Project:	FP100
Title:	DLL Interface

Our ref.:	AS0548300
Date	5 January 2005
Keywords:	Windows 32-bit DLL 2de release
Author:	Richard Knoop
Status:	© 2005 Actum Solutions, open to customers

Newtonstraat 27
1704 SB Heerhugowaard
The Netherlands

Tel. +31 - (0)72 576 2555
Fax. +31 - (0)72 576 2559

url: http://www.actum.com

K.v.K. Alkmaar nr. 370.50256

Contents

Introduction

The FP100 can be used to load or change the software of a control unit. It programs the control modules using the ICP (In-Circuit Programming) that many micro-controller nowadays provide.

This document describes the DLL interface to access the FP100 without using the standard user interface.

The FP100 DLL runs on a 32-bit MS-Windows environment. Other platforms can be supplied upon request.

Interface functions

FP100Open

Description

Opens the communication to the FP100.

Syntax

FP100RC FP100Open(void)

Parameters

None

Return Value

FP100_OK if successful, otherwise the return value is an FP100 error code, see also Definitions.

Remarks

This is the first call to the FP100 functions.

Example

This example shows how to open the FP100.

```
// Load FP100 DLL
if (FP100Open() == FP100_OK)
{
  // start communicating with the FP100
}
else
{
  printf("FP100 can not be located.\n");
}
```

FP100Close

Description

Closes the communication to the FP100.

Syntax

FP100RC FP100Close(void)

Parameters

None

Return Value

FP100_OK if successful, otherwise the return value is an FP100 error code, see also Definitions.

Remarks

This is the last call to the FP100 functions before the DLL is unloaded.

Example

This example shows how to close the FP100 communication.

```
if (FP100Close() == FP100_OK)
{
  // unload FP100 DLL
  // Close your program
}
else
{
  printf("Error while closing the FP100 communication.\n");
}
```

FP100Reset

Description

Resets the FP100 status. All error codes will be reset.

Syntax

FP100RC FP100Reset(void)

Parameters

None

Return Value

FP100_OK if successful, otherwise the return value is an FP100 error code, see also Definitions.

Remarks

This function is used after an error occurred and before a next prepare or programming cycle is started.

Example

This example shows how to reset the FP100 status.

```
if (FP100Reset() == FP100_OK)
{
  // restart any tasks
}
else
{
  printf("Error while reseting the FP100 status.\n");
}
```

FP100Version

Description

Return the FP100 version

Syntax

FP100RC FP100Reset(unsigned long *version)

Parameters

version: returns the version of the FP100, 100H (== 256D) represents V1.00

Return Value

FP100_OK if successful, otherwise the return value is an FP100 error code, see also Definitions.

Remarks

This function is used after an error occurred and before a next prepare or programming cycle is started.

Example

This example shows how to reset the FP100 status.

```
unsigned long version;
if (FP100Version(&version) == FP100_OK)
{
  printf("Using FP100 V%ld.%02ld\n",version/100,version%100
}
else
{
  printf("Error while getting the FP100 version.\n");
}
```

FP100Status

Description

Returns the current status and progress of the FP100.

Syntax

FP100RC FP100Status(FP100STATUS *status, FP100PHASE *phase)

Parameters

Status: returns the current operating status of the FP100.

Phase: returns the (sub)phase when the FP100 is either preparing or programming.

Return Value

FP100_OK if successful, otherwise the return value is an FP100 error code, see also Definitions.

Remarks

When an error occurs this function will return the last error code, use FP100Reset to reset this.

Example

This example shows how to monitor the FP100 progress.

```
FP100PHASE phase;
FP100STATUS status;

if (FP100Status(&status,&phase) == FP100_OK)
{
  switch(status)
  {
    case FP100_IDLE:
      printf("all tasks are done.\n");
      break;
    case FP100_DISCONNECTED:
      printf("The FP100 is disconnected.\n");
      break;
    case FP100_PREPARING:
      printf("Preparing phase %ld.\n",phase);
      break;
    case FP100_PREPARE_ERROR:
      printf("Prepare error in phase %ld.\n",phase);
      break;
    case FP100_PROGRAMMING:
      printf("Programming phase %ld.\n",phase);
      break;
    case FP100_PROGRAM_ERROR:
      printf("Program error in phase %ld.\n",phase);
      break;
  }
}
```

FP100Prepare

Description

Loads a FP100 project file and start the prepare phase.

Syntax

FP100RC FP100Prepare(char *FP100Project)

Parameters

FP100Project:	contains a path to a FP100 project file
FP100Settings:	contains the path in which the FP100 targets are located

Return Value

FP100_OK if successful, otherwise the return value is an FP100 error code, see also Definitions.

Remarks

The FP100 project file is created using the FP100programmer. It contains all settings and references to the code files.

