

ETHERNET **POWERLINK**

POWERLINK 的 windows 实现

王谨秋

POWERLINK 开发板，POWERLINK 协议软件和技术支持，IO 模块，运动控制器，工业机器人控制器。

QQ 群： [151181908](#)

联系人： 王先生

手机： 13917489045（微信同号）

邮箱： openpowerlink@163.com

网址： <http://www.openat.cn>

POWERLINK 的 windows 实现

1 编译

编译过程参照 powerlink 说明中的“Building”下的“Building openPOWERLINK”的说明。

Main Page	Ethernet POWERLINK	openCONFIGURATOR	openPOWERLINK Stack	Software Manual	Contribute	Revision History	Glossary
▼ openPOWERLINK Ethernet POWERLINK openCONFIGURATOR ▼ openPOWERLINK Stack Directory Structure Components ▼ Building Building openPOWERLINK Building Stack Libraries Building Drivers Building Demo Applications Building Hardware Platforms ▶ Supported Platforms ▶ Software Manual ▶ Contribute Revision History Glossary Licenses <pre>> cd build > cmake ..</pre> Cross Compilation CMake allows cross compiling for another platform. Therefore "cross platform toolchain files" will be delivered configure the build to use a cross compiler and build environment you have to select the appropriate cross too <pre>> cmake -DCMAKE_TOOLCHAIN_FILE=<PATH_TO_TOOLCHAIN_FILE></pre> Build Steps The following build steps are necessary to build an openPOWERLINK solution: • Build the hardware platform (This step is only necessary for embedded targets) • Build the openPOWERLINK stack libraries • Build the necessary openPOWERLINK drivers • Build your application (or a delivered demo application)							

其中有一项说明“Build Steps”。

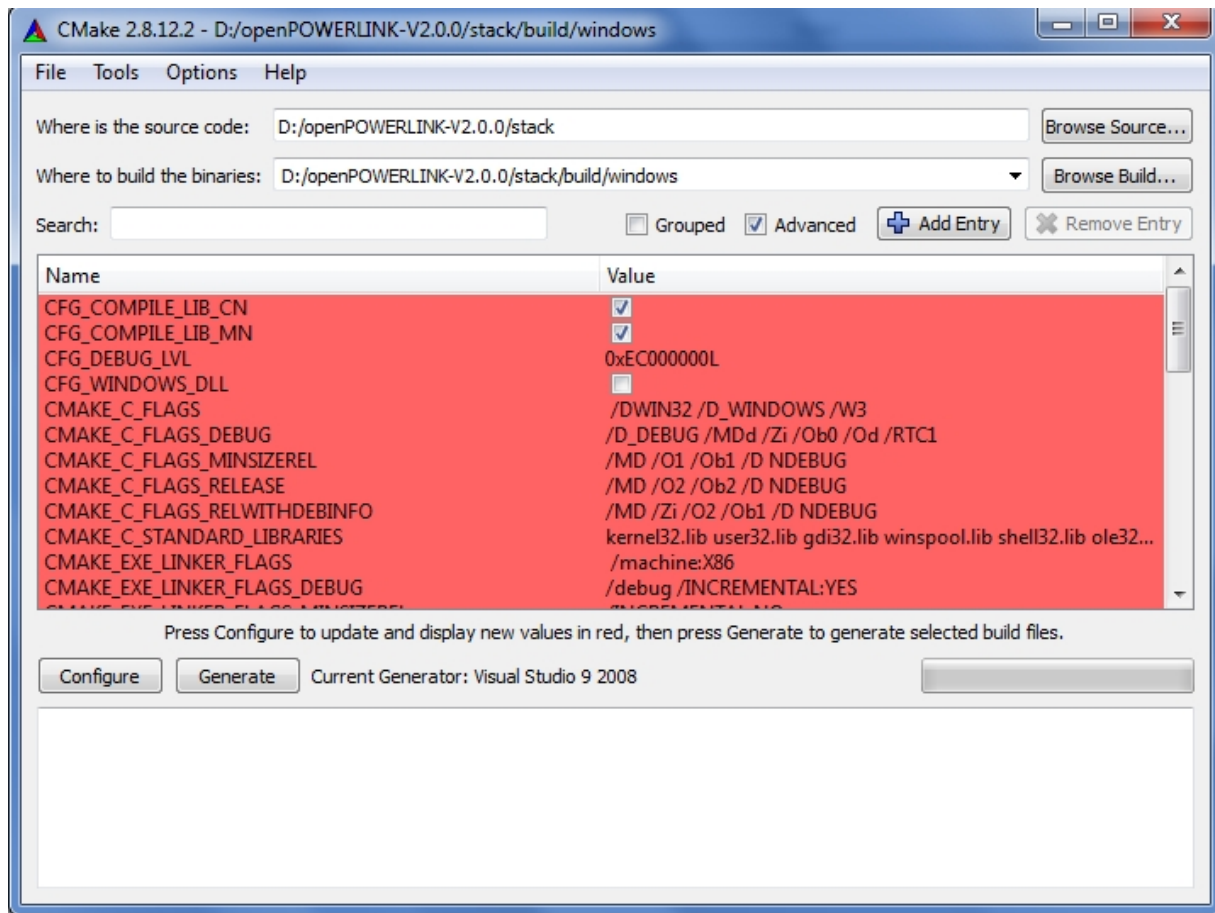
1.1 首先 build openPOWERLINK stack libraries.

该部分的说明参照“Build stack libraries”中关于 windows 部分的说明。

Main Page	Ethernet POWERLINK	openCONFIGURATOR	openPOWERLINK Stack	Software Manual	Contribute	Revision History
▼ openPOWERLINK Ethernet POWERLINK openCONFIGURATOR ▼ openPOWERLINK Stack Directory Structure Components ▼ Building Building openPOWERLINK Building Stack Libraries Building Drivers Building Demo Applications Building Hardware Platforms ▶ Supported Platforms ▶ Software Manual ▶ Contribute Windows Follow the steps below to build the stack on a Windows system using NMake. Open a Visual St commands: • Creating debug libraries <pre>> cd <openPOWERLINK_directory>\stack\build\windows > cmake -G"NMake Makefiles" -DCMAKE_BUILD_TYPE=Debug ..\.. > nmake > nmake install</pre> • Creating release libraries <pre>> cd <openPOWERLINK_directory>\stack\build\windows > cmake -G"NMake Makefiles" -DCMAKE_BUILD_TYPE=Release ..\.. > nmake > nmake install</pre>						

