up_control.dll

UP_APP01 – up_control.dll description

Application note

ASIX s.r.o.
Na Popelce 38/17
150 00 Prague
Czech Republic

www.asix.net

support@asix.net

sales@asix.net

ASIX s.r.o. reserves the right to make changes to this document, the latest version of which can be found on the Internet.

ASIX s.r.o. renounces responsibility for any damage caused by the use of ASIX s.r.o. products.

Table of Contents

1

up_control.dll

1.1 Description

The up_control.dll enables user to control the UP software using functions contained in the library. It contains basic programming functions.

The library have to be in the same directory as the up.exe file.

First the process have to be configured using UP_ProgConfig function then any function working with the selected programmer can be called, it requests the UP to do the work and returns an error code.

The state of the operation can be checked using UP_ProgState or UP_ProgStateEx functions.

When finished, the error code of the operation can be read using UP_LastErrorCode function, the meaning of the returned error code is the same as the return codes returned by UP on the commandline, when not finished the UP_LastErrorCode returns -1.

The library can control up to 8 programmers at once. With increasing number of the controlled programmers the computer load will also grow up, which will affect programming speed.

UP_GetProgList function returns list of available programmers, calling of this function clears settings done by UP_ProgConfig function.

up_control64.dll is 64 bit version of up_control.dll, both of them are contained in the UP software installation directory.

1.2 List of the functions

```
int __stdcall UP_Prog(int prog_index, bool code,
 bool data, bool boot, bool cfg);
int __stdcall UP_DiffProg(int prog_index, bool c
ode, bool data, bool boot, bool cfg);
int __stdcall UP_Erase(int prog_index, bool code
, bool data, bool boot);
int __stdcall UP_BlankCheck(int prog_index, bool
 code, bool data, bool boot, bool cfg);
int __stdcall UP_Verify(int prog_index, bool cod
e, bool data, bool boot, bool cfg);
int __stdcall UP_Read(int prog_index, bool code,
 bool data, bool boot, bool cfg);

int __stdcall UP_ProgState(int prog_index, int *
ProgressBarValue);
int __stdcall UP_ProgStateEx(int prog_index, int
 *MainProgressBar, int *TaskProgressBar, int *Ta
sk);
int __stdcall UP_LastErrorCode(int prog_index);
int __stdcall UP_Cancel(int prog_index);
int __stdcall UP_ProgConfig(int prog_index, char
 *UP_project, int prog_type, int prog_SN, char *
NewDataFile, char *EEFile);
int __stdcall UP_SetManualSN(int prog_index, boo
l DefineSN, int SN);
int __stdcall UP_GetProgList(int prog_type, int
*sn_list, int count, int *count_returned);
int __stdcall UP_CleanUp(void);
```

Note: The UP_ProgConfig function expects that the strings pointed to by UP_project, NewDataFile and EEFile are ANSI strings.

1.3 Functions description

1.3.1 UP_Prog

The function asks for programming of the connected device.

The UP_ProgConfig function have to be called first to define parameters.

Function definition:

int __stdcall UP_Prog(int prog_index, bool code, bool data, bool boot, bool cfg);

Parameters:

prog_index - Index of the selected programmer.

code - When true, it programs code memory.

data - When true, it programs data memory.

boot - When true, it programs boot memory.

cfg - When true, it programs configuration memory.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_UP_MISSING - The library was not able to find up.exe file.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

ERR_NOT_CONFIGURED - The UP_ProgConfig has not been called first or it has not been successful.

Example:

```
FuncRes = UP_Prog(0, 1, 1, 1, 1); // With progra
mmer 0 program all available memories.
```

1.3.2 UP_DiffProg

The function asks for differential programming of the connected device.

The UP_ProgConfig function have to be called first to define parameters.

