up_control.dll

UP_APP01 – popis up_control.dll

Aplikační poznámka

ASIX s.r.o.
Na Popelce 38/17
150 00 Praha 5 - Košíře

www.asix.cz

podpora@asix.cz

obchod@asix.cz

ASIX s.r.o. si vyhrazuje právo změny tohoto dokumentu, jehož aktuální podobu naleznete na Internetu.

ASIX s.r.o. nenese žádnou zodpovědnost za škody způsobené použitím produktu firmy ASIX s.r.o.

20.8.2021

Obsah

1

up_control.dll

1.1 Popis

Knihovna up_control.dll umožňuje uživateli ovládat software UP v ní obsaženými funkcemi. Knihovna obsahuje základní programovací funkce.

Knihovna musí být ve stejném adresáři jako soubor up.exe.

Nejprve je potřeba nastavit potřebné parametry funkcí UP_ProgConfig. Následně lze zavolat jakoukoliv z funkcí pracujících s programátorem a ta zavolá program UP k provedení požadované operace a vrátí chybový kód.

Stav operace může být sledován funkcí UP_ProgState.

Po dokončení operace lze přečíst chybový kód operace funkcí UP_LastErrorCode, význam vráceného chybového kódu je stejný jako u chybových kódů vracených programem UP na příkazovém řádku. Pokud není operace dokončena, UP_LastErrorCode vrátí -1.

Knihovna může ovládat maximálně 8 prográtorů současně. S přibývajícím počtem ovládaných programátorů bude více zatěžován počítač, to bude mít vliv i na rychlost programování.

Funkce UP_GetProgList vrátí seznam dostupných programátorů, zavolání této funkce zruší nastavení provedená funkcí UP_ProgConfig.

up_control64.dll je 64bitová verze up_control.dll, obě knihovny jsou obsaženy v instalačním adresáři programu UP.

1.2 Seznam funkcí

```
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_LastErrorCode(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);
```

Poznámka: Funkce UP_ProgConfig očekává, že v místě kam ukazují proměnné UP_project, NewDataFile a EEFile jsou řetězce typu ANSI string.

1.3 Popis funcí

1.3.1 UP_Prog

Funkce pošle žádost o naprogramování připojené součástky.

Nejprve je třeba nastavit potřebné parametry funkcí UP_ProgConfig.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

code - Pokud je true, programuje paměť programu.

data - Pokud je true, programuje datovou paměť.

boot - Pokud je true, programuje paměť boot.

cfg - Pokud je true, programuje konfigurační paměť.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_UP_MISSING - Nebylo možné nalézt soubor up.exe.

ERR_WRONG_PROG_INDEX - Index programátoru není v povoleném rozsahu.

ERR_NOT_CONFIGURED - Funkce UP_ProgConfig nebyla zavolána nebo nebyla úspěšně dokončena.

Příklad:

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

1.3.2 UP_DiffProg

Funkce pošle žádost o rozdílové programování připojené součástky.

Nejprve je třeba nastavit potřebné parametry funkcí UP_ProgConfig.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

code - Pokud je true, programuje paměť programu.

data - Pokud je true, programuje datovou paměť.

boot - Pokud je true, programuje paměť boot.

cfg - Pokud je true, programuje konfigurační paměť.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_UP_MISSING - Nebylo možné nalézt soubor up.exe.

ERR_WRONG_PROG_INDEX - Index programátoru není v povoleném rozsahu.

ERR_NOT_CONFIGURED - Funkce UP_ProgConfig nebyla zavolána nebo nebyla úspěšně dokončena.

Příklad:

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

1.3.3 UP_Erase

Funkce pošle žádost o smazání připojené součástky.

Nejprve je třeba nastavit potřebné parametry funkcí UP_ProgConfig.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

code - Pokud je true, maže paměť programu.

data - Pokud je true, maže datovou paměť.

boot - Pokud je true, maže paměť boot.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_UP_MISSING - Nebylo možné nalézt soubor up.exe.

ERR_WRONG_PROG_INDEX - Index programátoru není v povoleném rozsahu.

ERR_NOT_CONFIGURED - Funkce UP_ProgConfig nebyla zavolána nebo nebyla úspěšně dokončena.

Příklad:

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

1.3.4 UP_BlankCheck

Funkce pošle žádost o provedení kontroly smazání připojené součástky.

Nejprve je třeba nastavit potřebné parametry funkcí UP_ProgConfig.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

code - Pokud je true, kontroluje paměť programu.

data - Pokud je true, kontroluje datovou paměť.

boot - Pokud je true, kontroluje paměť boot.

cfg - Pokud je true, kontroluje konfigurační paměť.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_UP_MISSING - Nebylo možné nalézt soubor up.exe.

ERR_WRONG_PROG_INDEX - Index programátoru není v povoleném rozsahu.

ERR_NOT_CONFIGURED - Funkce UP_ProgConfig nebyla zavolána nebo nebyla úspěšně dokončena.

Příklad:

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

1.3.5 UP_Verify

Funkce pošle žádost o provedení kontroly naprogramování připojené součástky.

Nejprve je třeba nastavit potřebné parametry funkcí UP_ProgConfig.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

code - Pokud je true, kontroluje paměť programu.

data - Pokud je true, kontroluje datovou paměť.

boot - Pokud je true, kontroluje paměť boot.

cfg - Pokud je true, kontroluje konfigurační paměť.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_UP_MISSING - Nebylo možné nalézt soubor up.exe.

ERR_WRONG_PROG_INDEX - Index programátoru není v

povoleném rozsahu.