Example

This example shows how to start the prepare phase.

```
if (FP100Prepare("My Documents\\motion.fpp","c:\\program files\\fp100\\targets") == FP100_OK)
{
  // start monitoring the status until idle or error.
}
else
  printf("Unable to start the prepare phase.\n");
```

FP100Program

Description

Start the programming phase

Syntax

FP100RC FP100Program(void)

Parameters

None

Return Value

FP100_OK if successful, otherwise the return value is an FP100 error code, see also Definitions.

Remarks

The program phase uses the data stored in the FP100 during the prepare phase.

Example

This example shows how to start the program phase.

```
if (FP100Program() == FP100_OK)
{
  // start programming phase.
}
else
  printf("Unable to start the program phase.\n");
```

Definitions

Excerpts from the header file FP100.H are included in this Appendix to explain any references in the descriptions of the functions in this document.

FP100RC (unsigned long)

```
FP100_OK = 0
FP100_INVALID_HANDLE = 1
FP100_DEVICE_NOT_FOUND = 2
FP100_DEVICE_NOT_OPENED = 3
FP100_IO_ERROR = 4
FP100_INSUFFICIENT_RESOURCES = 5
FP100_INVALID_PARAMETER = 6
FP100_FP100_PROJECT_FILE_NOT_FOUND = 7
FP100_INVALID_FP100_PROJECT_FILE = 8
FP100_VERSION_MISMATCH = 9
FP100_INVALID_ARGS = 16
FP100_OTHER_ERROR = 17
```

FP100STATUS (unsigned long)

```
FP100_IDLE = 0
FP100_DISCONNECTED = 1
FP100_PREPARING = 2
FP100_PREPARE_ERROR = 3
FP100_PROGRAMMING = 4
FP100_PROGRAM_ERROR = 5
```

FP100PHASE (unsigned long)

```
FP100_PREPARE_ERASE_CODE = 0
FP100_PREPARE_WRITE_CODE = 1
FP100_PREPARE_VERIFY_CODE = 2
FP100_PREPARE_ERASE_DATA = 3
FP100_PREPARE_WRITE_DATA = 4
FP100_PREPARE_VERIFY_DATA = 5
FP100_PREPARE_WRITE_SETTING = 6
FP100_PREPARE_VERIFY_SETTING = 7

FP100_PREPARE_ERROR_CODEFILE = 20                  // can not open code file
FP100_PREPARE_ERROR_DATAFILE = 21                  // can not open data file
FP100_PREPARE_ERROR_SETTINGSFILE = 22              // can not open settings file
FP100_PREPARE_ERROR_ERASE = 23                     // unable to erase the FP100 memory
FP100_PREPARE_ERROR_PROGRAM = 24                   // unable to program the FP100 memory
FP100_PREPARE_ERROR_VERIFY = 25                    // unable to verify the FP100 memory
FP100_PREPARE_ERROR_PROGRAM_SETTINGS = 26          // unable to program the settings
FP100_PREPARE_ERROR_VERIFY_SETTINGS = 27           // unable to verify the settings

FP100_PROGRAM_VERIFY_DEVICE_ID = 100
FP100_PROGRAM_ERASE_OPTIONS = 101
FP100_PROGRAM_ERASE_CODE = 102
FP100_PROGRAM_BLANK_CHECK_CODE = 103
FP100_PROGRAM_PROGRAM_CODE = 104
FP100_PROGRAM_VERIFY_CODE = 105
FP100_PROGRAM_ERASE_DATA = 106
FP100_PROGRAM_BLANK_CHECK_DATA = 107
FP100_PROGRAM_PROGRAM_DATA = 108
FP100_PROGRAM_VERIFY_DATA = 109
FP100_PROGRAM_PROGRAM_OPTIONS = 110
FP100_PROGRAM_VERIFY_OPTIONS = 111
FP100_PROGRAM_RUN = 112
```

```
FP100_PROGRAM_ERROR_DEVICEID = 120                  // wrong device ID
FP100_PROGRAM_ERROR_READOUT_PROTECTION = 121        // unable to remove the read-out protection
FP100_PROGRAM_ERROR_ERASE_CODE = 122                // unable to erase the code
FP100_PROGRAM_ERROR_BLANK_CHECK_CODE = 123          // code memory is not blank
FP100_PROGRAM_ERROR_PROGRAM_CODE = 124              // unable to program the code
FP100_PROGRAM_ERROR_VERIFY_CODE = 125               // verify error
FP100_PROGRAM_ERROR_ERASE_DATA = 126                // unable to erase the code
FP100_PROGRAM_ERROR_BLANK_CHECK_DATA = 127          // code memory is not blank
FP100_PROGRAM_ERROR_PROGRAM_DATA = 128              // unable to program the code
FP100_PROGRAM_ERROR_VERIFY_DATA = 129               // verify error
FP100_PROGRAM_ERROR_PROGRAM_OPTIONS = 130           // unable to program the options
FP100_PROGRAM_ERROR_VERIFY_OPTIONS = 131            // verify error
FP100_PROGRAM_ERROR_ICP = 132                       // unable to setup the ICP communication
```