Chipcon Products from Texas Instruments

Migrating Application from 1.4.X

Document Number: F8W-2007-0027

Texas Instruments, Inc.
San Diego, California USA
(619) 497-3845

Version	Description	Date
1.0	Initial release.	11/08/2007

Table of Contents

1. Purpose

This document describes, step by step, how to migrate from an application written for Z-Stack-1.4.2 or earlier to Z-Stack-1.4.3. The reason for this migration is a change in the design of the ZDO module to reduce code size. The sample application, TransmitApp, will be used as the example application.

2. Assumptions

You are already familiar with the lay out of a Z-Stack application and you are already familiar with TransmitApp (or another Z-Stack sample application, such as GenericApp).

3. Changes to ZDO

The ZDO layer was changed to include a generic interface to deliver ZDO service discovery information to an application without changing code in the ZDO modules (ZDApp.c, ZDObject.c and ZDProfile.c). This change also reduced the code size of the ZDO module by only including parsing functions when needed.

Read or skim through the following:

- “Z-Stack Developer's Guide (F8W-2006-0022).pdf”, section 6 – “ZDO Message Requests”
- “Z-Stack API _F8W-2006-0021_.pdf”, section 3.1.7 – “ZDO Parsing Functions”

4. Changes to TransmitApp

The changes that we are making to TransmitApp all deal with binding or finding a destination address. In the application, we use Switch 2 to do an End Device Bind and Switch 4 to find a matching device. [The descriptions of these 2 features are not covered by this document.]

Before, the application would call helper functions in ZDO to send the End Device Bind or Match Descriptor, and when the action was completed ZDO would send a ZDO_NEW_DSTADDR OSAL message to the event processor function to set a new destination address. These 2 helper functions were deleted.

Now, the application will:

- Register for ZDO Service Discovery Response messages
- Send a ZDO Service Discovery Request message by calling a ZDO Profile function
- Receive the ZDO Service Discovery Response message (through a new OSAL message - ZDO_CB_MSG)
- Call a ZDO Parsing function to parse the received message into a structure.
- Process the received response message

First, we will need to add an include file for the new parsing functions:

```
#include "ZDObject.h"
```

Next, add a function prototype for the new response message processor:

```
/*********************************************************************
 * LOCAL FUNCTIONS
 */
void TransmitApp_ProcessZDOMsgs( zdoIncomingMsg_t *inMsg );
```

We will need to register for 2 ZDO Service Discovery Response messages. At the bottom of `TransmitApp_Init(),` add the following lines of code:

```
ZDO_RegisterForZDOMsg( TransmitApp_TaskID, End_Device_Bind_rsp );
ZDO_RegisterForZDOMsg( TransmitApp_TaskID, Match_Desc_rsp );
```

Next, since the ZDO_NEW_DSTADDR OSAL message doesn't exist anymore, remove the following lines of code from `TransmitApp_ProcessEvent()`:

```
byte dstEP;
zAddrType_t *dstAddr;
ZDO_NewDstAddr_t *ZDO_NewDstAddr;
```

And remove:

```
case ZDO_NEW_DSTADDR:
  ZDO_NewDstAddr = (ZDO_NewDstAddr_t *)MSGpkt;

  dstEP = ZDO_NewDstAddr->dstAddrDstEP;
  dstAddr = &ZDO_NewDstAddr->dstAddr;
  TransmitApp_DstAddr.addrMode = (afAddrMode_t)dstAddr->addrMode;
  TransmitApp_DstAddr.endPoint = dstEP;
  if ( dstAddr->addrMode == Addr16Bit )
  {
    TransmitApp_DstAddr.addr.shortAddr = dstAddr->addr.shortAddr;
  }
  break;
```

Add code to receive the new ZDO Service Discovery Callback/Response message in `TransmitApp_ProcessEvent()`. Change:

```
case KEY_CHANGE:
  TransmitApp_HandleKeys( ((keyChange_t *)MSGpkt)->state,
                              ((keyChange_t *)MSGpkt)->keys );
  break;
```

To:

```
case ZDO_CB_MSG:
  TransmitApp_ProcessZDOMsgs( (zdoIncomingMsg_t *)MSGpkt );
  break;

case KEY_CHANGE:
  TransmitApp_HandleKeys( ((keyChange_t *)MSGpkt)->state,
                              ((keyChange_t *)MSGpkt)->keys );
  break;
```

Add the following function to process the incoming ZDO Service Discovery Message:

```
/*********************************************************************
```

We need to change the functions that are called when a key is pressed. In TransmitApp_HandleKeys() , add the following to the top of the function:

```
zAddrType_t dstAddr;
```

In TransmitApp_HandleKeys() , change the following:

```
if ( keys & HAL_KEY_SW_2 )
{
  // Initiate an End Device Bind Request for the mandatory endpoint
  ZDApp_SendEndDeviceBindReq( TransmitApp_epDesc.endPoint );
}
```

to:

```
if ( keys & HAL_KEY_SW_2 )
{
  HalLedSet ( HAL_LED_4, HAL_LED_MODE_OFF );

  // Initiate an End Device Bind Request for the mandatory endpoint
  dstAddr.addrMode = Addr16Bit;
  dstAddr.addr.shortAddr = 0x0000; // Coordinator
  ZDP_EndDeviceBindReq( &dstAddr, NLME_GetShortAddr(),
       TransmitApp_epDesc.endPoint,
       TRANSMITAPP_PROFID,
       TRANSMITAPP_MAX_CLUSTERS, (cId_t *)TransmitApp_ClusterList,
       TRANSMITAPP_MAX_CLUSTERS, (cId_t *)TransmitApp_ClusterList,
       FALSE );
}
```

And lastly, in TransmitApp_HandleKeys() , change the following:

```
if ( keys & HAL_KEY_SW_4 )
{
  // Initiate a Match Description Request (Service Discovery)
  //  for the mandatory endpoint
  ZDApp_AutoFindDestination( TransmitApp_epDesc.endPoint );
}
```

to:

```
if ( keys & HAL_KEY_SW_4 )
{
  HalLedSet ( HAL_LED_4, HAL_LED_MODE_OFF );

  // Initiate a Match Description Request (Service Discovery)
  dstAddr.addrMode = AddrBroadcast;
  dstAddr.addr.shortAddr = NWK_BROADCAST_SHORTADDR;
  ZDP_MatchDescReq( &dstAddr, NWK_BROADCAST_SHORTADDR,
      TRANSMITAPP_PROFID,
      TRANSMITAPP_MAX_CLUSTERS, (cId_t *)TransmitApp_ClusterList,
      TRANSMITAPP_MAX_CLUSTERS, (cId_t *)TransmitApp_ClusterList,
      FALSE );
}
```