1.2 在 cmake 中生成 visual studio 的工程：

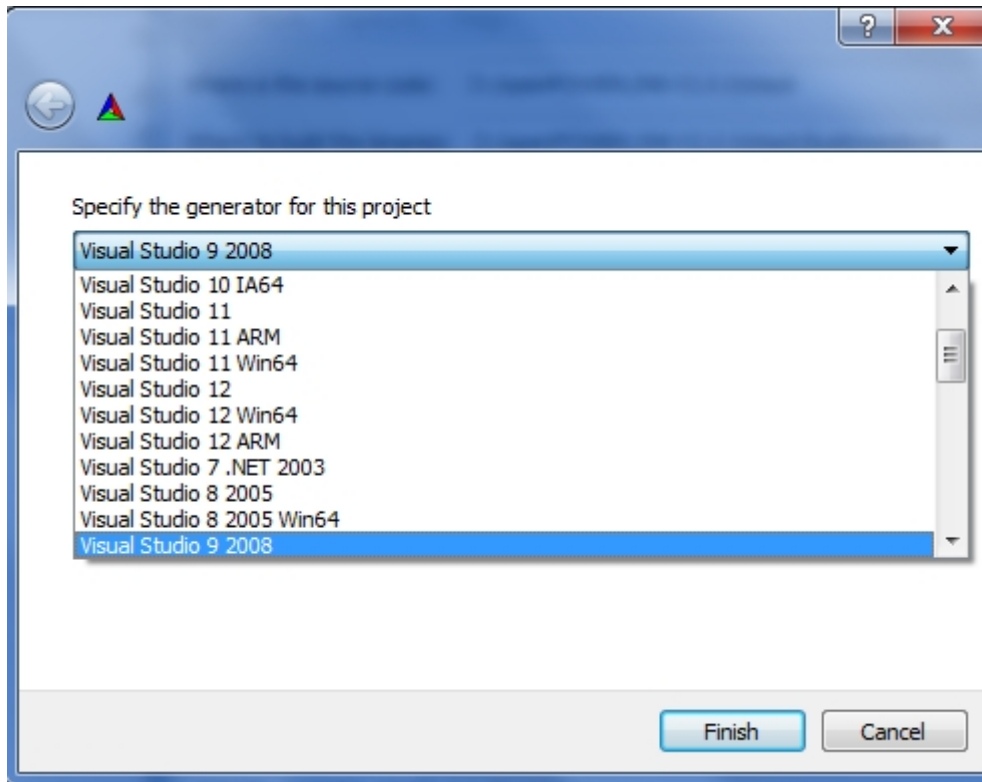
Cmake 的版本需要 2.8.7 以上。我使用的是 cmake-2.8.12.2-win32-x86。打开 cmake, 如下图所示：



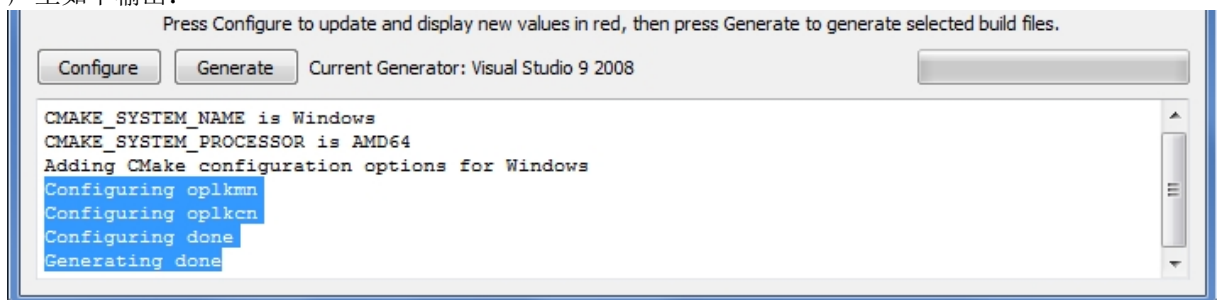
点击“Where is the source code”右边的“Browse Source...”，将其设置为<openPOWERLINK2.0的根目录>/stack。

点击“Where to build the binaries”右边的“Browse Build...”，将其设置为<openPOWERLINK2.0的根目录>/stack/build/windows。

最后点击“Generate”，会弹出如下对话框，提示要使用的visual studio的版本，我用的是vs2008，因此我选择“visual studio 9 2008”。

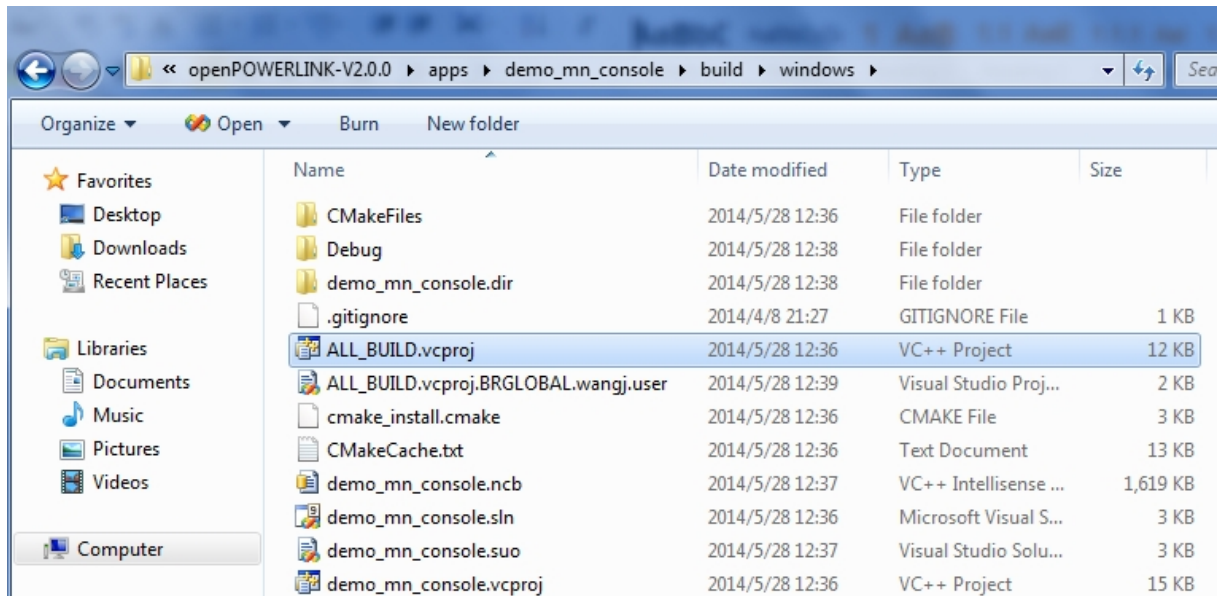


产生如下输出:



