Application Note: JN-AN-1135

Smart Energy HAN Solutions

This Application Note introduces a range of NXP solutions for the ZigBee PRO Smart Energy market. The current Home Area Network offering includes the following devices:

- **A combined Energy Service Portal and Electricity Meter**
- **An In-Premise Display**
- **Standalone Meters (Electricity and Gas)**
- **A Range Extender**

The accompanying software uses the ZigBee PRO SE clusters to transfer data between the various devices in the network. These devices were developed using NXP's ZigBee PRO, JenOS, Smart Energy and JN516x Integrated Peripheral APIs.

1 Introduction

This Application Note demonstrates a typical Smart Energy (SE) Home Area Network (HAN), for use in a domestic household, based on the NXP JN516x microcontrollers. The network conforms to the ZigBee Alliance's Smart Energy Profile (SEP) to provide the consumer with the following information:

- Energy Consumption
- Price Tariffs
- Text Messaging
- Demand Response Load Control (DRLC) events
- Time

The Application Note is also capable of remotely updating device firmware via the Over-The-Air (OTA) upgrade cluster.

This document uses terminology from ZigBee SEP v1.0 and therefore refers to an 'Energy Service Portal' device and 'Simple Metering' cluster (known respectively as an 'Energy Service Interface' and 'Metering' cluster in SEP v1.1).

1.1 System Overview

The example network consists of a combined Energy Service Portal and Electricity Meter (ESP-EM), an In-Premise Display (IPD), a standalone Gas Meter, a standalone Electric Meter and a Range Extender. The sections below provide a brief introduction to each node, for advanced user information refer to Section 5.

1.1.1 Combined Energy Service Portal and Electricity Meter

The ESP is the ZigBee Co-ordinator and Trust Centre, and is responsible for forming the network and controlling which nodes are allowed to join the network securely. In a typical deployment, the ESP would have an out-of-band connection to the utility provider – this connection is known as the 'backhaul'. The purpose of the backhaul connection is to allow

 © NXP Laboratories UK 2013

utility companies to remotely read metering data and provide dynamic information to the consumer, such as new price tariffs or DRLC events.

The node also includes a simulated electricity meter.

1.1.2 In-Premise Display

The IPD is a Sleeping End Device that allows multiple ESP and Simple Metering server endpoints to be queried. The node wakes up and gathers information from the network every 7 seconds. This information is then provided to the consumer via an LCD.

1.1.3 Standalone Electricity & Gas Meters

The standalone Meter nodes are permanently powered devices that act as routers once they have joined the network. As with the ESP-EM, both nodes maintain a Simple Metering server with a simulated load, which the IPD can then query.

1.1.4 Range Extender

The Range Extender node is a router that can be used for extending the range of the SE-HAN. It is acts as template for Manufacturer Specific cluster or profile development.

1.2 Network Architecture

Figure 1 below shows one example of how an SE-HAN can be deployed:

