CoordinatorEB.c部分：

#include "OSAL.h"

#include "AF.h"

#include "ZDApp.h"

#include "ZDObject.h"

#include "ZDProfile.h"

#include <string.h>

#include <ioCC2530.h>

#include "DebugTrace.h"

#include "GenericApp.h.h"

#if !defined( WIN32 )

#include "OnBoard.h"

#endif

/\* HAL \*/

#include "hal\_lcd.h"

#include "hal\_led.h"

#include "hal\_key.h"

#include "hal\_uart.h"

unsigned char uartbuf[2];

// This list should be filled with Application specific Cluster IDs.

const cId\_t GenericApp\_ClusterList[GENERICAPP\_MAX\_CLUSTERS] =

{

GENERICAPP\_CLUSTERID

};

const SimpleDescriptionFormat\_t GenericApp\_SimpleDesc =//简单设备描述符

{

GENERICAPP\_ENDPOINT, // int Endpoint;

GENERICAPP\_PROFID, // uint16 AppProfId[2];

GENERICAPP\_DEVICEID, // uint16 AppDeviceId[2];

GENERICAPP\_DEVICE\_VERSION, // int AppDevVer:4;

GENERICAPP\_FLAGS, // int AppFlags:4;

GENERICAPP\_MAX\_CLUSTERS, // byte AppNumInClusters;

(cId\_t \*)GenericApp\_ClusterList, // byte \*pAppInClusterList;

0, // 改 byte AppNumInClusters;

(cId\_t \*)NULL // 改 byte \*pAppInClusterList;

};

// This is the Endpoint/Interface description. It is defined here, but

// filled-in in GenericApp\_Init(). Another way to go would be to fill

// in the structure here and make it a "const" (in code space). The

// way it's defined in this sample app it is define in RAM.

endPointDesc\_t GenericApp\_epDesc;//节点描述符

byte GenericApp\_TaskID; // Task ID for internal task/event processing任务优先级

// This variable will be received when

// GenericApp\_Init() is called.

byte GenericApp\_TransID; // This is the unique message ID (counter)数据发送序列号

void GenericApp\_MessageMSGCB( afIncomingMSGPacket\_t \*pckt ); //消息处理函数

void GenericApp\_SendTheMessage( void );//数据发送函数

static void rxCB(uint8 port,uint8 event);

void GenericApp\_Init( byte task\_id )//任务初始化函数

{

halUARTCfg\_t uartConfig;

GenericApp\_TaskID = task\_id;

GenericApp\_TransID = 0;//发送序列包置零，每发送一个包，序列自加一

GenericApp\_epDesc.endPoint = GENERICAPP\_ENDPOINT;

GenericApp\_epDesc.task\_id = &GenericApp\_TaskID;

GenericApp\_epDesc.simpleDesc

= (SimpleDescriptionFormat\_t \*)&GenericApp\_SimpleDesc;

GenericApp\_epDesc.latencyReq = noLatencyReqs;

// Register the endpoint description with the AF

afRegister( &GenericApp\_epDesc );//节点注册，注册后才能使用OSAL提供的系统服务

uartConfig.configured =TRUE;

uartConfig.baudRate =HAL\_UART\_BR\_9600;

uartConfig.flowControl =FALSE;

uartConfig.callBackFunc =rxCB;

HalUARTOpen(0,&uartConfig);

}

static void rxCB(uint8 port,uint8 event)

{

HalUARTRead(0,uartbuf,2);

GenericApp\_SendTheMessage();

}

UINT16 GenericApp\_ProcessEvent( byte task\_id, UINT16 events )

{

afIncomingMSGPacket\_t \*MSGpkt;

if ( events & SYS\_EVENT\_MSG )

{

MSGpkt = (afIncomingMSGPacket\_t \*)osal\_msg\_receive( GenericApp\_TaskID );

while ( MSGpkt )

{

switch ( MSGpkt->hdr.event )

{

case AF\_INCOMING\_MSG\_CMD://是指的收到从其他节点应用层发给自己的数据包

GenericApp\_MessageMSGCB( MSGpkt );

break;

default:

break;

}

osal\_msg\_deallocate( (uint8 \*)MSGpkt );//释放内存

MSGpkt = (afIncomingMSGPacket\_t \*)osal\_msg\_receive( GenericApp\_TaskID );

}

return (events ^ SYS\_EVENT\_MSG);

}

return 0;

}

void GenericApp\_MessageMSGCB( afIncomingMSGPacket\_t \*pkt )//消息处理

{

unsigned char buffer[10];

switch ( pkt->clusterId )

{

case GENERICAPP\_CLUSTERID: // "the" message

osal\_memcpy(buffer,pkt->cmd.Data,5);//将数据拷贝到缓冲区

HalUARTWrite(0,buffer,5); //往串口写数据，送往PC

HalLedBlink(HAL\_LED\_3,0,50,500);

break;

}

}

void GenericApp\_SendTheMessage( void )

{

afAddrType\_t my\_DstAddr;

my\_DstAddr.addrMode = (afAddrMode\_t)AddrBroadcast;

my\_DstAddr.endPoint = GENERICAPP\_ENDPOINT;

my\_DstAddr.addr.shortAddr = 0xFFFF;

AF\_DataRequest( &my\_DstAddr, &GenericApp\_epDesc,

GENERICAPP\_CLUSTERID,

osal\_strlen(uartbuf)+1,

uartbuf,

&GenericApp\_TransID,

AF\_DISCV\_ROUTE, AF\_DEFAULT\_RADIUS );

HalLedBlink(HAL\_LED\_1,0,50,800);

}

CoordinatorEB.h部分：

#include "ZComDef.h"

#define GENERICAPP\_ENDPOINT 10

#define GENERICAPP\_PROFID 0x0F04

#define GENERICAPP\_DEVICEID 0x0001

#define GENERICAPP\_DEVICE\_VERSION 0

#define GENERICAPP\_FLAGS 0

#define GENERICAPP\_MAX\_CLUSTERS 1

#define GENERICAPP\_CLUSTERID 1

extern void GenericApp\_Init(byte task\_id);

extern UINT16 GenericApp\_ProcessEvent(byte task\_id,UINT16 evens);