ICP Family Programmers

DLL Description

1 Installation
Run DLL installation file “IcpDll_setup_dll_XXX.exe”.

2 Files
- IcpDll.dll
- IcpWinComLine.exe
- c_icpexp.h
- fr_exp.h
- DLL Description.pdf
- ICP Command Line.pdf
- \Lib_Borland\IcpDll.lib
- \Lib_Microsoft\IcpDll.lib

3 ICP Firmware Options
DLL/Command Line Support (D) should be activated in order to use DLL functions

4 General Sequence of Operations

Step	*Function*	*Description*	*Usage*
1.	IcpStartApplication()	Starts ICP application	Should be called once, mandatory. Next call can be done after application is closed by IcpEndApplication()
2.	IcpInitCom()	Initializes COM port	Once, mandatory. Next call can be done after COM is released by IcpReleaseCom()
3.	IcpLoadHexAndSerFile()	Loads hex and/or serialization files	Once or repeated, not mandatory
4.	IcpDoAction()	Executes action according to ACTION_LIST	Once or repeated, not mandatory
5.	IcpReadStaResOneCh()	Reads result of standalone operation for selected channel	Once or repeated, not mandatory. Not required in PC-driven mode
6.	IcpReleaseCom()	Releases COM port	Once or repeated, not mandatory
7.	IcpEndApplication()	Closes the application	Once, mandatory

5 Return Values

DLL functions return value according to enum AUTO_ERROR_LEVEL below:

```
enum AUTO_ERROR_LEVEL { //return values

AUTO_OK                          = 0,  //operation OK
AUTO_DB_ERR                      = 1,  //database error
AUTO_COM_ERR                     = 2,  //communication error
AUTO_VDD_ERR                     = 3,  //Vdd overload error
AUTO_VPP_ERR                     = 4,  //Vpp overload error
AUTO_HEX_ERR                     = 5,  //HEX file loading error
AUTO_SER_ERR                     = 6,  //serialization file error
AUTO_VER_ERR                     = 7,  //verification error
AUTO_ERR_NO_SPACE                = 8,  //no space selected
AUTO_SAVE_ERR                    = 9,  //file save error
AUTO_SOCK_ERR                    = 10, //socket communication error (obsolete)
AUTO_I2C_ERR                     = 11, //UUT I2C communication error
AUTO_DLL_ERR                     = 12, //DLL programming is not supported
AUTO_KEY_ERR                     = 13, //key generation error
AUTO_CFG_ERR                     = 14, //config. file error
AUTO_COM_NUM_ERR                 = 15, //invalid COM number
AUTO_COM_BUSY_ERR                = 16, //selected COM is busy
AUTO_COM_BAUD_ERR                = 17, //invalid baud rate
AUTO_COM_NO_OPEN                 = 18, //can't open COM port
AUTO_USER_CANCEL                 = 19, //user cancel
AUTO_IN_PROGRESS                 = 20, //operation in progress
AUTO_BC_ERR                      = 21, //blank check error
AUTO_OP_NOT_ALLOW                = 22, //operation not allowed for selected programmer
AUTO_FW_INVALID                  = 23, //firmware invalid-firmware upgrade needed
AUTO_24LC_ADDR_ERR               = 24, //24LC01 address (offset) is out of range
AUTO_DM_ADDR_ERR                 = 25, //DM range error
AUTO_FIRM_ERR                    = 26, //firmware version error
AUTO_NO_SUB                      = 27, //no ICP-SUB PCB
AUTO_NO_SUP_KEE                  = 28, //no keeloq support
AUTO_NO_SUP_DSPIC                = 29, //no dsPIC support
AUTO_ICP2_REQ                    = 30, //ICP2 required
AUTO_DEV_ERR                     = 31, //device selection error (unspecified error)
AUTO_PROG_MISMATCH               = 32, //mismatch between selected and
                                       //detected programmers
AUTO_PRJ_INVALID                 = 33, //Invalid environment
AUTO_PRJ_DB_FIRM_PC_MIS          = 34, //mismatch between PC and firmware database
AUTO_PRJ_DB_FIRM_AT45_MIS        = 35, //mismatch between environment and
                                        //firmware database
AUTO_DLL_SUPPORT_REQIURED        = 36, //"GO" pressed on hardware and no
                                        //DLL/standalone support
AUTO_PRJ_CS                      = 37, //environment CS error
AUTO_STA_IDLE                    = 38, //programmer is idle or standalone
                                        //operation can't be started
AUTO_STA_BUSY                    = 39, //standalone operation: programmer busy
AUTO_ENV_ERR                     = 40, //environment file error
AUTO_PM_RANGE                    = 41, //invalid PM range specified
AUTO_DEMO_ERR                    = 101 } //demo version
```