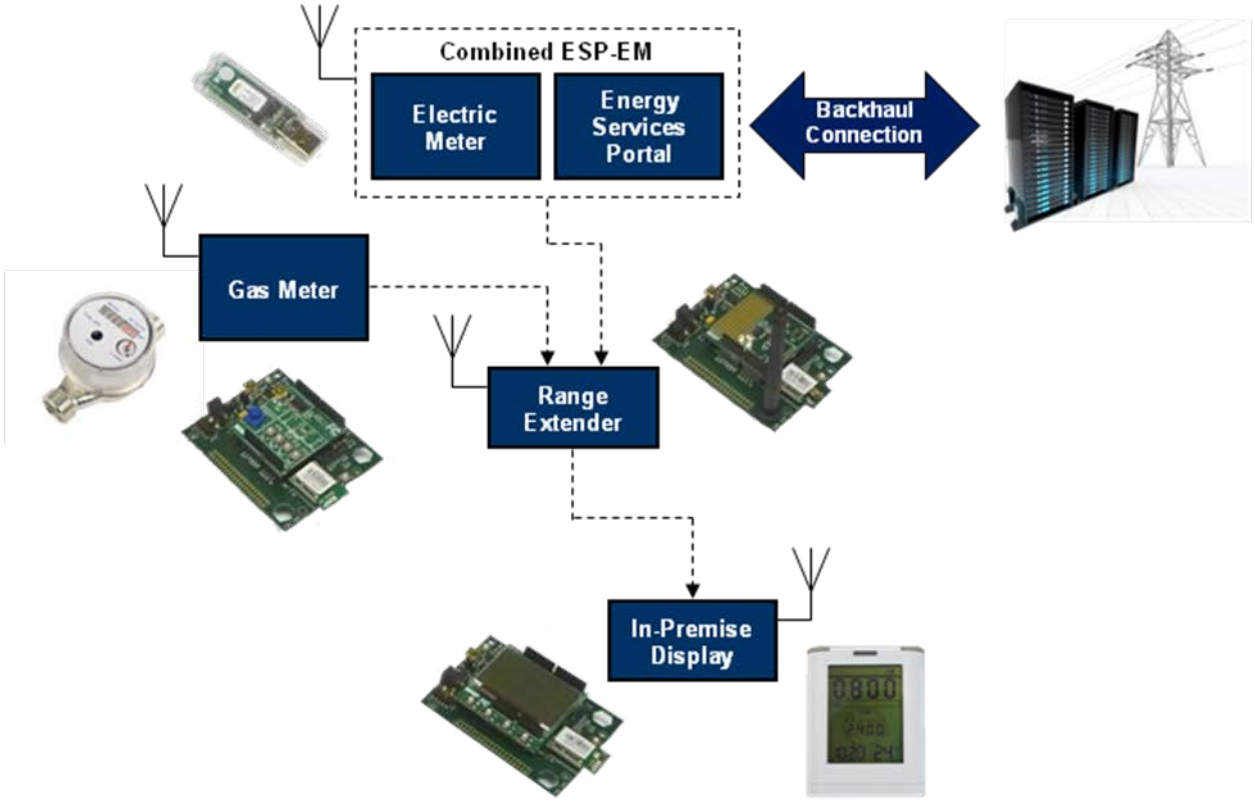

Figure 1: Example SE-HAN Deployment

 © NXP Laboratories UK 2013

2 Compatibility

The software provided with this Application Note has been tested with the following evaluation kit and SDK (Software Developer's Kit) versions:

Product Type	Part Number	Version or Build	Supported Chips
Evaluation Kit	JN516x-EK001	-	JN5168
JN516x ZigBee SE SDK Libraries	JN-SW-4064	993	JN5168
SDK Toolchain	JN-SW-4041	v1.1	JN5168

3 Loading the Application

Table 1 below shows a compatibility matrix of which JN516x Evaluation Kit expansion boards can be used with this Application Note:

Device Type	Carrier Board	Expansion Board (+ Carrier Board)			JN5168 USB Dongle
		Generic	LCD	Lighting/Sensor	
ESP-EM (ESP_METER_NODE)	○	●			○
IPD (IPD_NODE_EVK)			●		
Electricity Meter (PLUG_METER_NODE)	○	●		○	○
Gas Meter (GAS_METER_NODE)	○	●		○	○
Range Extender (RANGE_EXT_NODE)	○	●	●	●	●

Table 1: Device Type – Evaluation Kit Compatibility Matrix

Key: ● – Preferred Hardware ○ – Reduced Functionality

The application binaries, shown in brackets above, are located within each node's build folder, e.g. **JN-AN-1135-Smart-Energy-HAN-Solutions\ESP_METER_NODE\Build**. The binaries must be loaded into the corresponding JN516x Evaluation Kit (JN516x-EK001) boards using the JN51xx Flash Programmer. For more information refer to the *JN51xx Flash Programmer User Guide (JN-UG-3007)*.

! ***Caution:*** *If loading this application for the first time the persistent data must be cleared in each of the devices using the 'Erase EEPROM' option in the JN51xx Flash Programmer. Failure to do so will result in unpredictable network behaviour.*

Note: For backwards compatibility, the application note can be made to support the JN5148 evaluation kit (JN514x-EK010) when combined with a JN516x module upgrade kit (JN516x-UG001), for more information refer to Section 6.2.

 © NXP Laboratories UK 2013

4 Running the Demonstration

This section describes how to demonstrate the SE-HAN application once the JN516x Evaluation Kit boards have been programmed with the relevant binaries (as described in Section 3).

4.1 Start-up sequence

- Power up the ESI-EM
    - The device will automatically form a network
- Power up the Gas Meter and Range Extender
    - The nodes will join automatically and extinguish all LEDs once complete
- Power up the IPD and press the 'Join' button (SW1) to initiate the joining process
    - The IPD will switch to the Simple Metering display once the joining and meter discovery stages are complete

Note 1: If any of the nodes fail to join ensure that the network context data has been cleared, see Section 5.1 for more information.

Note 2: If the IPD joins the network before the Gas Meter it must re-discover all available meters before the new meter's data can be queried and displayed. To do this simply reset the IPD device.

4.2 Modifying a Meter's Simulated Load

Each of the metering device types (including the ESP-EM) simulate a real-time load of a fixed commodity type (electricity, gas and water). This real-time load is also known as Instantaneous Demand and the unit of measure is specified by the meter's commodity type (kWh, m^3 and L^3 respectively).

If the meter binary is programmed into a JN516x Carrier Board and Generic Expansion Board, the Instantaneous Demand attribute is updated with an ADC reading of the on-board variable potentiometer. This can therefore be adjusted on-the-fly to simulate a change in load.

4.3 Transferring Data with a Secured Tunnel

The Tunnelling cluster provides a secure transport mechanism for metering protocols (e.g. DLMS/COSEM, IEC61107, NSI C12, M-Bus) within the payload of standard ZigBee frames.

Pressing switch SW4 will trigger the tunnel creation and tunnelled data transfer. Data will be transferred between the ESI_EM and the EM nodes.

The NXP *ZigBee PRO Smart Energy User Guide (JN-UG-3059)* provides a more detailed description of the Tunnelling cluster.

 © NXP Laboratories UK 2013

4.4 User Interface

The IPD has 7 display screens:

- Simple Metering (default)
- Price
- DRLC
- Messaging
- Message Confirmation
- Historical Data
- Settings

The left and right arrow buttons (SW3 & SW4 respectively) cycle through the different screens. The remaining two action buttons (SW1 & SW2) perform a range of functionalities and are detailed in the sub-sections below.

