the file type, see the file types in identify file function for more information.

E2_ProgramEEPROM(FileName:LPTSTR; FileType:Byte):LongWord;

This function programs the eeprom with the file provided in filename, of file type, see the file types in identify file function for more information.

E2_VerifyEEPROM(FileName:LPTSTR; FileType:Byte):LongWord;

This function verifies the eeprom with the file provided in filename, of file type, see the file types in identify file function for more information.

E2_WriteProgMode(BT:Byte):LongWord;

This function sets the programming mode for the 93C devices, the options are:

08 = x8 mode
16 = x16 mode

3

Exported Functions

E2_WriteStatus(BT:Byte):LongWord;

Sets the status byte on the 25c devices.

See 25c datasheets for more information.

E2_ReadStatus:byte;

This function returns the current settings of the status register.

E2_ProgramWriteAll(BT:Byte):LongWord;

Writes a value to all of the eeprom. This function only works on the 93C devices.

E2_SetResetPolarity(BT:Byte):LongWord

Sets the state of the MCU reset line from the dongle so that an MCU that may be connected to the eeprom can be placed into reset during programming operations.

4

Programming the Device

Full examples are given in the documentation that allow you to create custom programming applications in different languages. Other functions that are not covered in this introduction are added from time to time. Again they are decribed fully in the documentation. Whatever your application, this DLL version of the E2 programming software is flexible enough to accommodate it

5

Support Information

Technical Support for the E2ISP DLL is available from:

Embedded Results UK:
support@kanda.com

If you have a bug report then please send this to the following address:
support@kanda.com

Please be sure to add the software version number that you suspect you have found a bug in and the operation that you performed before you detected the bug. It may also be useful to send the e2isp.log file that can be located in the same directory as the applications executable.

For sales queries please email:
sales@kanda.com

6

Notes

VB defines have been fully tested

The Visual C++ headers have been tested

The Borland C++ headers have been tested

Errorcodes:

0	- Operation OK
1	- Invalid Hardware
2	- Invalid Device
3	- Hardware Not open
4	- ISP Exception / Interface Connection Timeouts etc.
5	- Target Not Found.
6	- Verification Failed.
7	- Device is not supported on the programming system specified.
10	- File does not exist
11	- File appears corrupt
12	- File larger than available buffer size
16	- Could not create output file. (permissions, filename etc.)
99	- Unknown Problem..