IS01 OEM DLL Instruction ®XELTEK

December 28, 2010

Index

1. Introduction

SuperPro ISP (in-system programmer) series provide software interface to facilitate third party system integration using specific Dynamic-link library (DLL) files. DLL files could be called through system design platforms such as Visual C++, PowerBuilder, Visual Basic, Labview…etc. to control SuperPro functions without using the SuperPro software.

2. Requirements

Software installed correctly.

3. OEM DLL API

DLL offers six functions:

ISP_Oem_Init,
ISP_Oem_Free,
ISP_Oem_Call,
ISP_Oem_Readbuf
UpdateFile
UpdateFileB

int ISP_Oem_Init(char * workpath);

Function:
Initialization of system
Arguments:
Workpath: The path where project files are located

int ISP_Oem_Free(int devno);

Function:
Close the selected ISP programmer
Arguments:
Devno: The number of ISP

int ISP_Oem_Call(int devno,char *function,char *par,char *retStr,int *retVal);

Arguments:
Devno: ISP programmer ID (if 1 ISP then Devno = 1, if N ISPs then Devno of the Nth ISP is N)
Function: **“Ping”**, **“LoadPrj”**, **“Run”**, **“GetStatus”**
retStr: return string value
retVal: return int value

functionName detail:
“Ping”: Communicate with ISP
Arguments: none
Return: if works OK, retStr = **“ping_ok”**,else retStr= **“ping_error”**
“LoadPrj”: load project file

Par: Project name, such as **“24.prj”**
Return: if works OK, retStr= **“LoadPrj_OK”**, else retStr= **“ping_error”**
“Run”: Start algorithm
Par: function name,such as “Auto”,”Program”
“GetStatus”: Get status of ISP
Par: none
Return: retStr has several cases:
“**scale**”: project progression, retVal is value , Example: 5 is 5%
“**func_ok**” func works, Example:"**Program_ok**” is program works / executed.
“**func_fail**” is function work fail, Example:"**Program_fail**” is program fail / did not execute.
“**UnloadPrj**”: unload project file
Par: none
Return: retStr = “**unloadprj_ok**” or retStr = “**unloadprj_fail**”
“**DisDebug**”: disable debug log
Par: none
Reutrn: retStr = “**disdebug_ok**”
“**EnDebug**”: enable debug log
Par: none
Return: retStr = “**endebug_ok**”
If enable debug log, files **ispoem.txe** ,**ispoem_t0.txt** … will be in the C directory.

int ISP_Oem_Readbuf(char *prjname, Int nbuf, int startaddr, int size, char *buf)
Function:
Get the data from project file
Arguments:
prjname: project file name
bufno: the index of buf, begin from 0
startaddr: the start address
size; the size of data
buf: the buffer where data will be saved
Return:
-1:file open error
Other: OK

unsigned int UpdateFile(wchar_t *src, wchar_t *dest, int nbuf, int type, int mode, __int64bufaddr, __int64 fileaddr, bool calcs, bool erase, DeviceInfo* pDevInfo)