4.4.1 Simple Metering

The simple metering screen is the IPD's default display and shows the following information:

- Time
- Instantaneous demand (numerically and graphically)
- Commodity type
- The current meter of interest (e.g. M1 is the 1st discovered meter)
- Battery Voltage
- Link Quality Indicator (LQI)

The 'Mode' button (SW1) cycles through displaying Instantaneous Demand, the estimated CO_2 equivalent and cost per hour on a button press. Alternatively holding the 'Mode' button for two seconds will force the IPD to switch to the next available meter (providing multiple meters have already been discovered in the HAN). The meter that is currently being displayed is shown on the right-hand side of the screen, e.g. M1, M2, etc.

Note: The meter's identifier (e.g. M1, M2, etc.) is assigned in the order in which the meters are discovered.

Pressing and releasing the 'GetP' button (SW2) will trigger the IPD to send an arbitrary 'Get Profile' command to the price server, to satisfy SEP certification requirements. See Section 5.6 for more information on SEP certification.

4.4.2 Price

The price screen displays the following information about any currently scheduled prices:

- The unit cost per hour
- Start time
- Duration

 © NXP Laboratories UK 2013

Note: The unit of currency is configurable via the Settings screen for demonstration purposes, not from a received price's commodity type.

Pressing and releasing the 'GetC' or 'GetS' buttons (SW1 and SW2 respectively) will initiate arbitrary 'Get Current Price' or 'Get Scheduled Price' commands to satisfy SEP certification requirements. See Section 5.6 for more information on SEP certification.

4.4.3 DRLC

The DRLC screen displays the following information about any received DRLC events:

- Event ID
- Start time
- Duration
- Event queue identifier (e.g. active, scheduled, expired, etc.)

Pressing and releasing the 'Opt' button (SW1) will trigger the IPD to opt out of the currently displayed DRLC event. To opt back in at a later time press and hold the 'Opt' button for two seconds.

Pressing and releasing the 'GetS' button (SW2) will cause the IPD to send out a 'Get Scheduled Event' command to the DRLC server.

4.4.4 Messaging

The message screen displays the text payload of any currently active message.

Pressing and releasing the 'GetM' button (SW1) triggers the IPD to send a 'Get Last Message Request' to the messaging server.

Pressing and releasing the 'Clear' button (SW2) clears any currently active message from the local message queue.

4.4.5 Message Confirmation

The IPD will switch automatically to the message confirmation screen if the ESP requests user confirmation of a message or a message cancelation.

4.4.6 Historical Data

The historical screen displays a graphical representation of which price tier energy has been consumed at. Historical data is maintained from when the IPD was last powered up and is not persisted.

4.4.7 Settings

The settings screen allows the following parameters to be customised in real time to aid demonstration purposes:

- Unit of currency
    - USD
    - EUR
    - GBP

 © NXP Laboratories UK 2013

- Time
    - 12 hour clock
    - 24 hour clock
- Fast poll mode
    - Triggers the IPD to send an arbitrary fast poll request command

Pressing and releasing the ‘Scroll’ button (SW1) scrolls through and highlights each of the items listed above.

Pressing and releasing the ‘Select’ button (SW2) actions/increments the current selection.

5 Advanced User Information

5.1 Saving Network Context

All device types are protected from losing their network configuration during a power failure by means of context saving. The required network parameters are automatically preserved in non-volatile memory by the ZigBee PRO stack. On restart, the radio channel, Extended PAN ID (EPID) and security keys are restored; a power failure is therefore transparent to the user.

Application-specific information can also be preserved in non-volatile memory and is most commonly used to preserve the application’s operating state.

During demonstration and development, it is often necessary to clear this context data. To clear context on any of the JN516x-EK001 Evaluation Kit carrier boards, hold down button ‘SW1’ whilst resetting the device. In order to clear context on the JN5168 USB Dongle the JN51xx Flash Programmer’s ‘Erase EEPROM’ functionality must be used due to the lack of physical buttons.

5.2 Multiple Meter Support

The IPD is currently configured to read metering data from up to two meters. To increase this further, increment the number of allocated endpoints in the ZPS configuration editor and zcl_options.h file (SE_NUMBER_OF_ENDPOINTS).

5.3 Security Keys and MAC Addresses

All nodes within an SE network are required to complete security handshaking with the network trust centre before certain communication links can be established – this process is known as Key Establishment (KE). In order to complete KE, each node must have previous knowledge of certain serialisation data, such as its MAC address and security key information.

For ease of use during development, this serialisation data has been hardcoded within the application – see Section 5.3.1 below. In a production environment, however, it is not feasible to have a custom binary for every device – for more information on production serialisation data, see Section 5.3.2.

5.3.1 Development

Each node’s serialisation data has been hardcoded within the application to aid development. The device’s original MAC address remains intact but is superseded by the application at runtime using the **ZPS_vSetOverrideLocalMacAddress** API. Table 2 below details each device type and its corresponding hardcoded MAC addresses.

 © NXP Laboratories UK 2013

Device Type	MAC Address
IPD_NODE	0x0000000000000001
ESP_METER_NODE	0x0000000000000002
PLUG_METER_NODE	0x0000000000000003
GAS_METER_NODE	0x0000000000000004
RANGE_EXT_NODE	0x0000000000000005

Table 2: Hardcoded MAC Addresses

Note: If you wish to introduce additional nodes into the network, further test certificates must be obtained from Certicom and added to the relevant node's **app_certificates.h** file.

5.3.2 Production

In production, device specific information known as serialisation data is embedded within the application binary (the same applies to an OTA image). For more information on how to combine an application binary with serialisation data refer to the *JET User Guide (JN-UG-3081)*.