1.3 用 visual studio 编译:

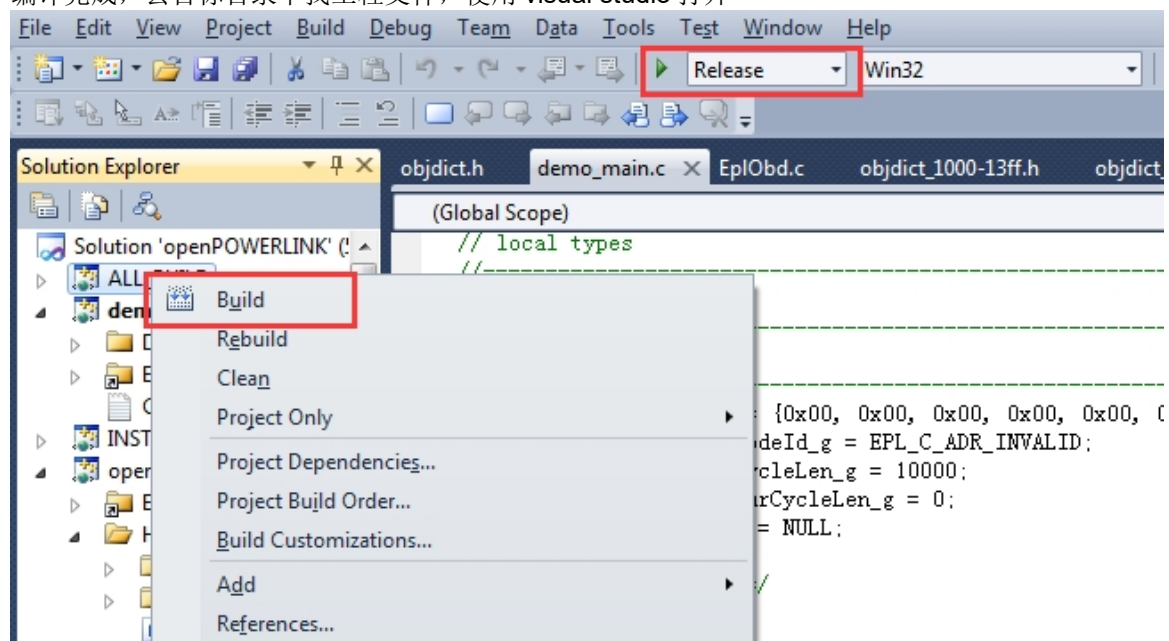
打开目录: <openPOWERLINK2.0 的根目录>\apps\demo_mn_console\build\windows。



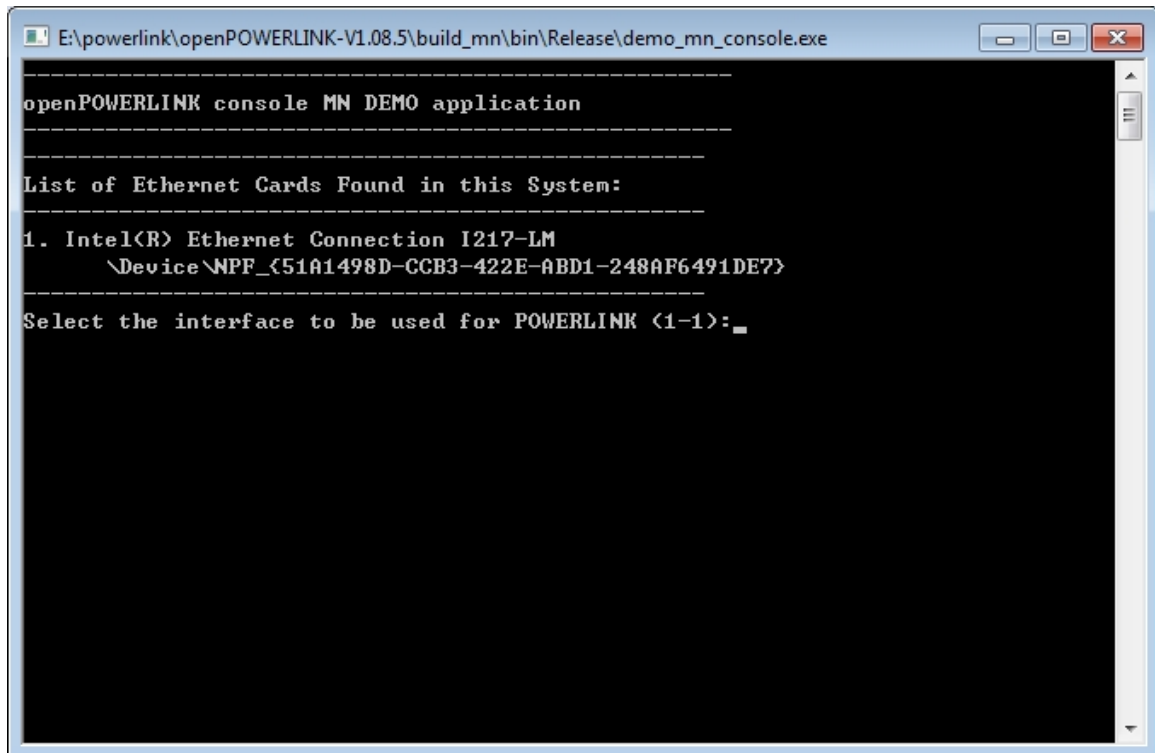
双击“ALL_BUILD.vcproj”，会自动启动 visual studio 2008。在 visual studio 右键单击，选中“ALL BUILD”在弹出的菜单中选择“生成”。这样就生成了应用程序，这个应用程序去调用前面生成的库。