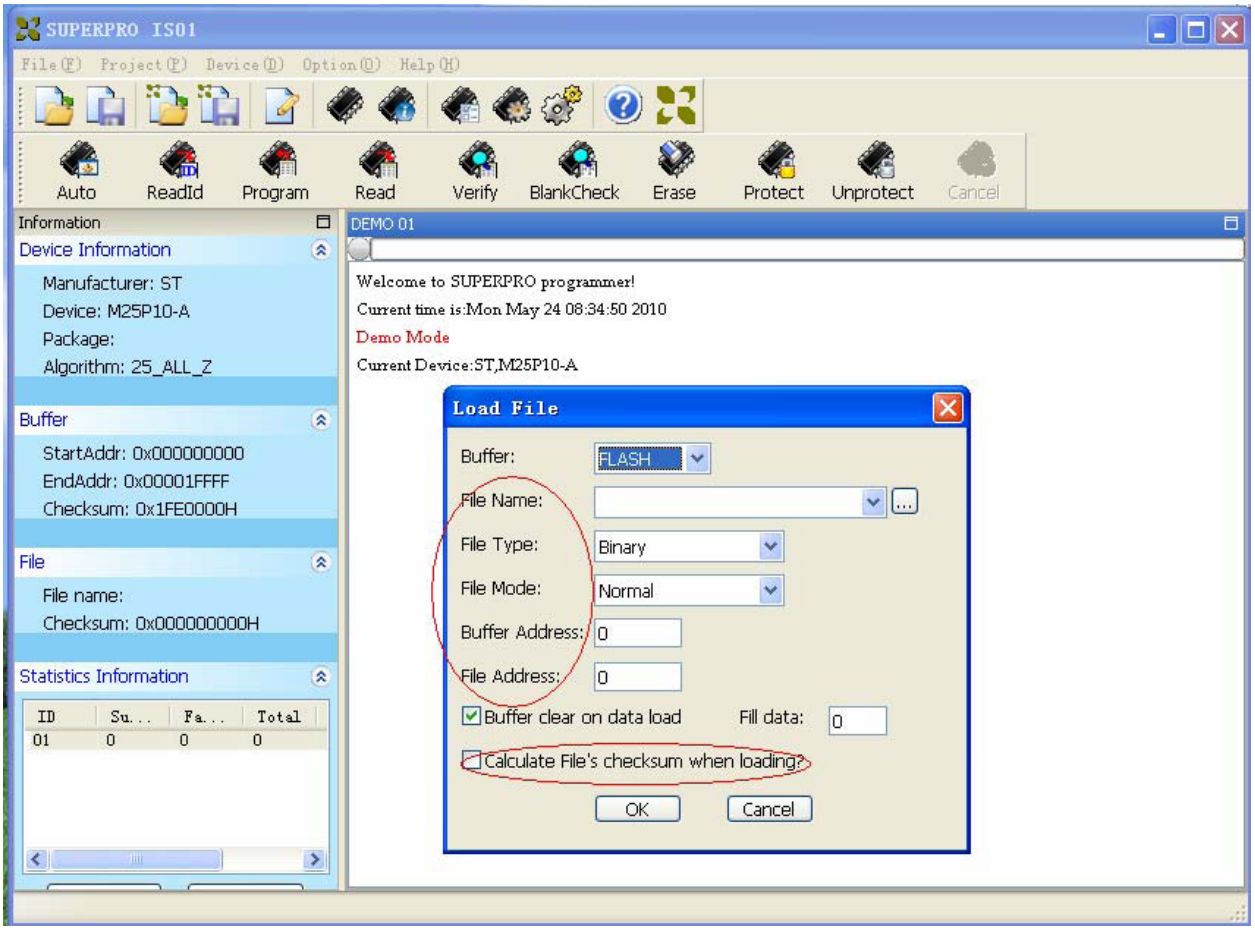
```
#pragma pack (1)
struct DeviceInfo
{
char DEVNAME[60];
char MANU[32];
unsigned int fileOffset;
int pin;
int line;
unsigned int checksum;
unsigned char bufType;
unsigned char nbuf;
};
#pragma pack ()
```

Function:
Replace date file of project
Return:
0:OK 1:File Opens Error 2:Checksum error 3: File error
4:File Type Error 6:Device not match
Arguments:
src: The file to be loaded (Use the absolute path such as **C:\OEM\filename** and not just the filename)
dest: prj file (Please use absolute path)
nbuf: the index of buffer, begin with 0

type: 0-binary; 1-hex; 2-motorola
mode: 0- Normal;
1- 1st of 2;
2- 2nd of 2;
3- 3rd of 4;
4- 4th of 4;
5- 1st 2 bytes of 4
6- 2nd 2 bytes of 4
bufaddr: buffer offset when loading
fileaddr: file offset when loading
calcs: whether to calculate file's checksum
erase: whether to erase the old data file
pDevInfo: Contains information for loading and returning a file with the struct defined below.

```
#pragma pack (1)
struct DeviceInfo
{
char DEVNAME[60];
char MANU[32];
unsigned int fileOffset;
int pin;
int line;
unsigned int checksum;
unsigned char bufType;
unsigned char nbuf;
};
#pragma pack ()
```

Note: The arguments type, mode, bufaddr, fileaddr, calcs have the same meaning when loading file with SuperProIS01 software.



Unsigned int UpdateFileB(wchar_t *src, wchar_t *dest, int nbuf, int type, int mode, __int64 bufaddr,__int64 fileaddr,bool erase)

Function:

Replace date file of project. Similar to UpdateFile, but does not have arguments for **calcs** and **pDevInfo**. UpdateFile calculates the data file check sum and this function does not.

4. Basic procedures for calling DLL

Generate project file with SuperProIS01 software (refer to the SuperProIS01 handbook for more details), then call OEM DLL to operate ISP.