The application is required to load the serialisation data at runtime from Flash memory. To enable this, modify each node's existing build command to the following:

```
CERTIFICATES=PRODUCTION_CERTS
```

5.4 Over-The-Air Upgrades

The Application Note has OTA enabled into the ESP-EM and IPD nodes by default. The relevant makefiles automatically run JET to modify the binaries post-build (for more information refer to the *JET User Guide (JN-UG-3081)*). The IPD makefile also generates an 'OTA upgrade' version of the same binary. For instructions on how to set up and initiate an OTA download see Sections 5.4.1 and 5.4.2 below.

5.4.1 OTA Download Configuration

The OTA download process requires **both** the OTA server and client to have external Flash memory for storage of the upgrade image. The JN516x-EK001 Evaluation Kit carrier boards all have external flash memory fitted but are not enabled by default. To enable use of the external Flash chip, remove the Expansion Board and set the 'SPI' header to the 'SSZ' option.

Note: When enabling the external Flash memory, pin 16 can no longer be used as general purpose I/O (DIO0).

The IPD upgrade image must be programmed directly into the external flash of the OTA server (ESP-EM). For instructions on how to program the carrier board's external Flash refer to the relevant Appendix in the *JN51xx Flash Programmer User Guide (JN-UG-3007)*.

5.4.2 OTA Download Initiation

Press ‘SW2’ on the ESP-EM, once the IPD has joined the network, to initiate the upgrade process. This will trigger the ESP-EM to notify the network that a new image is available for download. The IPD will then trigger the download automatically providing the binary is of the correct type.

> **Note:** For subsequent IPD upgrade binaries the ‘Application Version’ number can be set by modifying the ‘`OTA_UPGRADE_VERSION`’ parameter in the IPD’s makefile.

5.5 Radio Recalibration

Although not enabled by default, all nodes have radio recalibration code included to compensate for temperature variations during deployment. The radio recalibration occurs during start-up and either on wake-up or periodically every minute (sleeping and permanently powered devices respectively).

There is the potential to miss on-air packets whilst the periodic recalibration is taking place. This, however, should not present a significant problem as any critical data will require an APS acknowledgement. On failing to receive an acknowledgement, the source node will automatically resend the data 1.6 seconds later.

In order to enable radio recalibration, uncomment the following lines of code, rebuild the application and program the relevant nodes:

In each node’s makefile, uncomment the following library inclusion:

```
#APPLIBS += Recal
```

In each node’s main header file (e.g. **app_meter_node.h**), uncomment the following definition:

```
//#define RADIO_RECALIBRATION
```

5.6 ZigBee Alliance SEP Certification

Additional functionality has been added to the ESP-EM and IPD nodes to aid SEP1.x certification. This functionality can be enabled by uncommenting the following line in the relevant node’s makefile:

```
#SE_CERTIFICATION
```

5.6.1 IPD Certification Functionality

The IPD has been modified to allow the device to pause once it has joined at the networking layer, thus allowing various Key Establishment tests to be performed. To initiate this functionality, press and hold ‘SW1’ for five seconds when attempting to join the network. Pressing and releasing ‘SW1’ will then trigger the device to continue starting up as normal.

5.6.2 ESP-EM Certification Functionality

The ESP-EM device has been modified to allow the transmission of hardcoded packets that are required for certain SEP test clauses, e.g. prices, DRLC events, messages, etc. The remaining two buttons on the Generic Expansion Board are used to navigate the certification state machine. ‘SW3’ increments the certification state (i.e. steps through the items in each test clause) and ‘SW4’ executes the functionality (e.g. transmits the relevant packet). The

 © NXP Laboratories UK 2013

currently supported test clauses for SE1.1b are shown in the ESP-EM's **app_certification.c** and **app_certification.h** files.

6 Developing with the Application Note

This section provides additional information that may be useful when developing with this Application Note.

Before commencing development of a Smart Energy project, you are recommended to familiarise yourself with the following documents:

[R1] - NXP ZigBee PRO Stack User Guide [JN-UG-3048]
[R2] - NXP JenOS User Guide [JN-UG-3075]
[R3] - NXP ZigBee PRO Smart Energy User Guide [JN-UG-3059]
[R4] - NXP ZigBee Cluster Library User Guide [JN-UG-3077]
[R5] - NXP JN516x Integrated Peripherals API User Guide [JN-UG-3087]
[R6] - ZigBee Smart Energy Profile Specification
[R7] - ZigBee Cluster Library Specification (ZCL)

The latest versions of [R1] to [R5] can be obtained from the NXP Wireless Connectivity TechZone, while [R6] and [R7] can be found on the ZigBee Alliance web site (www.zigbee.org).

6.1 Debugging the Application

6.1.1 Serial Debug

Each node in the Application Note prints out debug information by default via the UART. This debug information can viewed by terminal emulator software, e.g. Tera Term. Connect the node of interest to a laptop using the supplied Mini-USB cable and configure the terminal emulator's COM port as follows:

BAUD	115200
Data	8 bit
Parity	None
Stop bit	1 bit
Flow control	None

Debug can be disabled for production by setting the 'Trace' flag in the relevant node's makefile to zero. The makefile also defines a subset of debug flags that allows localised debug statements to be enabled or disabled on mass, e.g. TRACE_START.