Function definition:

int __stdcall UP_DiffProg(int prog_index, bool code, bool data, bool boot, bool cfg);

Parameters:

prog_index - Index of the selected programmer.

code - When true, it programs code memory.

data - When true, it programs data memory.

boot - When true, it programs boot memory.

cfg - When true, it programs configuration memory.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_UP_MISSING - The library was not able to find up.exe file.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

ERR_NOT_CONFIGURED - The UP_ProgConfig has not been called first or it has not been successful.

Example:

```
FuncRes = UP_DiffProg(0, 1, 1, 1, 1); // With pr
ogrammer 0 program all available memories.
```

1.3.3 UP_Erase

The function asks for erasing of the connected device.

The UP_ProgConfig function have to be called first to define parameters.

Function definition:

int __stdcall UP_Erase(int prog_index, bool code, bool data, bool boot);

Parameters:

prog_index - Index of the selected programmer.

code - When true, it erases code memory.

data - When true, it erases data memory.

boot - When true, it erases boot memory.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_UP_MISSING - The library was not able to find up.exe file.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

ERR_NOT_CONFIGURED - The UP_ProgConfig has not been called first or it has not been successful.

Example:

```
FuncRes = UP_Erase(0, 1, 1, 1); // With programmer 0 erase all available memories.
```

1.3.4 UP_BlankCheck

The function asks for blank check of the connected device.

The UP_ProgConfig function have to be called first to define parameters.

Function definition:

int __stdcall UP_BlankCheck(int prog_index, bool code, bool data, bool boot, bool cfg);

Parameters:

prog_index - Index of the selected programmer.

code - When true, it does blank check of code memory.

data - When true, it does blank check of data memory.

boot - When true, it does blank check of boot memory.

cfg - When true, it does blank check of configuration memory.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_UP_MISSING - The library was not able to find up.exe file.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

ERR_NOT_CONFIGURED - The UP_ProgConfig has not been called first or it has not been successful.

Example:

```
FuncRes = UP_DiffProg(1, 1, 0, 0, 0); // With programmer 1 blank check code memory.
```

1.3.5 UP_Verify

The function asks for verification of the connected device.

The UP_ProgConfig function have to be called first to define parameters.

Function definition:

int __stdcall UP_Verify(int prog_index, bool code, bool data, bool boot, bool cfg);

Parameters:

prog_index - Index of the selected programmer.

code - When true, it verifies code memory.

data - When true, it verifies data memory.

boot - When true, it verifies boot memory.

cfg - When true, it verifies configuration memory.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_UP_MISSING - The library was not able to find up.exe file.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

ERR_NOT_CONFIGURED - The UP_ProgConfig has not been called first or it has not been successful.

Example:

```
FuncRes = UP_Verify(2, 1, 1, 0, 0); // With programmer 2 verify code and data memories.
```

1.3.6 UP_Read

The function asks for reading of the connected device.

The UP_ProgConfig function have to be called first to define parameters.

Function definition:

int __stdcall UP_Read(int prog_index, bool code, bool data, bool boot, bool cfg);

Parameters:

prog_index - Index of the selected programmer.

code - When true, it verifies code memory.

data - When true, it verifies data memory.

boot - When true, it verifies boot memory.

cfg - When true, it verifies configuration memory.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_UP_MISSING - The library was not able to find up.exe file.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

ERR_NOT_CONFIGURED - The UP_ProgConfig has not been called first or it has not been successful.

Example:

```
FuncRes = UP_Read(0, 1, 1, 1, 1); // With programmer 0 read all available memories.
```

1.3.7 UP_ProgState

The function returns state of the selected programmer.

Function definition:

int __stdcall UP_ProgState(int prog_index, int *ProgressBarValue);

Parameters:

prog_index - Index of the selected programmer.

ProgressBarValue - Returns value of UP software main ProgressBar.

Return values:

PROG_STATE_DONE - The last operation has beed

finished.

PROG_STATE_BUSY - The programmer is busy.

PROG_STATE_NOT_USED - The programmer has not been used yet.

PROG_STATE_WRONG_INDEX - The programmer index is out of range.