2 执行

编译完成，去目标目录下找工程文件，使用 visual studio 打开



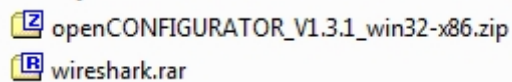
编译工程，然后选择 release 调试



选择网口，就能正常启动程序

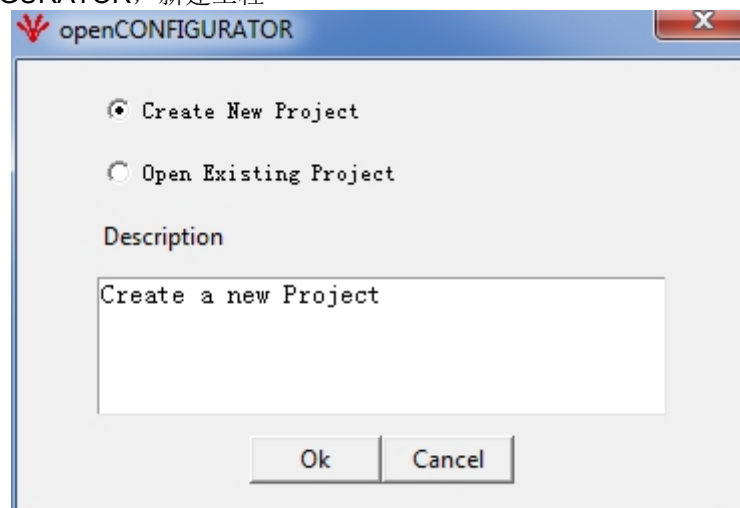
2.1 使用 openCONFIGURATOR 配置主从站通信

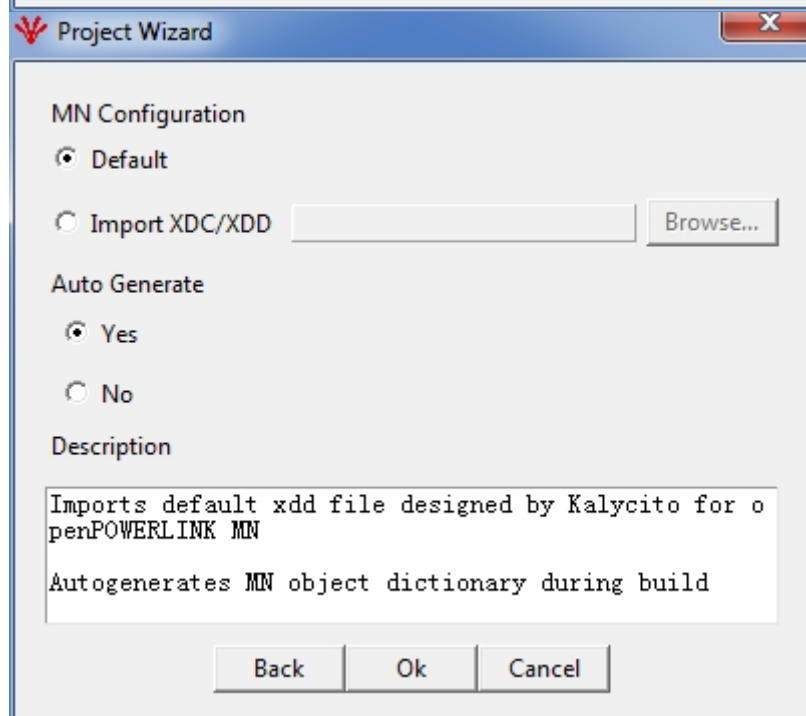
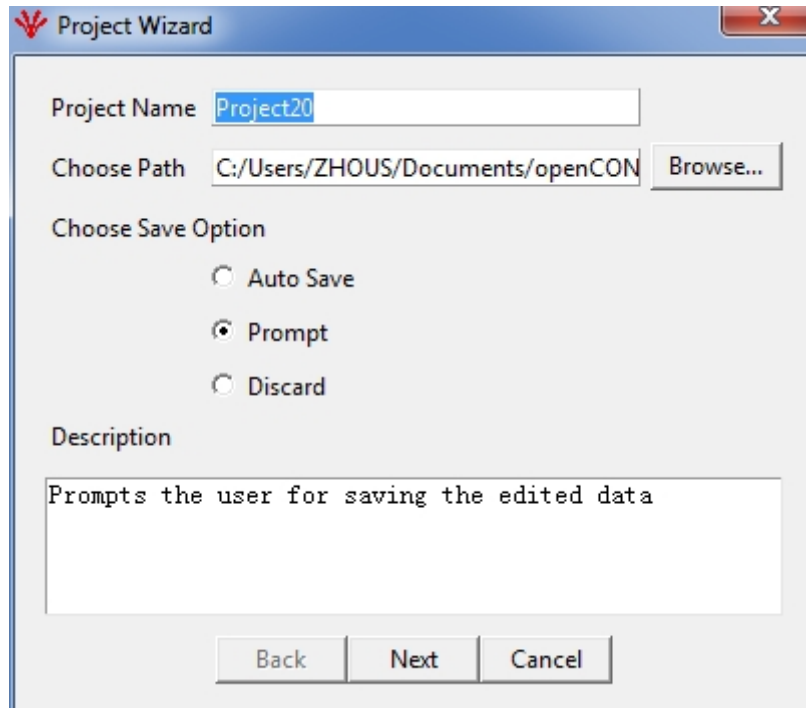
安装组网和抓包工具