Example: Operate chip 24C02, and store project file in **C:\Oem**.

Steps:

1) Create dir c:\Oem
2) Select 24C02 in SuperProIS01 software, Load data file and Edit Auto, Generate project file: **24.prj**, store in **C:\Oem**
3) Call ISP_Oem_Init("C:\\Oem\\") to initialize system.
4) Call ISP_Oem_Call(1,"Ping","1",retStr,&retVal) for communication set up.
   If **retStr=="ping_ok",** the communication with programmer is ok, else if **returnStrinf=="ping_error",** the communication is not ok.
5) Call **ISP_Oem_Call(1,"LoadPrj","24.prj",retStr,&retVal)** to load project file
6) Call **ISP_Oem_Call(1,"Run","Auto",retStr,&retVal)** to start "Auto" operation
7) Call **ISP_Oem_Call(1,"GetStatus","",retStr,&retVal)** to check status of ISP in loop.
   There are two check modes:
   A: If run "**Auto**", when **retStr="Auto_ok"**, then work passes, and if retVal=0x81, then work fails. retStr stores the reason for failed data.
   B: When program is run one step at a time, such as "**Program**", the result could be checked in two ways:

1:retStri=”Program_ok” or retStri=”Program_fail”
2:if retVal=0x80, it means work ok, and if retVal=0x81,means work fail.

8) Call ISP_Oem_Free(1) to close ISP system.

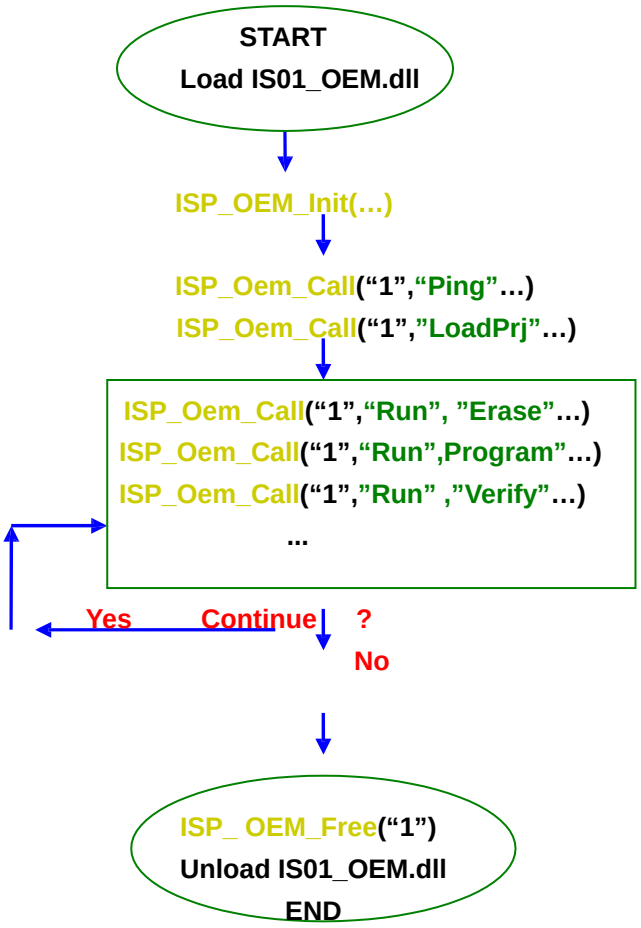
Note: When devno is 1, the first ISP is closed.

