Chipcon Products from Texas Instruments

Application Note: Protocol Versioning and 16-bit Cluster IDs

Document Number: F8W-2006-0006

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

Version	Description	Date
1.0	Initial release.	05/01/2006
1.1	Changed document name. Updated title page.	05/21/2007

Table of Contents

1. Purpose

This document summarizes the effects of implementing ZigBee Protocol Version 1.1. In particular is the interoperability behavior with respect to Version 1.0 networks and the effects of 16-bit Cluster IDs.

2. Protocol version interoperability

2.1 Mixed protocol versions

The ZigBee Specification precludes Protocol Version 1.0 interoperability with Version 1.1 on the same PAN. That is, each PAN runs either Version 1.0 or Version 1.1. The Z-Stack distributions beginning with Version 1.4.0 comply with this requirement

2.2 Determine Protocol Version

2.2.1 Runtime support in Z-Stack

The ZigBee specification now calls out the attribute *nwkProtocolVersion* in the Network Information Base (**`NIB)`** to hold the network protocol version. Getting and setting this attribute is supported by the **`NLME_GetRequest()`** and **`NLME_SetRequest()`** APIs. The **`NIB`** attribute ID is **`0x98`**. There is also a new API

```
byte NLME_GetProtocolVersion()
```

which wraps the **`NLME_GetRequest()`** for the protocol version and returns the protocol version directly.

The legal values are:

```
0x01 for ZigBee Protocol Version 1.0
0x02 for ZigBee Protocol Version 1.1
```

When setting a value the value is not range checked so it is possible to set an illegal value for the protocol version. This feature can be used to force the device to behave in a specific way as defined below in Table 1.

2.2.2 Z-Tool Support

The protocol version may also be read and set using Z-Tool. Using Z-Tool you can gain access to the **`NLME_GetRequest() and NLME_SetRequest()`** functions. The Z-Tool commands are **`NLME_GET_REQUEST`** and **`NLME_SET_REQUEST`**, respectively. The **`NIBAttribute`** parameter should be set to 0x98. When setting a value the interface will permit setting a value other than a legal value. This feature can be used to force the device to behave in a specific way as defined below in Table 1.

2.2.3 Setting protocol version on-the-fly

If setting the protocol version has changed the version that was running it is strongly recommended that the device be reset. The reason is that there are a number of runtime behaviors that are controlled by the version. If the version is changed on-the-fly and the network does not conform to the new version things may not operate correctly.

2.3 Backward compatibility

The ZigBee specification permits backward compatibility. Devices that support Protocol Version 1.1 may also support Protocol Version 1.0. The Z-Stack distributions beginning with Version 1.4.0 support backward compatibility.

2.3.1 Joining a network

The following Table describes the behavior of a joining device for Z-Stack distributions beginning with Version 1.4.0. A new preprocessor macro **`DEF_PROTO_VERS`** may be used to control the default protocol version of the network to be joined. There are guards in the code to prevent illegal and missing definitions in the IDE. Legal values are:

```
0x01 for ZigBee Protocol Version 1.0

0x02 for ZigBee Protocol Version 1.1
```

Note that for the IAR IDE simply defining **`DEF_PROTO_VERS`** is equivalent to **`DEF_PROTO_VERS=1`**. Since 1 is a legal value the build will complete normally with the default protocol version being set to 1.0. The device will only join a 1.0 network.

The macro takes precedence in the run-time determination of which protocol version is in force. That is, if the macro is defined it is respected independent of all other contexts.

When a device joins a network the version under which it joins will be set in the **`NIB`** and therefore in NV as well. Thus NV will always reflect the network version of the most recently joined network.

PAN	Macro	Behavior	Comments
PAN ID Configured	DEF_PROTO_VERS defined	Will join with the protocol version specified by DEF_PROTO_VERS. Value saved in NV.	If PAN does not support the version specified by the macro the **device will not join**.
	DEF_PROTO_VERS not defined	Will join with the protocol version supported by that PAN.	Either version 1.1 or Version 1.0.
PAN ID Not configured	DEF_PROTO_VERS defined	Will join the first network with the protocol version specified by DEF_PROTO_VERS. Value saved in NV.	If no networks match protocol specified by the macro the **device will not join**.
	DEF_PROTO_VERS not defined	Will join the first network found whose version matches version in NV.	If NV is valid and if no networks with that version found **device will not join**. If NV not valid, will join first 1.1 network. If no 1.1 networks then join first 1.0 network.

