ds30 Loader API reference

The ds30 Loader package comes with a gui. But the core functions can used without the supplied gui using ds30 Loader.dll. This document covers the API (application programming interface) of the ds30 Loader dll. The API consists of numerous classes that each contains their own functions and datatypes. These are described in this document.

Table of contents

Introduction

ds30 Loader

ds30 Loader is an open source bootloader for PIC18, PIC24 and dsPIC30 families of MCUs from Micropchip. It supports all devices in each family out of the box (those in production), only minor adjustments need to be done in firmware. The firmware is written in asm/asm30 and comes with a preconfigured MPLAB-project. The GUI is written in C#.

Prerequisites and Requirements

ds30 Loader is written in C# for .NET 2.0. During development, Microsoft Visual C# 2008 has been used. There is a free version available for download. It is called Visual C# 2008 express edition. See appendix A for a link.

Mono / MonoDevelop is a free alternative to .NET and Visual Studio. It has successfully been tested with ds30 Loader under both Windows and Linux.

Some components of the GHelper library are used by ds30 Loader. This library is not covered in this document.

Trademarks

All rights to copyrights, registered trademarks, and trademarks reside with their respective owners.

Class overview

ds30 Loader consist of numerous classes, these are describe briefly in this section.

clsDevice

This class holds the properties of a single Microchip device, for instance PIC18F2550.

clsDeviceFamily

This class holds the properties of a single Microchip device family such as dsPIC33F. Instances of clsDevice are members of this class

clsDevices

This class holds instances of the clsDeviceFamily class. This class is static.

clsDownloadingEventArgs

This class is supplied to the clsds30Loader.Download event and contains information about that event.

clsDownloadSettings

An instance of this class must be initialized with the wanted settings and then supplied to clsds30Loader.Download().

clsds30Loader

This is the main class that contains all vital code to do the actual download. This class is static.

clsHex

This is the base class representing a single hex-file. It contains basic functions and properties.

clsHexXXX

These classes are derived from clsHex and contains specific code to different Microchip device families.

clsHexFileParseEventArgs

This class is supplied to the clsHex.HexFileParse event and contains information about that event.

clsHexFileValidateEventArgs

This class is supplied to the clsHex.HexFileValidate event and contains information about that event.

clsParseSettings

An instance of this class must be initialized with the wanted settings and then supplied to clsHexXXX.Parse().

Class details

clsDevice

Datatypes & enums

None

Functions

None

Variables, properties and constants

public int configCount
Number of configuration words

public int eepromSizeB
Size of eeprom in bytes

public int eepromSizeW
Size of eeprom in words. This may or may not have the same value as eepromSizeB depending of eeprom word size.

public int eepromStartAddress
Address in memory where eeprom is mapped

public clsDeviceFamily family
Devicefamily this device belongs to

public int flashSizeP
Size of the flashmemory in pages

public int id
Identification number

public string name
Device name

public int pageSizeR
Size of a page in rows

public int rowsizeW
Size of a row in words.

Events

None

clsDeviceFamily

Datatypes & enums

None

Functions

DeviceAdd

Adds a device.

Definition:
```
public void DeviceAdd( clsDevice pobjDevice, ref bool pbResult )
```

Example code:
```
clsDevice objDevice = new clsDevice( objDeviceFamily, 301, "1220", 0x1000,
256, 16, 8, 4 );
bool bDeviceAddResult = false;
objDeviceFamily.DeviceAdd( objDevice, ref bDeviceAddResult );
```

DeviceExists

Returns true if a device with the specified id exist.

Definition:
```
public bool DeviceExists( int piID )
```

Example code:
```
bool bDeviceExist = clsDevices.DeviceExists( 10 );
```

DeviceGet

Returns the specified device. Null is returned if it does not exist.

Definitions:
```
public clsDevice DeviceGet( string pstrDeviceName )
public clsDevice DeviceGet( int piID )
```

Example code:
```
clsDevice objDevice = objDeviceFamily.DeviceGet( "pic18f2550" );
if ( objDevice == null ) {
    MessageBox.Show( "Device not found" );
}
```

DevicesGet
Returns the hashtable containing all devices

Definition:
```
public Hashtable DevicesGet()
```

Example code:
```
Hashtable htDevices = objDeviceFamily.DevicesGet();
```

Variables, properties and constants

public string name
Name of the devicefamily

public int pageSizeR
Size of a page in rows

public int rowSizeW
Size of row in words

Events
None

clsDevices

Datatypes & enums

None

Functions

DeviceFamilyAdd

Adds a devicefamily.

Definition:
```
static public void DeviceFamilyAdd(
    clsDeviceFamily pobjDeviceFamily,
    ref bool pbResult
)
```

Example code:
```
clsDeviceFamily objDeviceFamily = new clsDeviceFamily( "PIC18F", 2, 16);
bool bDeviceFamilyAddResult = false;
clsDevices.DeviceFamilyAdd( objDeviceFamily, ref bDeviceFamilyAddResult );
```

DeviceFamilyExists

Returns true if the specified device family exist.