Example Code:

```
#include <windows.h>
#include <time.h>
#include <stdio.h>
typedef int (*T_OEM_Init)(char *workpath);
typedef int (*T_OEM_Free)(int devno);
typedef int (*T_OEM_Call)(int devno, char *function, char *parameter,
char *retStr,int *retVal);
typedef int (*T_OEM_Read)(char *prjname, int bufno, int startaddr, int size, char *buff);
T_OEM_Init OEM_Init=NULL;
T_OEM_Free OEM_Free=NULL;
T_OEM_Call OEM_Call=NULL;
T_OEM_Read OEM_Read=NULL;
int Auto (void)
{
char retStr[80];
int retInt;
int ret;
time_t st, now;
ret=OEM_Call(1,"Run", "Auto", retStr, &retInt);
if(ret){
printf("Error:%s\n",retStr);
}
time(&st);
while(1) {
OEM_Call(1, "GetStatus", "", retStr, &retInt);
if(_stricmp(retStr,"Auto_ok")== 0){
printf("Auto ok\n");
break;
}
if(retInt==0x81){ //NOTE!
printf("Fail:%s\n",retStr);
break;
}
time(&now);
if((now - st) >10) {
printf("Timeout error while auto\n");
break;
}
}
return retInt;
}
int Program (void)
{
char retStr[80];
int retInt;
int ret;
time_t st, now;
ret=OEM_Call(1,"Run", "Program", retStr, &retInt);
if(ret){
printf("Error:%s\n",retStr);
}
time(&st);
```

```
while(1) {
OEM_Call(1, "GetStatus", "", retStr, &retInt);
if(_stricmp(retStr,"Program_ok")== 0 ||retInt==0x80){
printf("Program ok\n");
break;
}
if(_stricmp(retStr,"Program_fail")== 0 ||retInt==0x81){
printf("Program fail\n");
break;
}
time(&now);
if((now - st) > 5) {
printf("Timeout error while programming serial flash\n");
break;
}
}
return retInt;
}
int main(void)
{
char retStr[80];
int retInt;
int ret;
HINSTANCE hInstLib=NULL;
hInstLib = LoadLibrary("C:\\OEM\\IS01_OEM.dll");
OEM_Init = (T_OEM_Init)GetProcAddress(hInstLib, "ISP_Oem_Init");
OEM_Free = (T_OEM_Free)GetProcAddress(hInstLib, "ISP_Oem_Free");
OEM_Call = (T_OEM_Call)GetProcAddress(hInstLib, "ISP_Oem_Call");
OEM_Read = (T_OEM_Read)GetProcAddress(hInstLib, "ISP_Oem_Readbuf");
if(OEM_Init == NULL) {
printf("Error loading programmer DLL\n");
return 0;
}
OEM_Init("C:\\Oem\\");
retInt = OEM_Call(1,"Ping", "",retStr, &retInt);
if(retInt)
return retInt;
OEM_Call(1, "LoadPrj", "24.prj", retStr, &retInt);
return 1;
}
```

One Device Work Flow Figure



5. Load data file into a project file

Example:

Project file **24c02.prj** , Data file **24.bin**
To load **24.bin** into **24c02.prj** without erasing the old file, the argument must be false, otherwise true.

```
Int Ret;
DeviceInfo devinfo;
Ret=UpdateFile(L"c:\\OEM\\24.bin",L"c:\\OEM\\24c02.prj", 0,0,0,0,0,False,False ,&devinfo)
If(Ret)
{
…//Some error occurs!
}
```

Note: Argument src and dest are **wchar_t** , not char.

UpdateFileB:

Int Ret;
Ret= UpdateFileB(L“c:\\OEM\\24.bin”,L “c:\\OEM\\24c128.prj”, 0, 0,0,0,0,false)
START
Load IS01_OEM.dll
ISP_OEM_Init(…)
ISP_Oem_Call(“1”,“Ping”…)
ISP_Oem_Call(“1”,”LoadPrj”…)
ISP_Oem_Call(“1”,“Run”, ”Erase”…)
ISP_Oem_Call(“1”,“Run”,Program”…)
ISP_Oem_Call(“1”,”Run” ,”Verify”…)
...
Yes Continue ?
No
ISP_ OEM_Free(“1”)

Unload IS01_OEM.dll
END

If(Ret)
{
…//Some error occurs!
}

6. Multi-devices Control

int ISP_Oem_Call(int **devno**,char *function,char *par,char *retStr,int *retVal);
int ISP_Oem_Free(int **devno**);

Note: Argument “**devno**” represents the SuperProIS01 number ID.
If have two SuperProIS01s are to be controlled, the devno may be 1 and 2. Every device has its own project file, and the devno represents the number of devices and project files. After system init, every SuperProIS01 will display its own ID number (such as 1 ,2 …) on the LCD.