Example:

```
int ProgressBar;
FuncRes = UP_ProgState(0, &ProgressBar); // With
 programmer 0 read all available memories.
```

1.3.8 UP_ProgStateEx

The function returns state of the selected programmer including actual task and task ProgressBar.

Function definition:

int __stdcall UP_ProgStateEx(int prog_index, int *MainProgressBar, int *TaskProgressBar, int *Task);

Parameters:

prog_index - Index of the selected programmer.

MainProgressBar - Returns value of UP software main ProgressBar.

TaskProgressBar - Returns value of task ProgressBar.

Task - Returns an identification of the actual task in accordance with constants. In the lowest byte (byte 0) of the Task variable there is the actual task, in the second byte (byte 1) there is a memory identification. E.g. when the actual task is the code memory programming, then in the Task variable there is: TASK_MEM_CODE | TASK_FUNC_PROG

Return values:

PROG_STATE_DONE - The last operation has beed finished.

PROG_STATE_BUSY - The programmer is busy.

PROG_STATE_NOT_USED - The programmer has not been used yet.

PROG_STATE_WRONG_INDEX - The programmer index is out of range.

Example:

```
int ProgressBar;
int TaskProgressBar;
int Task;
FuncRes = UP_ProgStateEx(0, &ProgressBar, &TaskP
rogressBar, &Task); // Read values of ProgressBa
rs and the Task identification
```

1.3.9 UP_LastErrorCode

The function returns error code of the last operation finished with the programmer. The returned value is the same as UP software returns on the commandline.

Function definition:

int __stdcall UP_LastErrorCode(int prog_index);

Parameters:

prog_index - Index of the selected programmer.

Return values: The returned value is the same as UP software returns on the commandline. For more information see programmer manual chapter **Program Return Codes**. When not finished it returns -1.

Example:

```
FuncRes = UP_LastErrorCode(0); // Read the last
error code of programmer 0.
```

1.3.10 UP_Cancel

The function cancels the task which is being executed by the selected programmer, same as the Cancel button in the UP software.

Function definition:

int __stdcall UP_Cancel(int prog_index);

Parameters:

prog_index - Index of the selected programmer.

Return values:

ERR_NONE - The function has been successfully called.

ERR_NOT_WORKING - Programmer does nothing.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

Example:

```
FuncRes = UP_Cancel(0); // Cancel the task of pr
ogrammer 0.
```

1.3.11 UP_ProgConfig

The function configures parameters for following operations. This is the first function which should be called.

Function definition:

int __stdcall UP_ProgConfig(int prog_index, char *UP_project, int prog_type, int prog_SN, char *NewDataFile, char *EEFile);

Parameters:

prog_index - Index of the selected programmer.

UP_project - Selects ppr project file of the UP software.

prog_type - Selects programmer in accordance with constants.

prog_SN - Programmer serial number, when it is 0, the project file defined value is used.

NewDataFile - Sets data file which replaces the one defined in the project, same as UP software /df commandline parameter.

EEFile - Sets data file for data memory which replaces the one defined in the project file, e.g. for AVR. It does the same as UP software /e commandline parameter.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_UP_MISSING - The library was not able to find up.exe file.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

ERR_NOT_CONFIGURED - The UP_ProgConfig has not been called first or it has not been successful.

ERR_FILE_DOES_NOT_EXIST - The selected project file does not exist.

Example:

```
char ppr_path[] ="C:\\projects\
\PIC18F67J10.PPR";
char file_path[] =""; // data files from ppr
FuncRes = UP_ProgConfig(0, ppr_path, SET_PROG_FR
OM_PROJECT, 0, file_path, file_path);
```

1.3.12 UP_SetManualSN

The function sets manual SN. It is required when the use of the manual SN is defined in the ppr file.

Function definition:

int __stdcall UP_SetManualSN(int prog_index, bool DefineSN, int SN);

Parameters:

prog_index - Index of the selected programmer.

DefineSN - If true, the manual SN will be sent to UP. By default it is false.