Definition:
```
static public bool DeviceFamilyExists( string pstrDeviceFamilyName )
```

Example code:
```
bool bDeviceFamilyExist = clsDevices.DeviceFamilyExists( "dsPIC33F" );
```

DeviceFamilyGet

Returns the specified device family. Null is returned if it does not exist.

Definition:
```
static public clsDeviceFamily DeviceFamilyGet(string pstrDeviceFamilyName)
```

Example code:
```
clsDeviceFmily objDeviceFamily = objDevices.DeviceFamilyGet( "PIC24H" );
if ( objDeviceFamily == null ) {
    MessageBox.Show( "Devicefamily not found" );
}
```

DeviceFamiliesGet

Returns the hashtable containing all devicefamilies.

Definition:
```
static public Hashtable DeviceFamiliesGet()
```

Example code:
```
Hashtable htDeviceFamilies = objDevices.DeviceFamiliesGet();
```

DeviceGet

Returns the specified device if found in any of the member devicefamilies. Null is returned if not found

Definitions:
```
static public clsDevice DeviceGet( string pstrDeviceName )
static public clsDevice DeviceGet( int piID )
```

Example code:
```
clsDevice objDevice = clsDevices.DeviceGet( "2550" );
if ( objDevice == null ) {
    MessageBox.Show( "Device not found" );
}
```

Init

Adds all supported devices and devicefamilies.

Definition:
```
static public void Init()
```

Example code:
```
clsDevices.Init();
```

Variables, properties and constants

None

Events

None

clsDownloadingEventArgs

Datatypes & enums

DownloadingEventType
This enum is used by the Downloading event to tell what is going.

DownloadingEventType.Started
Tells that a download is starting.

DownloadingEventType.Info
Generic information about the download process.

DownloadingEventType.Success
The last operation was succesfull.

DownloadingEventType.Error
The last operation failed. Download is aborted

DownloadingEventType.Warning
The user should be notified about a problem.

DownloadingEventType.Completed
The download completed.

DownloadingEventType.ProgressStarted
Some sort of progress is started. Used to show a progressbar.

DownloadingEventType.Progress
Progress value has changed.

DownloadingEventType.ProgressEnded
Progress ended. Used to hide a progressbar.

Functions

Constructor

Definition:
```
public clsDownloadingEventArgs(
    EventType pEventType,
    string pstrMessage,
    int piTabLevel
)
```

Example code:

```
clsDownloadingEventArgs objDownloadingEventArgs = new
clsDownloadingEventArgs( EventType.started, "Downloading...", 0 );
```

Variables, properties and constants

public EventType eventType
The eventtype.

public string message
Message to provide additional information about the event.

public int tabLevel
Tablevel of message.

Events
None

clsDownloadSettings

Datatypes & enums

None

Functions

None

Variables, properties and constants

public bool activateDTR
Gets or sets device activation by dtr.

public bool activateRTS
Gets or sets device activation by rts.

public bool noGoto
When set 0x00 goto to bootloader is not written.

public bool polltime
Gets or sets polltime in ms, default is 100.

public bool timeout
Gets or sets timeout time in ms, default is 5000.

public bool resetDtr
Gets or sets device reset by dtr.

public bool resetRts
Gets or sets device reset by rts.

public bool resetTime
Time that dtr or rts is held high for device reset.

public bool writeConfigs
Gets or sets whether configs are to be downloaded or not.

public bool writeEeprom
Gets or sets whether eeprom data are to be downloaded or not.

public bool writeProgram
Gets or sets whether ordinary flash data are to be downloaded or not.

Events

None

clsds30Loader

Datatypes & enums

None

Functions

Abort()

Aborts a download in progress.

Definition:

```
static public void Abort()
```

Example code:

```
if ( userPressedAbortButton )
    clsds30Loader.Abort()
```

Download()

Initiates a download to the device.

Definition

```
static public void Download (
    clsDevice pobjDevice,
    clsSerialPort pobjPort,
    clsHex pobjHex,
    clsds30LSettings pobjSettings,
    int iTabLevel,
    ref bool pbResult
)
```

Example code

```
clsDevice objDevice = clsDevices.DeviceGet( "2550" );
clsSerialPort objPort = new clsSerialPort();
clsHex18F objHex = new clsHex18F( "c:\testapp.hex" );
clsds30LSettings objSettings = new clsds30LSettings();

objPort.Setup( ”COM1”, 115200 );
objSettings.writeProgram = true;

bool bDownloadResult = false;
clsds30Loader.Download( objDevice, objPort, objHex, objSettings, 0, ref
bDownloadResult );
```

FindLoader()

Searches for the bootloader by sending hello and polling for an answer.