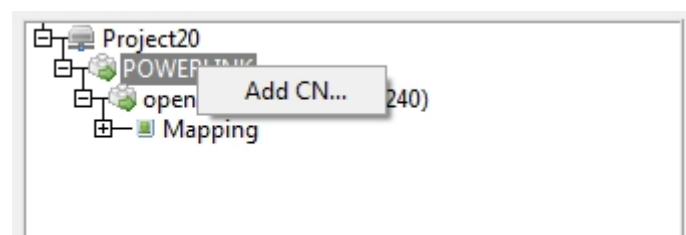
关于 POWERLINK 的组网，请参考相关的详细教程。

打开 openCONFIGURATOR，新建工程

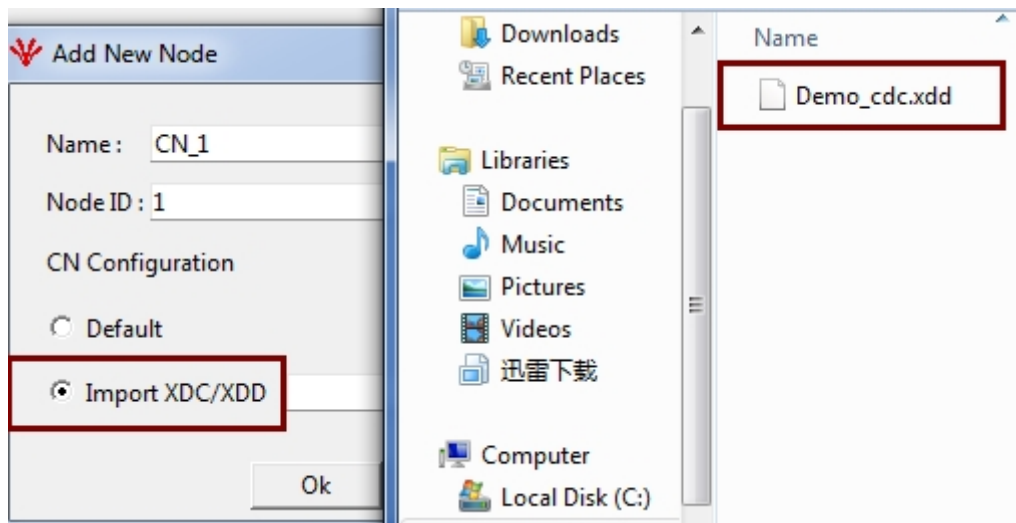




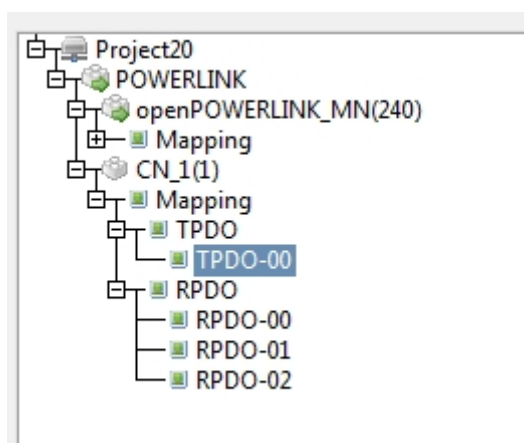
右键单击添加 CN



选择提供的 XDD 文件



配置 TPDO

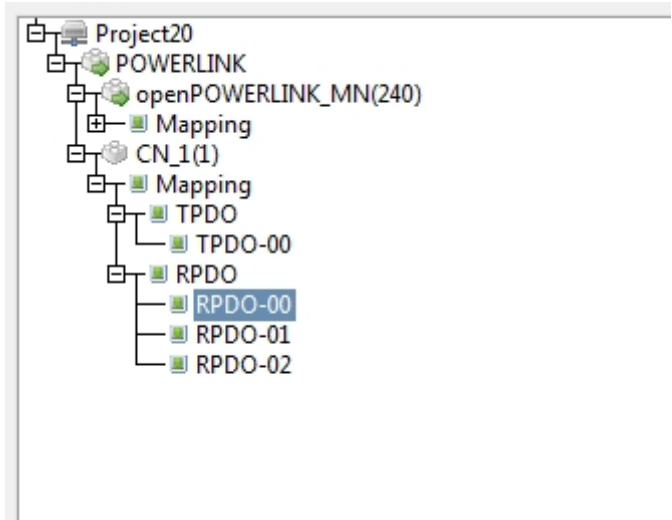


选择索引，子索引，单击 save

S.No	Object	Sub-object	Length	Offset
1	cn_2_mn_obj1 (0x6000)	sub_obj_01 (0x01)	0x0020	0x0000
2	cn_2_mn_obj1 (0x6000)	sub_obj_02 (0x02)	0x0020	0x0020
3	(0x0)	(0x0)	0x0	0x0040
4	(0x0)	(0x0)	0x0	0x0040
5	(0x0)	(0x0)	0x0	0x0040
6	(0x0)	(0x0)	0x0	0x0040

Save
Discard

配置 RPDO



同样选择索引，子索引， save

S.No	Object	Sub-object	Length	Offset
1	mn_2_cn_obj1 (0x6200)	sub_obj_01 (0x01)	0x0020	0x0000
2	mn_2_cn_obj1 (0x6200)	sub_obj_02 (0x02)	0x0020	0x0020
3	(0x0)	(0x0)	0x0	0x0040
4	(0x0)	(0x0)	0x0	0x0040
5	(0x0)	(0x0)	0x0	0x0040
6	(0x0)	(0x0)	0x0	0x0040

Save

Discard

选择循环周期

Network Browser

Project20

POWERLINK

openPOWERLINK_MN(240)

Mapping

CN_1(1)

Mapping

TPDO

TPDO-00

RPDO

RPDO-00

RPDO-01

RPDO-02

Properties

Node name

openPOWERLINK_MN

Node number

240

Cycle time

10000

祘

Advanced

Loss of SoC tolerance

300

祘

Asynchronous MTU size

300

Bytes

Asynchronous timeout

100000

ns

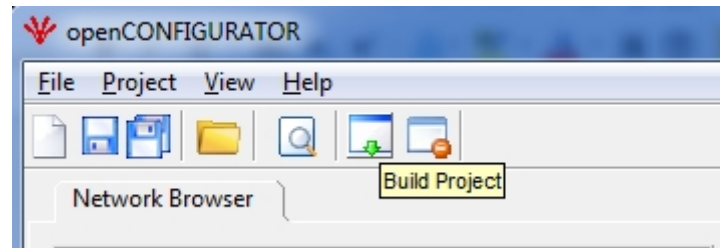
Multiplexing prescaler

0

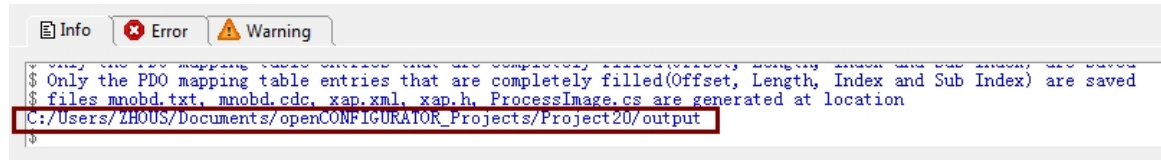
Save

Discard

编译工程



去生成路径寻找导出文件

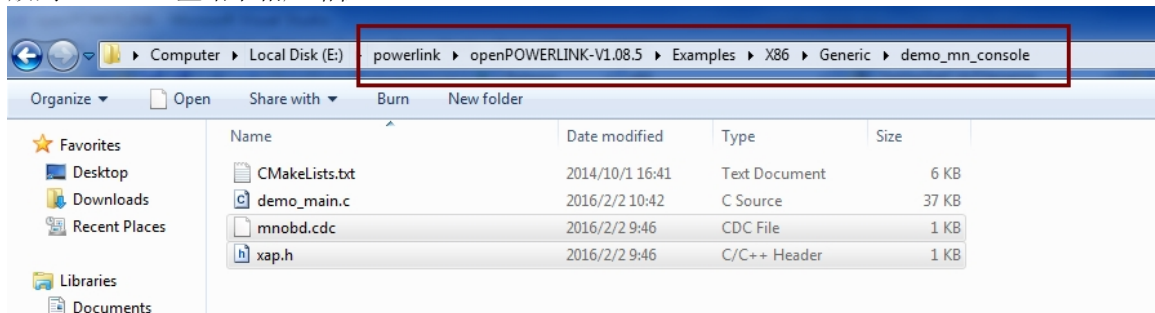


复制以下两个文件 “mnobd.cdc” 和 “xap.h”