6 Base Functions

6.1 IcpStartApplication

Description: Starts application and loads configuration file

Prototype: int DLL_FUNC IcpStartApplication (char *aFileCfg)

Parameters: aFileCfg - ICP configuration file to be loaded, usually "icp01.cfg"

Example: int Stat = IcpStartApplication ("c:\MyProject\Icp01.cfg");

6.2 IcpInitCom

Description: Initializes RS-232/USB/Bluetooth COM port

Prototype: int __stdcall IcpInitCom (int aOverCfg, int aComPort, int aBaudRate)

Parameters:

aOverCfg:	0-gets communication parameters from *.cfg file 1-overrides *.cfg file with aPort and aBaudRate
aPort:	0-COM1, 1-COM2,...
aBaudRate:	see enum BAUD_RATES

Example: Stat = IcpInitCom(0, 0, 0); //use settings from *.cfg file

6.3 IcpLoadHexAndSerFile

Description: Loads HEX and serialization files

Prototype: int __stdcall IcpLoadHexAndSerFile (char *aFileHex, char *aFileSer)

Parameters: aFileHex - pointer to char string containing HEX file name

aFileSer - pointer to char string that contains serialization file name

Example 1: Stat = IcpLoadHexAndSerFile ("c:\MyProject\1.hex", "c:\MyProject\1.ser");

Example 2: Stat = IcpLoadHexAndSerFile ("c:\MyProject\1.hex", ""); //hex file only

Example 3:
```
Stat = IcpLoadHexAndSerFile ("", "c:\MyProject\1.ser"); //serialization file
        //only (standalone operation)
```

6.4 IcpDoAction

Description: Executes programming, verification and other actions specified in enum ACTION_LIST below:

```
enum ACTION_LIST {
        ACT_PROG                = 1,    //programming (PC-driven mode)
        ACT_VER                 = 2,    //verification (PC-driven mode)
        ACT_READ                = 3,    //read (PC-driven mode)
        ACT_BC                  = 4,    //blank check (PC-driven mode)

        ACT_STA_PROG            = 5,    //programming (standalone mode)
        ACT_STA_GET_RES         = 6,    //get all results of the last
                                        //operation (standalone mode)
        ACT_STA_CLR_RES         = 7,    //clear all results of the last
                                        //operation (standalone mode)
        ACT_STA_START_PROG      = 8,    //start standalone programming
                                        //w/o monitoring
};
```

Prototype:

```
int DLL_FUNC  IcpDoAction(    int aAction,
                              unsigned int aMemorySpace,
                              unsigned int aPmUserRange,
                              unsigned int aPmAddrBeg,
                              unsigned int aPmAddrEnd,
                              unsigned int aSaveResult,
                              char* aReadFile  );
```

Parameters: **aAction**: one of values of ACTION_LIST. Note: ICP software automatically removes memory spaces which do not exist in the selected device

aMemorySpace: sum of values of MEMORY_SPACES below:

```
enum MEMORY_SPACES {
        PM_SPACE          = 0x0001, //program memory (PM)
        ID_SPACE          = 0x0002, //ID locations
        DM_SPACE          = 0x0004, //data memory (EEPROM)
        CM_SPACE          = 0x0008, //calibration memory (not supported)
        FU_SPACE          = 0x0010, //configuration bits
        ALL_SPACE = PM_SPACE | ID_SPACE | DM_SPACE | FU_SPACE
};
```

aPmUserRange:	0-use PM range from database (full range), 1-override with aPmAddrBeg and aPmAddrEnd
aPmAddrBeg:	start address of PM if aPmUserRange==1
aPmAddrEnd:	end address of PM if aPmUserRange==1
aSaveResult:	1-operation result will be written to file "auto01.res"
aReadFile:	hex file to be saved after read (aAction==ACT_READ)