SN - Defines the SN itself.

Return values:

ERR_NONE - The function has been successfully called.

ERR_PROG_BUSY - The programmer is busy.

ERR_WRONG_PROG_INDEX - The programmer index is out of range.

Example:

```
FuncRes = UP_SetManualSN(0, 1, 0x1234);
```

1.3.13 UP_GetProgList

The function returns list of available programmers, it returns the list only when no programmer is being used by the library. Calling of this function clears settings done by the UP_ProgConfig function.

Function definition:

int __stdcall UP_GetProgList(int prog_type, int *sn_list, int count, int *count_returned);

Parameters:

prog_type - Selects programmer type in accordance with constants.

sn_list - Array of integer, which returns the list of the serial numbers of the available FORTE programmers. The serial nubers are returned as 24bit values, same as they are listed in the UP software.

count - Variable defining the number of the serial numbers to be read.

count_returned - Variable returning number of serial numbers, which have been returned in sn_list.

Return values:

ERR_NONE - The function has been successfully executed.

ERR_PROG_BUSY - A programmer is busy.

ERR_DRIVER - The programmer driver error.

In **sn_list** the function returns list of available programmers and the count of the returned values is returned in **count_returned**.

Example:

```
int prog_list[20];
int prog_list_count;
int FuncRes;
FuncRes=UP_GetProgList(PROG_FORTE, prog_list, 20
, &prog_list_count);
```

1.3.14 UP_CleanUp

The function will end up the work of the library. This function has to be called before calling the FreeLibrary function.

Function definition:

int __stdcall UP_CleanUp(void);

Return values:

ERR_NONE - The function has been succesfully executed.

ERR_PROG_BUSY - A programmer is busy.

1.4 Constants

```
SET_PROG_FROM_PROJECT=0;
SET_PROG_PRESTO=1;
SET_PROG_FORTE=2;

PROG_PRESTO=1;
PROG_FORTE=2;
```

1.5 Functions error codes

```
ERR_NONE=0;
ERR_PROG_BUSY=1;
ERR_UP_MISSING=2;
ERR_WRONG_PROG_INDEX=3;
ERR_NOT_CONFIGURED=4;
ERR_FILE_DOES_NOT_EXIST=5;
ERR_DRIVER=6;
ERR_NOT_WORKING=7;
```

1.6 UP_ProgState return values

```
PROG_STATE_DONE=0;
PROG_STATE_BUSY=1;
PROG_STATE_NOT_USED=2;
PROG_STATE_WRONG_INDEX=3;
```

1.7 UP_ProgStateEx Task values

```
TASK_MEM_MASK=0xFF00;
TASK_FUNC_MASK=0xFF;

TASK_MEM_NONE=0;
TASK_MEM_CODE=0x100;
TASK_MEM_DATA=0x200;
TASK_MEM_BOOT=0x300;
TASK_MEM_CFG=0x400;
TASK_MEM_ID=0x500;
TASK_MEM_OTHER=0xFF00;

TASK_FUNC_NONE=0x00;
TASK_FUNC_READ=0x01;
TASK_FUNC_BLANK_CHECK=0x02;
TASK_FUNC_VERIFY=0x03;
TASK_FUNC_VERIFY1=0x04;
TASK_FUNC_VERIFY2=0x05;
TASK_FUNC_PROG=0x06;
TASK_FUNC_ERASE=0x07;
TASK_FUNC_ERASE_ALL=0x08;
TASK_FUNC_DIFF_PROG=0x09;
TASK_FUNC_FILE_LOAD=0x0A;
TASK_FUNC_OTHER=0xFF;
```

2

Document history

Document revision	Modifications made
2019-09-19	Document created.
2020-10-08	Added UP_GetProgList function and constants.
2021-05-21	The number of supported programmers has been changed.
2021-08-20	Added UP_CleanUp function.
2024-09-17	Added UP_ProgStateEx function.
2024-11-06	Added UP_Cancel function.