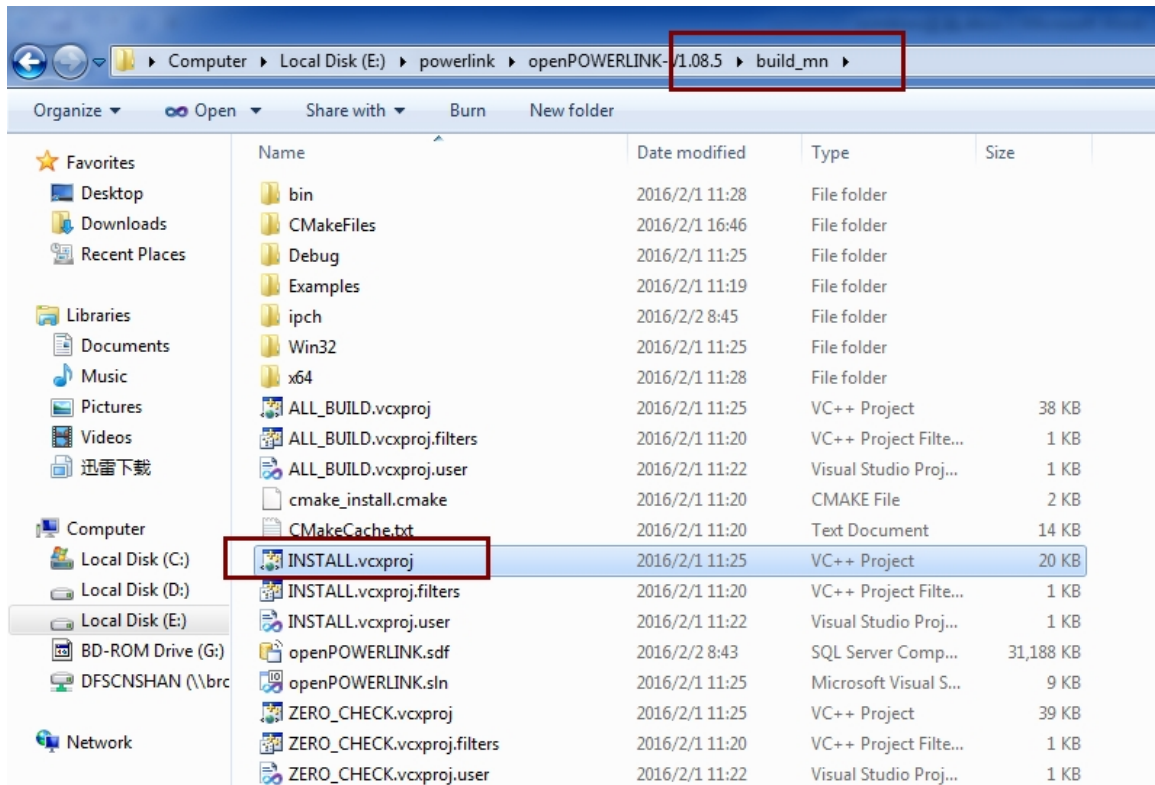
mnobd.cdc	2016/2/2 9:46	CDC File	1 KB
mnobd.txt	2016/2/2 9:46	Text Document	2 KB
ProcessImage.cs	2016/2/2 9:46	Visual C# Source f...	1 KB
xap.h	2016/2/2 9:46	C/C++ Header	1 KB
xap.xml	2016/2/2 9:46	BaiduBrowser HT...	1 KB