6.1.2 On-Air Packets

The demo uses the following pre-configured link key and channel mask should you wish to capture on-air data packets with a protocol analyser (such as Ubilogix Ubiqua):

Pre-configured Link Key	0xFF11223344556677889 9AABBCCDDEE00
Channel Mask	11, 14, 15, 19, 20, 24 and 25

6.2 Building and Downloading the Application

This section provides application build instructions, if you simply wish to use the ready built application binaries supplied in the zip, see Section 3.

The software provided with this Application Note is designed for use with JN516x microcontrollers. JN516x applications can be built using the Eclipse IDE or makefiles.

In order to build the supplied software, the application's folder must be placed in the **Application** folder of the NXP/Jennic SDK installation:

<JN516x_SDK_ROOT>\Application

where **<JN516x_SDK_ROOT>** is the path into which the SDK was installed (by default, this is **C:\Jennic**). The **Application** directory is automatically created when you install the SDK.

To build the applications and load them into JN516x-based boards, follow the instructions below:

1. Ensure that the project directory is located in

<JN516x_SDK_ROOT>\Application

where **<JN516x_SDK_ROOT>** is the path into which the SDK was installed.

2. Start the Eclipse platform and import the relevant project files (**.project** and **.cproject**) as follows:

a) In Eclipse, follow the menu path **File>Import** to display the **Import** dialogue box.

b) In the dialogue box, expand **General**, select **Existing Projects into Workspace** and click **Next**.

c) Enable **Select root directory**, browse to the **Application** directory and click **OK**.

d) In the **Projects** box, select the project to be imported and click **Finish**.

3. Build an application. To do this, ensure that the project is highlighted in the left panel of Eclipse and use the drop-down list associated with the hammer icon  in the Eclipse toolbar to select the relevant build configuration – once selected, the application will automatically build. Repeat this to build the other node types.

The binary files will be created in the relevant **Build** directories, the resulting filenames indicating the chip type (e.g. **JN5168**) for which they were built.

Note 1: When importing a ZigBee PRO application for the first time you may need to perform a clean build. To do this select the relevant build configuration as described above, right-click on the project and select 'Clean Project'

Note 2: For backward compatibility with the JN5148 evaluation kit (JN5148-EK010) add the following to each node's build configuration or makefile: `JENNIC_PCB=DEVKIT2`

4. Load the resulting binary files into the boards. You can do this using the JN51xx Flash Programmer, which can be launched from within Eclipse or used directly (and is described in the *JN51xx Flash Programmer User Guide (JN-UG-3007)*).

 © NXP Laboratories UK 2013

6.3 Application Start-up

This section describes the typical start-up flow of an NXP ZigBee PRO device. Note that not all devices sleep, hence the 'Warm Start' path is not always applicable.