Definition

```
static public void FindLoader(
    clsDevice pobjDevice,
    clsSerialPort pobjPort,
    clsds30LSettings pobjSettings,
    int iTabLevel,
    ref bool pbResult
)
```

Example code

```
clsDevice objDevice = clsDevices.DeviceGet( "2550" );
clsSerialPort objPort = new clsSerialPort();
clsds30LSettings objSettings = new clsds30LSettings();

objPort.Setup( ”COM1”, 115200 );

bool bFindLoaderResult = false;
clsds30Loader.FindLoader( objDevice, objPort, objSettings, 0, ref
bFindLoaderResult );
```

Variables, properties and constants

bool debugMode

When set to true additional debug information is outputted via the Download event.

string strVersion

This string contains the version number of the ds30 Loader library.

Events

Downloading

This event is raised during download containing information that may be presented to the user.

Definition

```
public delegate void DownloadingDelegate( object sender,
clsDownloadingEventArgs e );
```

Example code

```
clsds30Loader.Downloading += new clsds30Loader.DownloadingDelegate(
ds30L_Downloading );

private void ds30L_Downloading( object obj, clsDownloadingEventArgs e )
{
    if ( e.eventType == DownloadingEventType.error ) {
        MessageBox.Show( ”Download failed: ” + e.message );
    } else if ( e.eventType == DownloadingEventType.success ) {
        MessageBox.Show( ”Download succeeded” );
    }
}
```

clsHex

Datatypes & enums

None

Functions

BytesToWrite

Returns the number of bytes that will be written with the specified settings.

Definition

```
virtual public int BytesToWrite(
    clsDevice pobjDevice,
    bool pbWriteProgram,
    bool pbWriteEEPROM,
    bool pbWriteConfigs
)
```

Example code

```
clsHex18F objHex = new clsHex18F( "c:\\test.hex" );
clsDevice objDevice = clsDevices.DeviceGet( "2550" );
int iBytesToWrite = objHex.BytesToWrite( objDevice, true, false, false );
```

ParseHexFile

Parses the hex-file specified by the filename property.

Definition

```
virtual public void ParseHexFile(
    clsDevice pobjDevice,
    int piBootloaderSizeR,
    bool pbNoGoto,
    bool bIgnoreOverwrite,
    int iTabLevel,
    ref bool pbResult
)
```

Example code

```
clsHex18F objHex = new clsHex18F( "c:\\test.hex" );
clsDevice objDevice = clsDevices.DeviceGet( "2550" );
bool bParseResult = false;
objHex.ParseHexFile( objDevice, 5, false, false, 0, ref bParseResult );
```

ValidateHexFile

Performs a basic fileformat and checksom validation.

Definition

```
public void ValidateHexFile( int iTabLevel, ref bool pbResult )
```

Example code

```
clsHex18F objHex = new clsHex18F( "c:\\test.hex" );
bool bValidationResult = false;
objHex.ValidateHexFile( 0, ref bValidationResult);
```

Variables, properties and constants

public int configWordsUsed
public int eeWordsUsed
public string filename
public bool hasValidProgram
public bool hasValidEEPROM
public bool hasValidConfigs
public int progRowsUsed
public int progWordsUsed

Events

HexFileValidate

This event is raised during validation and contains information that may be presented to the user.

HexFileParse

This event is raised during parsing and contains information that may be presented to the user.

clsHexFileParseEventArgs

Datatypes & enums

public enum EventType
EventType.started
EventType.info
EventType.warning
EventType.failed
EventType.success

Functions

None

Variables, properties and constants

public EventType eventType
The eventtype.

public string message
Message to provide additional information about the event.

public int tabLevel
Tablevel of message.

Events

None

clsHexFileValidateEventArgs

Datatypes & enums

public enum EventType
EventType.started
EventType.failed
EventType.success

Functions

None

Variables, properties and constants

public EventType eventType
The eventtype.

public string message
Message to provide additional information about the event.

public int tabLevel
Tablevel of message.

Events

None

clsParseSettings

Datatypes & enums

None

Functions

CompareTo

Compares settings with another instance.

Definition

```
public bool CompareTo( clsParseSettings pobjSettings )
```

Example code

CopyFrom

Copies settings from another instance.

Definition

```
public void CopyFrom( clsParseSettings pobjSettings )
```

Example code

Variables, properties and constants

int bootloaderSizeR

Gets or sets bootloader size in rows.

clsDevice device

Gets or sets device.

DateTime fileTimeStamp

Gets or sets time of last parse.

bool ignoreOverwrite

When set, overwrite of bootloader is accepted.

bool noGoto

When set 0x00 goto to bootloader is not written.

Events

None

Appendix A – Links

ds30 Loader homepage
http://mrmackey.no-ip.org/elektronik/ds30loader/

Microsoft Visual C# express edition
http://www.microsoft.com/express/vcsharp/

Mono / MonoDevelop
http://www.mono-project.com