Type: BaiduBrowser HTML Document

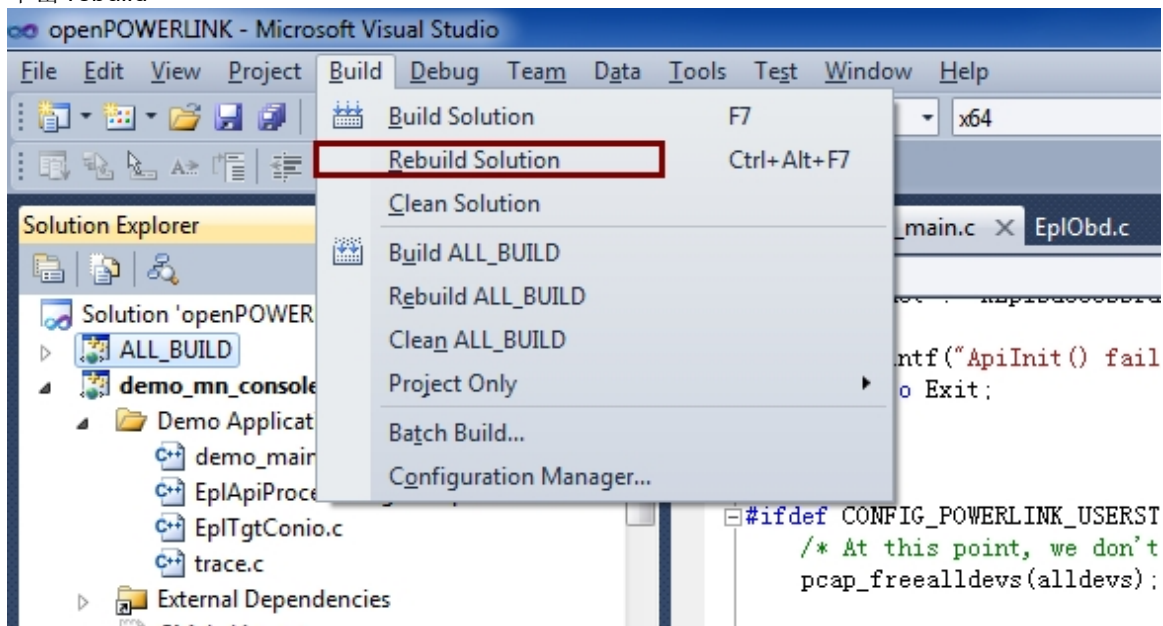
放到 windows 主站下相应路径



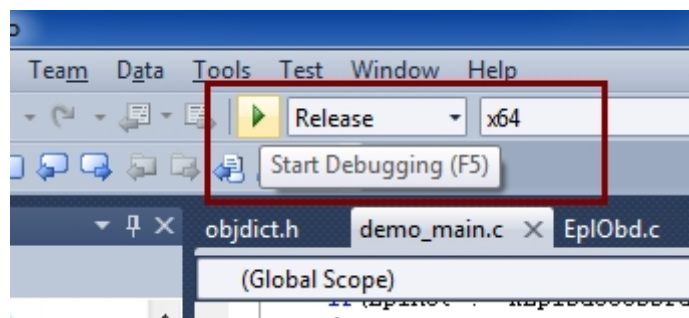
再次打开 windows 主站



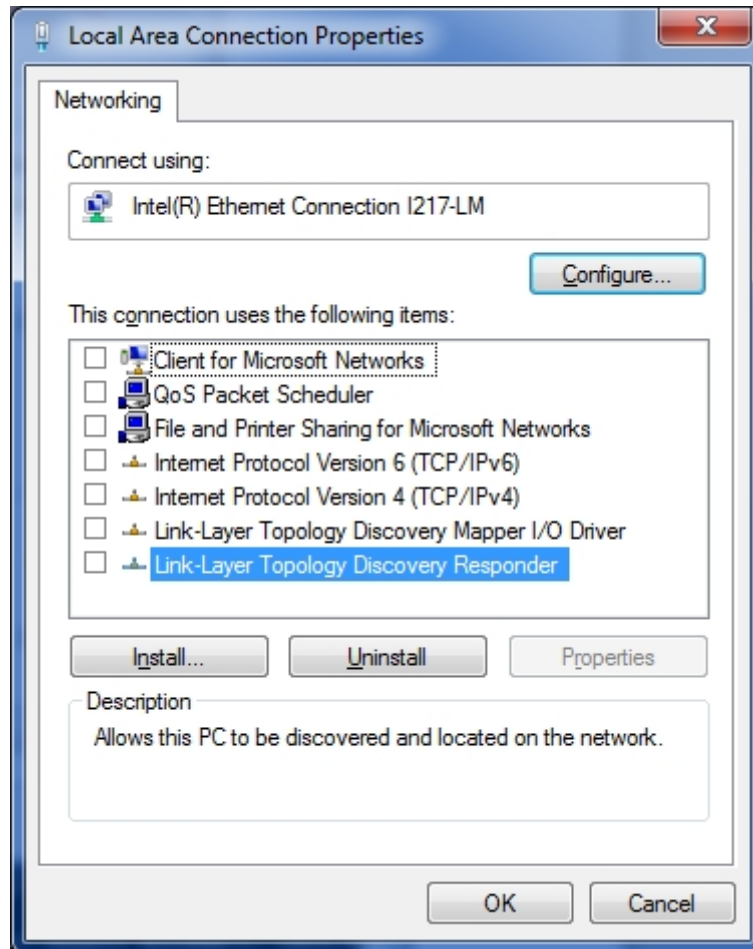
单击 rebuild



然后进行调试



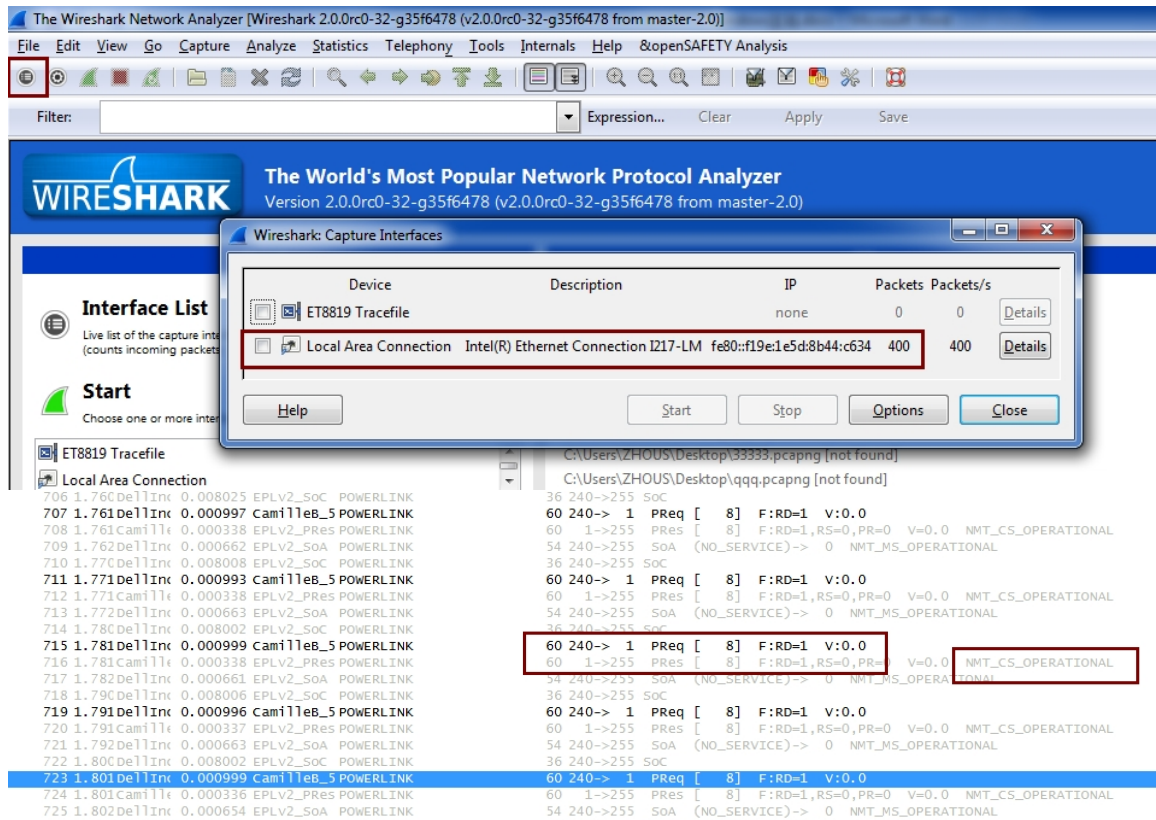
运行前需要关闭网口 TCP/IP，然后将网口用网线连接至 FPGA 从站



然后选择网口，运行 windows 主站，可以看到 CFM 配置信息，以及从站正常进入 operational 的状态

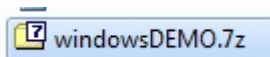
```
E:\powerlink\openPOWERLINK-V1.08.5\build_mn\bin\Release\demo_mn_console.exe
2/02 09:52:50 - 102/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1600/1, 2016/0
2/02 09:52:50 - 117/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1600/2, 2016/0
2/02 09:52:50 - 132/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1A00/1, 2016/0
2/02 09:52:50 - 147/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1A00/2, 2016/0
2/02 09:52:50 - 162/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1600/0, 2016/0
2/02 09:52:50 - 170/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1A00/0, 2016/0
2/02 09:52:50 - 178/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1010/1, 2016/0
2/02 09:52:50 - 182/190 Bytes2016/02/02 09:52:50 - -> SDO Abort=0x6020000, Error=0x0)
2016/02/02 09:52:50 - AppChEvent(Node=0x1, CFM-Progress: Object 0x1006/0, 2016/0
2/02 09:52:50 - 186/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppChEvent(Node=0x1, ConfReset)
2016/02/02 09:52:51 - AppChEvent(Node=0x1, NmtState=NmtCsNotActive)
2016/02/02 09:52:51 - AppChEvent(Node=0x1, NmtState=NmtCsPreOperational2)
2016/02/02 09:52:51 - AppChEvent(Node=0x1, Found)
2016/02/02 09:52:51 - AppChEvent(Node=0x1, NmtState=NmtCsReadyToOperate)
2016/02/02 09:52:51 - AppChEvent(Node=0x1, NmtState=NmtCsOperational)
```

可以使用 wireshark 进行抓包，查看通信是否正常

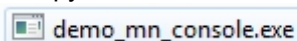


2.2 使用 Exe 直接运行 windows 主站进行组网（无 visual studio）

对于有些用户没有安装 Visual Studio，我们提供了 exe 的运行文件，直接打开即可打开



将上一章节配置的 cdc 与 xap.h 文件 copy 到 bin/release/下面，然后运行



同样可以方便的运行 windows 主站，但是无法修改 windows 主站的协议栈

```

E:\powerlink\openPOWERLINK-V1.08.5\build_mn\bin\Release\demo_mn_console.exe
2/02 09:52:50 - 102/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1600/1, 2016/0
2/02 09:52:50 - 117/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1600/2, 2016/0
2/02 09:52:50 - 132/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1A00/1, 2016/0
2/02 09:52:50 - 147/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1A00/2, 2016/0