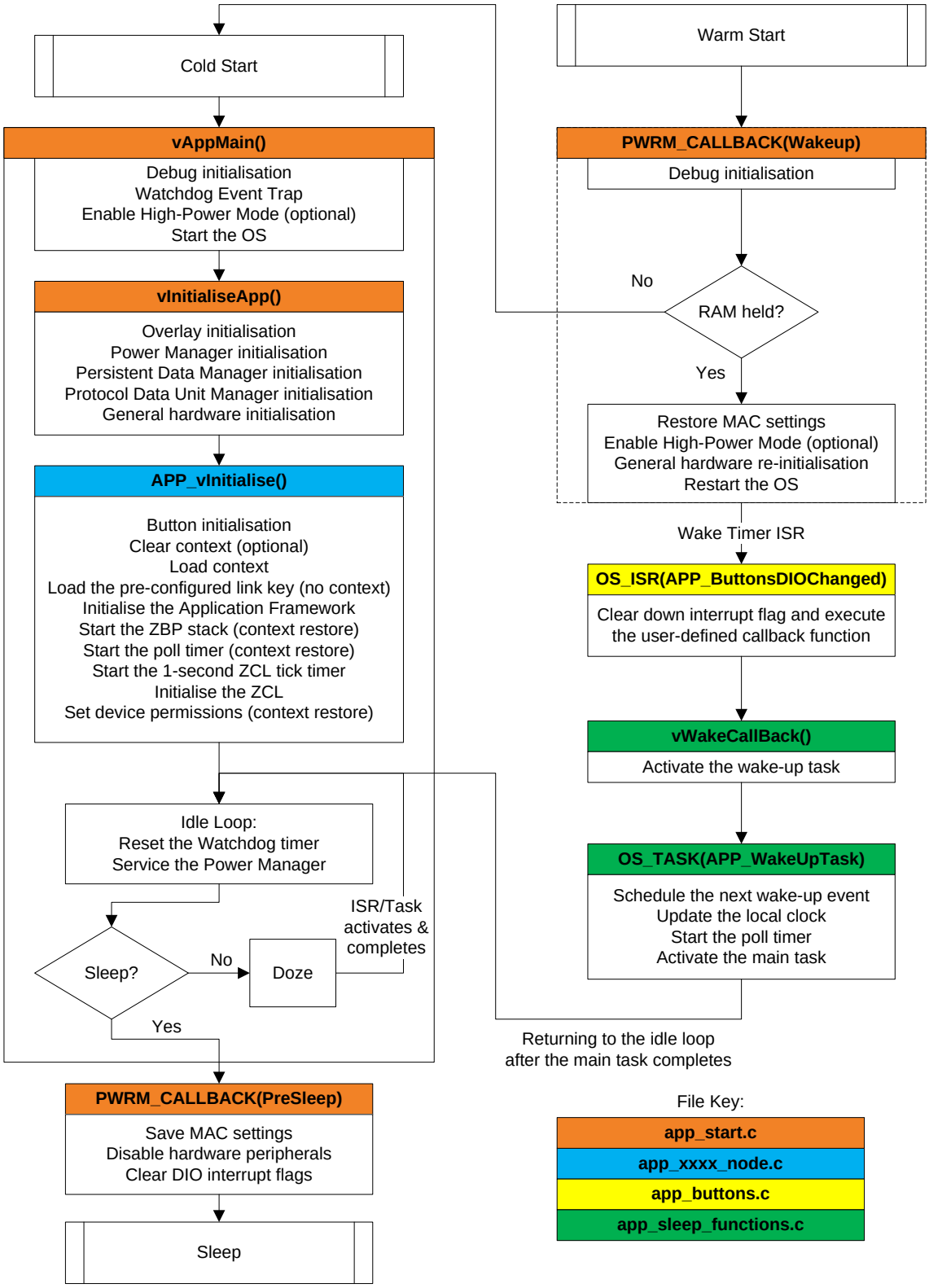

Figure 2: Typical Start-up Flow

 © NXP Laboratories UK 2013

6.4 Advanced IPD Information

6.4.1 The Discovery Process

After joining the HAN, the IPD sends out a match descriptor request to discover the Key Establishment server and, once located, attempts to complete Key Establishment. If successful, the node then attempts to discover and bind to any other relevant cluster servers. This entire process is detailed in Figure 3 below.

Figure 3: IPD Main Task Flow Diagram

6.4.2 A Typical Wake-up Cycle

After the IPD enters the normal running state, it reads the time from the ESP, followed by any active price and messaging information. The device then reads metering information from the currently selected meter.

Once the metering information is received, the screen is updated with the instantaneous demand value. The device then enters sleep mode, waking up every 7 seconds to request instantaneous demand. For more information on a typical wake-up cycle, refer to Figure 4.

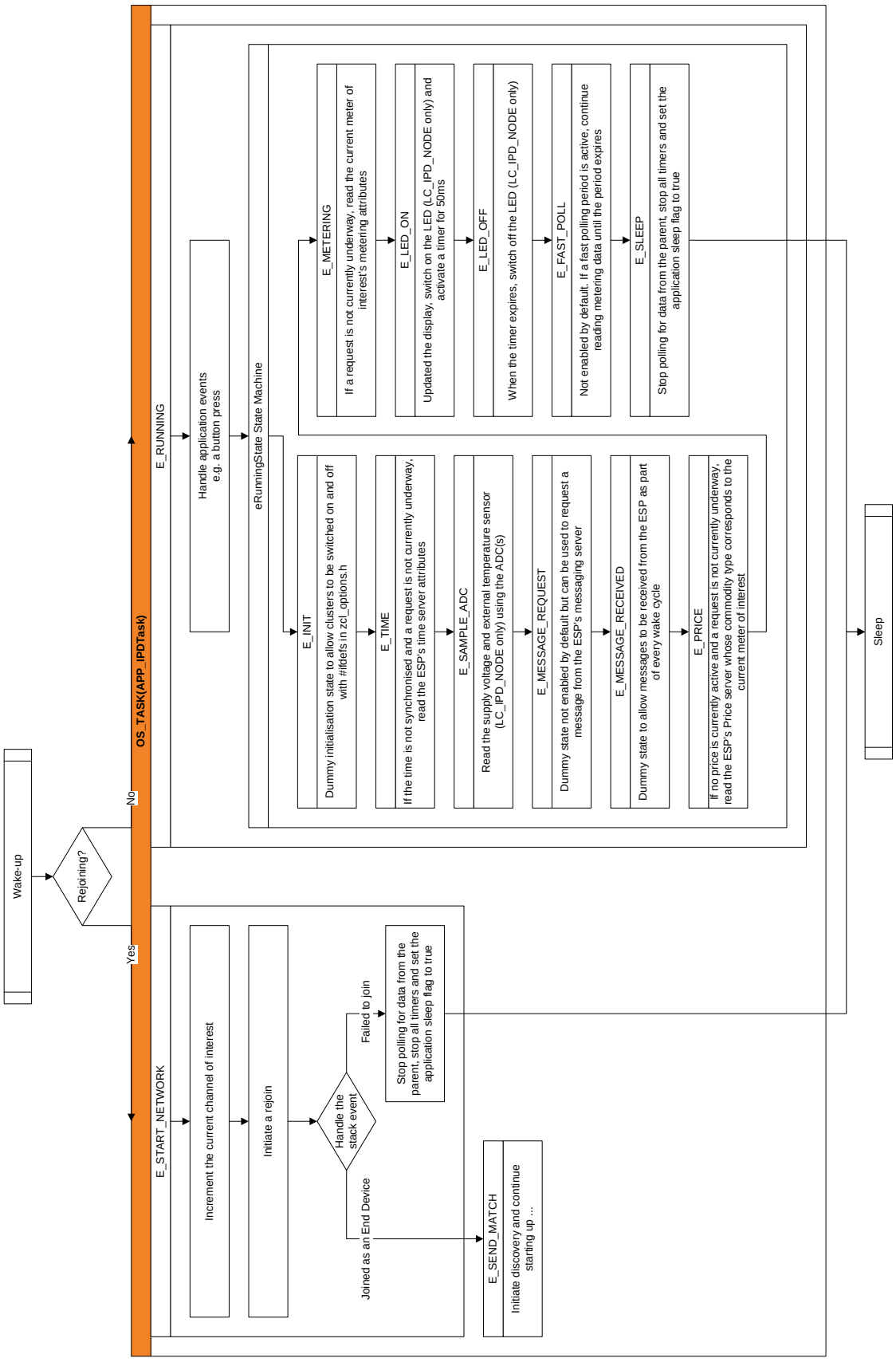

Figure 4: Typical Wake-up Cycle

6.4.3 Guidelines for Modifying the IPD

This section highlights the key areas of interest within the code, in case the developer wishes to alter the IPD's functional states or switch to a different user interface.