ERR_NOT_CONFIGURED - Funkce UP_ProgConfig nebyla zavolána nebo nebyla úspěšně dokončena.

Příklad:

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

1.3.6 UP_Read

Funkce pošle žádost o přečtení připojené součástky.

Nejprve je třeba nastavit potřebné parametry funkcí UP_ProgConfig.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

code - Pokud je true, čte paměť programu.

data - Pokud je true, čte datovou paměť.

boot - Pokud je true, čte paměť boot.

cfg - Pokud je true, čte konfigurační paměť.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_UP_MISSING - Nebylo možné nalézt soubor up.exe.

ERR_WRONG_PROG_INDEX - Index programátoru není v povoleném rozsahu.

ERR_NOT_CONFIGURED - Funkce UP_ProgConfig nebyla zavolána nebo nebyla úspěšně dokončena.

Příklad:

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

1.3.7 UP_ProgState

Funkce vrátí informaci o stavu zvoleného programátoru.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

ProgressBarValue - Vrátí hodnotu hlavního ProgressBaru programu UP.

Return values:

PROG_STATE_DONE - Poslední operace byla dokončena.

PROG_STATE_BUSY - Programátor je zaneprázdněn.

PROG_STATE_NOT_USED - Programátor ještě nebyl použit.

PROG_STATE_WRONG_INDEX - Index programátoru není v povoleném rozsahu.

Příklad:

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

1.3.8 UP_LastErrorCode

Funkce vrátí chybový kód poslední dokončené operace.

Vrácené hodnoty odpovídají hodnotám vraceným programem UP na příkazovém řádku.

Definice funkce:

int __stdcall UP_LastErrorCode(int prog_index);

Parametry:

prog_index - Index programátoru.

Návratové hodnoty: Vrácené hodnoty odpovídají hodnotám vraceným programem UP na příkazovém řádku. Pro více informací viz manuál programátoru **Návratové kódy programu**. Pokud operace není dokončena, funkce vrátí -1.

Příklad:

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

1.3.9 UP_ProgConfig

Funkce nastaví parametry pro následující operace. Toto je první funkce, která by měla být zavolána.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

UP_project - Zvolí projektový soubor ppr programu UP.

prog_type - Vybere programátor podle konstant.

prog_SN - Specifikuje sériové číslo programátoru, pokud je 0, použije se hodnota definovaná v souboru projektu.

NewDataFile - Specifikuje datový soubor, který nahradí datový soubor definovaný v projektu, stejně jako parametr /df programu UP.

EEFile - Specifikuje datový soubor pro datovou paměť, který nahradí soubor definovaný v projektu, např. pro AVR. Je to stejná funkce, kterou dělá parametr /e programu UP.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_UP_MISSING - Nebylo možné nalézt soubor up.exe.

ERR_WRONG_PROG_INDEX - Index programátoru není v povoleném rozsahu.

ERR_NOT_CONFIGURED - Funkce UP_ProgConfig nebyla zavolána nebo nebyla úspěšně dokončena.

ERR_FILE_DOES_NOT_EXIST - Zvolený soubor projektu neexistuje.

Příklad:

```
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);
```

1.3.10 UP_SetManualSN

Funkce nastaví manuální SN, které je vyžadováno, pokud je jeho použití definováno v souboru ppr.

Definice funkce:

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

Parametry:

prog_index - Index programátoru.

DefineSN - Pokud je true, manuální sériové číslo bude posláno programu UP. Výchozí hodnota je false.

SN - Definuje vlastní sériové číslo.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_WRONG_PROG_INDEX - Index programátoru není v

povoleném rozsahu.

Příklad:

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

1.3.11 UP_GetProgList

Funkce vrátí seznam dostupných programátorů, seznam je vrácen jen když knihovna zrovna žádný programátor nepoužívá. Zavoláním této funkce se zruší nastavení, která byla dříve provedena funkcí UP_ProgConfig.

Definice funkce:

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

Parametry:

prog_type - Výběr typu programátoru podle konstant.

sn_list - Pole hodnot typu integer. Vrací seznam SN dostupných programátorů FORTE. Vrací SN v délce 24bitů, jako je zobrazováno v programu UP.

count - Požadovaný počet hodnot.

count_returned - V této proměnné je vrácen počet skutečně vrácených hodnot, počet dostupných programátorů.

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně zavolána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

ERR_DRIVER - Chyba ovladače programátoru.

V **sn_list** funkce vrací seznam dostupných programátorů a počet vrácených hodnot v **count_returned**.

Příklad:

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

1.3.12 UP_CleanUp

Funkce ukončí práci knihovny. Tuto funkci je potřeba zavolat před voláním funkce FreeLibrary.

Definice funkce:

int __stdcall UP_CleanUp(void);

Návratové hodnoty:

ERR_NONE - Funkce byla úspěšně vykonána.

ERR_PROG_BUSY - Programátor je zaneprázdněn.

1.4 Konstanty

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

PROG_PRESTO=1;
PROG_FORTE=2;
```

1.5 Návratové hodnoty funkcí

```
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;
```

1.6 Návratové hodnoty funkce UP_ProgState

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

2

Historie dokumentu

Revize dokumentu	Provedené úpravy
19.9.2019	Dokument vytvořen.
8.10.2020	Doplněna funkce UP_GetProgList a konstanty.
21.5.2021	Změněn údaj o množství podporovaných programátorů.
20.8.2021	Doplněna funkce UP_CleanUp.
	Doplněn popis parametru prog_SN funkce UP_ProgConfig.