2/02 09:52:50 - 162/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1600/0, 2016/0
2/02 09:52:50 - 170/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1A00/0, 2016/0
2/02 09:52:50 - 178/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1010/1, 2016/0
2/02 09:52:50 - 182/190 Bytes2016/02/02 09:52:50 - -> SDO Abort=0x6020000, Error=0x0)
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, CFM-Progress: Object 0x1006/0, 2016/0
2/02 09:52:50 - 186/190 Bytes2016/02/02 09:52:50 - >
2016/02/02 09:52:50 - AppCbEvent(Node=0x1, ConfReset)
2016/02/02 09:52:51 - AppCbEvent(Node=0x1, NmtState=NmtCsNotActive)
2016/02/02 09:52:51 - AppCbEvent(Node=0x1, NmtState=NmtCsPreOperational2)
2016/02/02 09:52:51 - AppCbEvent(Node=0x1, Found)
2016/02/02 09:52:51 - AppCbEvent(Node=0x1, NmtState=NmtCsReadyToOperate)
2016/02/02 09:52:51 - AppCbEvent(Node=0x1, NmtState=NmtCsOperational)
  
```

3 通信数据配置

如果通信能够正常建立，本部分用于验证数据是否能正常传输。

3.1 主站设定传输数据

在第三章放入配置组网文件的地方，查看 xap.h

```

# define COMPUTED_PI_OUT_SIZE 8
typedef struct
{
    unsigned CN1_M00_cn_2_mn_obj1_sub_obj_01:32;
    unsigned CN1_M00_cn_2_mn_obj1_sub_obj_02:32;
} PI_OUT;

# define COMPUTED_PI_IN_SIZE 8
typedef struct
{
    unsigned CN1_M00_mn_2_cn_obj1_sub_obj_01:32;
    unsigned CN1_M00_mn_2_cn_obj1_sub_obj_02:32;
} PI_IN;

#endif
  
```

可以看到，收发各 64 位，PI_OUT 为对于网络来讲的发出，则主站这边是接收，相应对于主站这边，PI_IN 为发送。

在 demo_main.c 里面找到相应的 AppCbSync()同步回调函数，该函数每周期执行一次，被用来作为数据更新处理的函数

修改函数内容，使用 AppProcessImageOut_g (AppProcessImageIn_g) +.(xap.h 内的相应内容)语句能够直接调用变量读写数据。

下图中将接收到的数据存入 `nodeVar_g[]`.`m_uiInput` 里面，并将发送的数据写入固定的值。
 然后每 500 个周期打印一次接收到的数据值。

```
objdict.h x demo_main.c x EplObd.c objdict_1000-13ff.h objdict_1b00-1fff.h EplCfg.h
(Global Scope) AppCbSync(void)
//
//-----
tEplKernel PUBLIC AppCbSync(void)
{
    tEplKernel      EplRet;
    static int      i=0;

    EplRet = EplApiProcessImageExchange(&AppProcessImageCopyJob_g);
    if (EplRet != kEplSuccessful)
    {
        return EplRet;
    }

    nodeVar_g[0].m_uiInput = AppProcessImageOut_g.CN1_M00_cn_2_mn_obj1_sub_obj_01;
    nodeVar_g[1].m_uiInput = AppProcessImageOut_g.CN1_M00_cn_2_mn_obj1_sub_obj_02;

    AppProcessImageIn_g.CN1_M00_mn_2_cn_obj1_sub_obj_01 = 0x01020304;
    AppProcessImageIn_g.CN1_M00_mn_2_cn_obj1_sub_obj_02 = 0xAABBCCDD;

    i++;
    if(i == 500)
    {
        printf("Receive is:%x,%x,%x\n",nodeVar_g[0].m_uiInput,nodeVar_g[1].m_uiInput);
        i = 0;
    }

    return EplRet;
}
```

修改完之后，编译整个工程。