6.4.3.1 Functionality

6.4.3.1.1 Operational State Machine

The operational state machine (**s_sDevice.eState**) is located within **app_ipd_node.c**. Additional states must be added to this switch statement if further operational modes are required.

6.4.3.1.2 Display State Machine

The display state machine (**app_eDisplayState**) is located in **app_display_x.c**. Changes to the display states should be made within the corresponding switch statement. The **app_display_x.c** file also contains the display configuration functions.

6.4.3.2 User Interface

6.4.3.2.1 Keypad

The button mask is located in **app_buttons.c**. If required, additional buttons should be added to the mask at the top of the file. An interrupt is generated if any of the buttons within this mask are actioned. The Interrupt Service Routine will then generate a button event which can be acted on in any of the operational states (**app_event_handler.c**).

 © NXP Laboratories UK 2013

Appendix A - Source File Descriptions

Automatically Generated Files

Every node has several files that are automatically generated at build-time by the RTOS and ZPS Configuration Diagrams. These files are not generally used by the developer but they are located in the respective node's **\Source\Plugins** folders, should they be of interest.

Common Files

A number of common files are used across all node types and are located within the **\Common\Source** folder. This section gives a brief description of each of the common files.

Filename	Description
app_buttons.c/h	Contains the System Controller Interrupt Service Routine which handles the button and wake timer interrupts. Also defines the button mask and assigns the relevant DIO.
app_smartenergy_demo.h	Contains global definitions for the SE HAN application.
app_zbp_utilities.c/h	Contains commonly used debug functions.
os_msg_types.h	Used to pass in application-specific 'includes' into the automatically generated files of the RTOS and ZPS Configuration diagrams.
StackMeasure.c/h	Used to track the maximum CPU stack usage for optimisation purposes.
app.zpscfg	This is the ZigBee PRO Stack Configuration Editor, which is used to configure generic network and node parameters. This includes profile, cluster, endpoint and RF channel configurations. For more information, refer to the "ZPS Configuration Editor" chapter of the *ZigBee PRO Stack User Guide (JN-UG-3048)*.

ESP-EM Source Files

This section gives a brief description of the combined ESP and EM node's source files, located in the **\ESP_METER_NODE\Source** folder.

Filename	Description
app_certificates.h	Production and development certificates, public key, private key and pre-configured link key information. For more information, refer to the "ZigBee SE Security" section of the *ZigBee PRO Smart Energy API User Guide (JN-UG-3059)*.
app_certification.c/h	Contains the certification state machine, used to send out hardcoded packets in accordance with the various SEP test clauses.
app_event_hander.c/h	Contains the main running state and, with it, the main stack and application event handler.
app_meter_node.c/h	Contains the main application task, the initialisation function and application state machine, including the network formation functions.
app_start.c	Contains the main function that is executed on device start-up (cold start). For more information on a typical application start-up flow, see Section 6.1.
app_zcl_task.c/h	Contains the ZigBee Cluster Library task which maintains the on-board RTC clock and handles events based on incoming data packets, such as a 'read attribute request'.
zcl_options.h	Used to customise which clusters and optional attributes the application uses.
App_ESP_METER_NODE_JN51xx.oscfgdiag	This is the Real Time Operating System (RTOS) configuration diagram, which is used to configure certain application building blocks, such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

Generic IPD Source Files

This section gives a brief description of the IPD node's source files, located in the **\IPD_NODE\Source** folder.

Filename	Description
app_adc.h	The header file for the ADC source files
app_certificates.h	Production and development certificates, public key, private key and pre-configured link key information. For more information, refer to the "ZigBee SE Security" section of the *ZigBee PRO Smart Energy API User Guide (JN-UG-3059)*.
app_display.h	The header file for the display source file, and includes defining the display states and LC build's character map.
app_event_hander.c/h	Contains the main running state and, with it, the main stack and application event handler.
app_ipd_node.c/h	Contains the main application task, the initialisation function and application state machine, including the network joining functions. This file also includes discovery (match descriptor requests) and binding to other nodes/endpoints of interest.
app_led.h	The header file for the LED driver source files.
app_sleep_functions.c/h	Contains the wake-up and poll tasks, as well as any associated sleep and wake-up functions.
app_start.c	Contains the main function that is executed on device start-up (cold start), as well as the pre-sleep and post-sleep callback functions. For more information on a typical application start-up flow, see Section 6.1.
app_zcl_task.c/h	Contains the ZigBee Cluster Library task which maintains the on-board RTC clock and handles events based on incoming data packets, such as a 'read attribute request'.
zcl_options.h	Used to customise which clusters and optional attributes the application uses.
App_IPD_NODE_JN51xx.oscfgdiag	The RTOS configuration diagram is used to configure application building blocks such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

IPD_NODE_EVK Source Files

This section gives a brief description of the Evaluation Kit specific source files, located in the **\IPD_NODE\Source\EVK** folder.