Table 1: Device Joining Behavior

The logic for controlling how the device joins is implemented in the routine **`ZDApp.c: ZDO_NetworkDiscoveryConfirmCB().`** Conditioning parameters in support of this code can be found in **`ZDApp.c:ZDApp_Init()`**.

2.3.2 Orphan joins

An orphan join presents an opportunity for confusion. A device once joined will not re-associate when trying to join a network after loss of parent. Instead, an Orphan Request will be sent. If the target network has changed versions the orphan join will succeed but the device will never get any packets. The joining device has no way to validate the network version when doing an orphan join. During the initial association the protocol version is available in the beacon response. But this information is not present during the exchange required for an orphan join.

This is not a likely scenario in general but it could happen during a network version upgrade.

2.3.3 Starting a network

A ZigBee Coordinator will behave as described in the following Table. The new preprocessor macro **`DEF_PROTO_VERS`** controls the default protocol version. There are guards in the code to prevent illegal and missing definitions in the IDE. The default is Version 1.1. As with a device joining the network legal values are

```
0x01 for ZigBee Protocol Version 1.0
0x02 for ZigBee Protocol Version 1.1
```

Note that for the IAR IDE simply defining **`DEF_PROTO_VERS`** is equivalent to **`DEF_PROTO_VERS=1`**. Since 1 is a legal value the build will complete normally with the default protocol version being set to 1.0. A 1.0 network will be started.

As with control during joining a network the macro takes precedence in the run-time determination of which protocol version is in force. That is, if the macro is defined it is respected independent of all other contexts.

The conditional references to the macro are in **`ZDApp.c`**. This will permit users who have only library distributions to modify the default behaviors of the macro definition.

Macro	Behavior	Comments
DEF_PROTO_VERS defined	Will start network with the protocol version defined NV_ DEF_PROTO_VERS. Value saved in NV.	Protocol version defined in NV will be updated to reflect NV_ DEF_PROTO_VERS.
DEF_PROTO_VERS not defined	Will start network with the protocol version defined in NV	If no valid protocol defined will start a Version 1.1 network. Protocol version defined in NV will be updated.

Table 2: Network Version Establishment Behavior

The logic for controlling what version network is started is implemented in the routine **`ZDApp.c:ZDApp_Init().`** Look there if you want to change the default protocol version when the macro is not present and NV is not defined. Look for a code comment with the phrase '**`CUSTOMER NOTE`**'.

3. Cluster IDs

3.1 General behavior

ZigBee Protocol Version 1.1 supports 16-bit cluster IDs. If a Protocol Version 1.1 compliant device joins a Protocol Version 1.0 network the cluster ID will be truncated. Only the low-order byte will be encapsulated in the APS frame for outbound frames. The high-order byte will be lost when the frame is sent. On inbound frames the high order byte of the cluster ID will be set to 0x0[1].

3.2 A note on ZDO response Cluster IDs

The ZigBee Device Object uses the high-order bit of the cluster ID to represent a ZDO response. In Protocol Version 1.0 where the Cluster IDs are 8 bits the msb (bit 7) is used. The Protocol Version 1.1 specification keeps the use of the 'msb' but the msb is bit 15 for the 1.1 Version. If a Protocol Version 1.1 compliant device comes up as a Protocol Version 1.0 device the correct location of the 'msb' is managed in the Z-Stack code.

[1] But see Section 3.2 with respect to ZDO response frames.

If you have written an application that uses this same msb mechanism to distinguish requests from responses you should review the application design to accommodate the compatibility design of the Z-Stack distributions beginning with Version 1.4.0 as shipped. In particular, since the high-order byte is discarded on sending and set to 0x0 on receiving when a 1.1-compliant device runs on a 1.0 network, the use of the ‘msb’ should be reviewed. One option is to not relocate the response bit and leave it at bit 7.