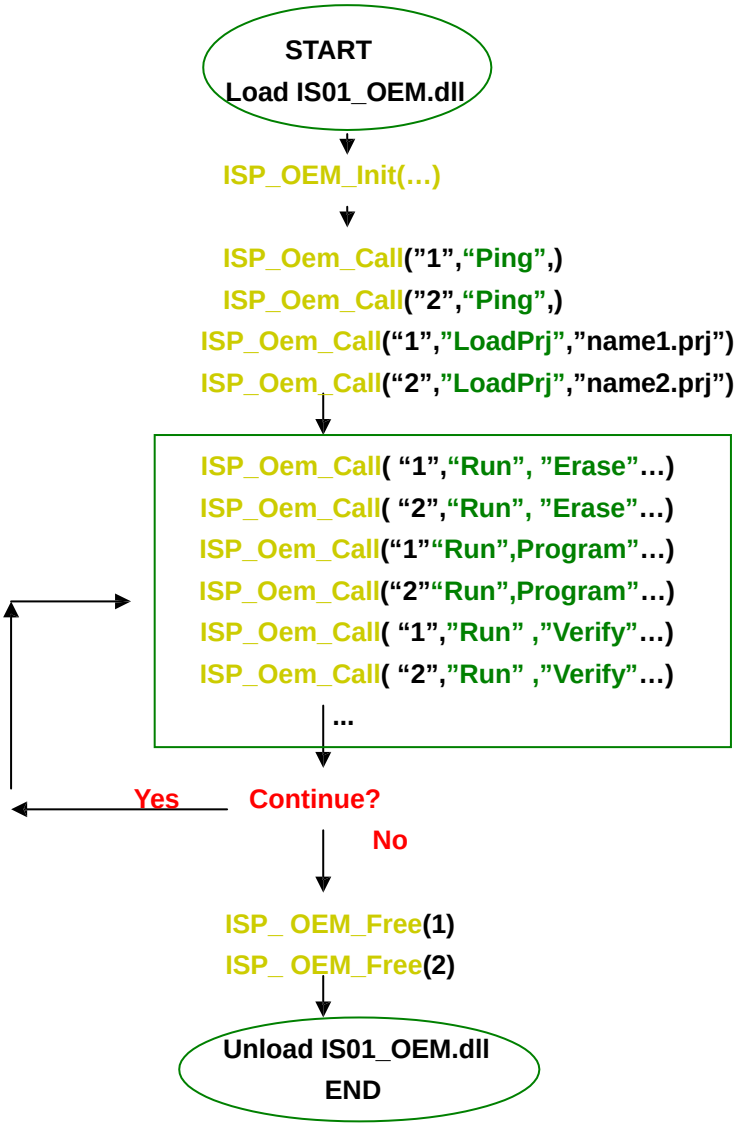
Example:

Running two devices on “Auto”.

```
int Autos (void)
{
char retStr[80];
int retInt;
int ret;
ret=OEM_Call(1,"Run", "Auto", retStr, &retInt);
if(ret){
printf("Error:%s\n",retStr);
}
ret=OEM_Call(2,"Run", "Auto", retStr, &retInt);
if(ret){
printf("Error:%s\n",retStr);
}
while(1) {
OEM_Call(1, "GetStatus", "", retStr, &retInt);
if(_stricmp(retStr,"Auto_ok")== 0){
printf("First Auto ok\n");
break;
```

```
}
if(retInt==0x81){ //NOTE !
printf("First Fail:%s\n",retStr);
break;
}
}
while(1) {
OEM_Call(2, "GetStatus", "", retStr, &retInt);
if(_stricmp(retStr,"Auto_ok")== 0){
printf("Second Auto ok\n");
break;
}
if(retInt==0x81){ //NOTE !
printf("Second Fail:%s\n",retStr);
break;
}
}
return retInt;
}
```

Multi-devices Work Flow Figure



7. Multi-chips Operation with One SuperProIS01

Two chips (same or different) could be operated at once with the project file a.proj and b.proj.

8. Other

Using DLL with Visual Basic
IS01_OEMB.dll is supplied to support Visual Basic

Function declaration:

Private Declare Function ISP_Oem_Init Lib "path\IS01_OEMB.dll" (ByVal path As String) As Long
Private Declare Function ISP_Oem_Free Lib "path\IS01_OEMB.dll" (ByVal devno As Long) As Long
Private Declare Function ISP_Oem_Call Lib "path\IS01_OEMB.dll" (ByVal devno As Long, ByVal funcname As
String, ByVal pars As String, ByVal retstr As String, ByRef retval As Long) As Long

Example for calling each function:

```
Dim ret As Long
ret = ISP_Oem_Init("C:\OEM")
Dim deno As Long
Dim ret As Long
devno = 1
ret = ISP_Oem_Free(devno)
Dim retstr As String
Dim retval As Long
retstr = Space(80)
ret = ISP_Oem_Call(1, "Loadprj", "24.prj", retstr, VarPtr(retval))
```

9. Works on Linux

There are three steps to operate SuperProIS01 on Linux.