Example 1 (PC-driven mode): Program (and verify) entire chip

```
int Stat = IcpDoAction( ACT_PROG,           //aAction
                        ALL_SPACE,          //aMemorySpace
                        0,                  //aPmUserRange
                        0,                  //aPmAddrBeg
                        0,                  //aPmAddrEnd
                        0,                  //aSaveResult
                        “”);                //aReadFile
```

Example 2 (PC-driven mode): Verify locations 0x0020-0x003F of PM

```
Stat = IcpDoAction(     ACT_VER,            //aAction
                        PM_SPACE,           //aMemorySpace
                        1,                  //aPmUserRange
                        0x0020,             //aPmAddrBeg
                        0x003F,             //aPmAddrEnd
                        0,                  //aSaveResult
                        “”);                //aReadFile
```

Example 3 (PC-driven mode): Read entire chip to file "c:\prj\hex1.hex"

```
Stat = IcpDoAction(     ACT_READ,           //aAction
                        ALL_SPACE,          //aMemorySpace
                        0,                  //aPmUserRange
                        0,                  //aPmAddrBeg
                        0,                  //aPmAddrEnd
                        0,                  //aSaveResult
                        "c:\prj\hex1.hex"); //aReadFile
```

Example 4 (Standalone mode): Program (and verify) all chip(s)

```
int Stat = IcpDoAction( ACT_STA_PROG,       //aAction
                        ALL_SPACE,          //aMemorySpace (Note 1)
                        0,                  //aPmUserRange
                        0,                  //aPmAddrBeg
                        0,                  //aPmAddrEnd
                        0,                  //aSaveResult
                        “”);                //aReadFile
```

Note 1: aMemorySpace parameter does not affect standalone operation since memory space is selected according to environment settings saved in ICP2 internal flash memory

6.5 IcpReleaseCom

Description: Releases serial communication port

Prototype: int __stdcall IcpReleaseCom (void)

Parameters: None

Example: Stat = IcpReleaseCom();

6.6 *IcpEndApplication*

Description: Terminates ICP DLL application

Prototype: int __stdcall IcpEndApplication (void)

Parameters: None

Example: Stat = IcpEndApplication();

6.7 *IcpReadDllVersion*

Description: Gets DLL version string

Prototype: int __stdcall IcpReadDllVersion (char *dllSoftwareVer)

Parameters: dllSoftwareVer – pointer to DLL version string

Example:
```
static char DllVersion[80];
Stat = IcpReadDllVersion (DllVersion);
```

6.8 *IcpReadStaResOneCh*

Description: Reads result of standalone operation for one channel. Should be called in loop for all channels after previous programming is done

Prototype: int __stdcall IcpReadStaResOneCh(unsigned int aCh);

Parameters: aCh: channel number, range 0...63. NOTE: aCh is 0 for ICP2 (not GANG)

Return Value: A) -1 if channel is not enabled
B) -2 if channel number is out of range (>63)
C) according to AUTO_ERROR_LEVEL

Example: Program 8 channels of ICP2-GANG and get results

Step 1: Execute standalone programming for 8 channels

```
#define CH_NUM  8  //8 channels
int Res[CH_NUM];
int Stat = IcpDoAction(  ACT_STA_PROG,        //aAction
                         ALL_SPACE,           //aMemorySpace
                         0,                   //aPmUserRange
                         0,                   //aPmAddrBeg
                         0,                   //aPmAddrEnd
                         0,                   //aSaveResult
                         "");                 //aReadFile
```

Step 2: Analyze result

```
if (Stat==AUTO_OK)
{
  ; //do nothing, all channels OK
}
else
{
  for (int i=0; i<CH_NUM; i++)
    Res[i]= IcpReadStaResOneCh[i]; //save all results
{
```

6.9 IcpEnableProgressWindow

Description: Enables/disables progress window (progress bar)

Prototype: int __stdcall IcpEnableProgressWindow(int aEnable);

Parameters: aEnable: 0-disable, 1-enable

7 Advanced Functions

See detailed description of advanced functions in file ***"c_icpexp.h"***