1. Prepare a USB driver for SuperProIS01 and install the SuperProIS01 library (lib).
2. Create a project file with selected chip and imaging data.
3. Control SuperProIS01 to Program , Verify…etc.

Files:
husb.c: The source of the SuperProIS01 USB driver.
Is01lib.tar.gz: SuperProIS01 library (lib).
libis01.so: Dynamic lib.
demo.c: Demo code.
is01 oem dll(s).pdf : Handbook

Preparation:

A: Compile the USB driver for SuperProIS01
Compile **husb.c** in the Linux platform, insmod **husb.ko**create, create the devices below:

```
mknod /dev/spis01_0 c 52 0
mknod /dev/spis01_1 c 52 1
mknod /dev/spis01_2 c 52 2
mknod /dev/spis01_3 c 52 3
```

B: Install SuperProIS01 library (lib)
Tar **xvf Is01lib.tar.gz** in a location such as: /home/work.
There will be two folders: **algo** and **lib**

Create a Project File:

int CreateProject(struct PRJPARAMETER* pPrjParameter,char* prjFile);

Function: create a project file for selected chip.

Note: The project file created by **CreateProject** has no imaging data file.
Call UpdateFile( ) to fill the project file with image data to be programmed.

Parameter:

1. struct PRJPARAMETER* pPrjParameter

struct PRJPARAMETER{
char manuName[40]; //manufacturer's name
char devName[60]; //Device name
unsigned char configWord[64]; //Device configure Word. Users should turn to Algorithm writer for the correct value.
struct OPOPTION option; // see below
char autos[20]; //The sequence is the same as that in function list of SPIS01's software.("Auto" is 0, then 1,2,3...)
};
struct OPOPTION{
char idCheck; // 0-uncheck, 1-check
float vcc; // 0-default value
float vio; // 0-default value
char isProductionMode; //0-production mode, 1 - not production mode
int delay; // delay value
char isPullUp; // 0-not pullup, 1-pullup
int clock; // 0 - default value
char verifyMode; // 0-verify once with VCC, 1-verify twice VCC with +-5%, 2-verify twice with VCC +-10%
char speedMode; // 0-high speed, 1-middle speed, 2-low speed
long long startAddr[5]; // 0-default value
long long endAddr[5]; // 0-default value
char autoInc; // the value should be 0
int countsetting; // the value should be 0
};

2. char* prjFile: path of project file

Return value:

0: success
1: load device error
2: algorithm file not exist
3: fpga file not exist

Operate IS01 with API

Functions offered in **libis01.so**:
ISP_Oem_Init,
ISP_Oem_Free,
ISP_Oem_Call,
UpdateFile
UpdateFileB
section 3: OEM DLL API gives description in detail.

Note: ISP_Oem_Readbuf not offered in Linux

Example:

The ATEMEL chip AT24C02 with image file "**a.bin**" is to be programmed. Assuming that The name of project that will be create is "**AT24.prj**" and work location is " **/home/nfs** ".

Step1: Compile **husb.c** , and install driver.
Make device nods: **mknod /dev/spis01_0 c 52 0**
mknod /dev/spis01_1 c 52 1
mknod /dev/spis01_2 c 52 2
mknod /dev/spis01_3 c 52 3
install lib of IS01: tar xvf Is01lib.tar.gz in the location where the application to run.

Step2: Create a project.
Call CreateProject( ) to create a project file AT24.prj.
Call Updata to fill AT24.prj with image data file a.bin.

Step3: Operate IS01
ISP_Oem_Init() to init location the project file will be put and check IS01 device on the platform.
ISP_Oem_Call(devno,"ping",…) to connect IS01
ISP_Oem_Call(devno,"loadprj","AT24.prj",retstr,&retval) to load project file to IS01.
ISP_Oem_Call(devno,"run","program",retstr,&retval) to program the chip.

Note: Linux only support one SuperProIS01 at same time, so the parameter "**devno**" must always be 1.
The **demo.c** contains a sample code on how to program SuperProIS01 on Linux.
Makefile:
demo: demo.c
gcc -Wall -o demo demo.c ./libis01.so

Note:

1: The application demo , libis01.so , algo , lib must be stored in the same location (aglo and lib are decompressing files from Is01lib.tar.gz)
2: Contact Xeltek for the SuperProIS01 cable/pin connections.
3: To Run Auto, contact Xeltek for help with edit auto help.
4: Some advanced functions such as loading POF file cannot be accomplished using DLL. Customer should covert the POF file to bin file for using it with DLL.