Filename	Description
app_adc_evk.c	The ADC source file used to read the supply voltage of the device.
app_display_evk.c	The main source code for generating the display, specific to the EVK build.
app_led_evk.c	Fall through functions to allow the LC code to be built from the same common file base. The EVK build does not actually have any LED functionality.
Symbol.h	Contains the LCD symbol maps for the display.
xsprintf.c/h	Used to format LCD data.

IPD_NODE_LC Source Files

This section gives a brief description of the Low Cost Reference Design specific source files, located in the **\IPD_NODE\Source\LC** folder.

Filename	Description
app_adc_lc.c	The ADC source file used to read the supply voltage and temperature sensor on the device.
app_display_lc.c	The main source code for generating the display, specific to the LC build.
app_lcd_driver.c/h	The source code for communicating with the I^2C LCD driver.
app_led_lc.c	The LED driver source code.

 © NXP Laboratories UK 2013

Standalone Meter Source Files

This section gives a brief description of the stand alone meter nodes' source files, located in the **\GAS_METER_NODE\Source** and **\PLUG_METER_NODE\Source** folders.

Filename	Description
app_certificates.h	Production and development certificates, public key, private key and pre-configured link key information. For more information, refer to the "ZigBee SE Security" section of the *ZigBee PRO Smart Energy API User Guide (JN-UG-3059)*.
app_event_hander.c/h	Contains the main running state and, with it, the main stack and application event handler.
app_meter_node.c/h	Contains the main application task, the initialisation function and application state machine, including the network joining functions.
app_start.c	Contains the main function that is executed on device start-up (cold start). For more information on a typical application start-up flow, see Section 6.1.
app_zcl_task.c/h	Contains the ZigBee Cluster Library task which maintains the on-board RTC clock and handles events based on incoming data packets, such as a 'read attribute request'.
zcl_options.h	Used to customise which clusters and optional attributes the application uses.
App_x_METER_NODE_JN51xx.oscfgdiag	The RTOS configuration diagram is used to configure application building blocks such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

Range Extender Source Files

This section gives a brief description of the Range Extender node's source files, located in the **\RANGE_EXT_NODE\Source** folder.

Filename	Description
app_certificates.h	Production and development certificates, public key, private key and pre-configured link key information. For more information, refer to the "ZigBee SE Security" section of the *ZigBee PRO Smart Energy API User Guide (JN-UG-3059)*.
app_event_hander.c/h	Contains the main running state and, with it, the main stack and application event handler.
app_range_ext_node.c/h	Contains the main application task, the initialisation function and application state machine, including the network joining functions.
app_start.c	Contains the main function that is executed on device start-up (cold start). For more information on a typical application start-up flow, see Section 6.1.
app_zcl_task.c/h	Contains the ZigBee Cluster Library task which maintains the on-board RTC clock and handles events based on incoming data packets, such as a 'read attribute request'.
zcl_options.h	Used to customise which clusters and optional attributes the application uses.
App_RANGE_EXT_NODE_JN51xx.oscfgdiag	The RTOS configuration diagram is used to configure application building blocks such as pre-emptive tasks, software timers, mutexes and Interrupt Service Routines. For more information, refer to the *JenOS User Guide (JN-UG-3075)*.

Appendix B - Build File Descriptions

Common Build Files

This section gives a brief description of the common build files, located in each node's **\Build** folder.

Filename	Description
App_Stack_Size.ld	Over-rides the default CPU stack size.
Makefile	The main makefile used to build the binary for this node.

IPD Specific Build Files

This section gives a brief description of the IPD node's build files, located in the **\IPD_NODE\Build** folder.

Filename	Description
App_Overlay_Discovery.ld	Used to create an application overlay for the match descriptor and binding functions (only included in JN514x applications).
App_Overlay_Display.ld	Used to create an application overlay for the display and display driver functions (only included in JN514x applications).
App_Overlay_Joining.ld	Used to create an application overlay for the joining functions (only included in JN514x applications).

Appendix C - Known Issues

ID	Severity	Description
lpap145	Minor	In certain circumstances, the IPD continually attempts to synchronise with the ESP's Price server on first-time start-up. The application automatically recovers within two minutes.

Appendix D - Application Code Sizes

Node	RAM Usage (kB)	Flash Usage (kB)
ESP_METER_NODE	28.4	194.2
IPD_NODE_EVK	27.3	207.6
GAS_METER_NODE	20.3	170.8
PLUG_METER_NODE	20.4	172.0
RANGE_EXT_NODE	19.7	160.0

 © NXP Laboratories UK 2013

Revision History

Version	Notes
1.0-1.2	Releases with earlier titles
2.0	Revised version under the title “Smart Energy HAN Solutions”
3.0	Application Note expanded to include a Range Extender, standalone Meter and multiple IPD build configurations.
3.1	Added SDK update for JN5148-Z01 chip variant
4.0	Updated for the JN516x chip family and evaluation kit
4.1	Added references to the Tunnelling cluster
4.2	Updated to support new LCD panel on LCD Expansion Board

Important Notice

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the *Terms and conditions of commercial sale* of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

All trademarks are the property of their respective owners.

NXP Laboratories UK Ltd
(Formerly Jennic Ltd)
Furnival Street
Sheffield
S1 4QT
United Kingdom

Tel: +44 (0)114 281 2655
Fax: +44 (0)114 281 2951

www.nxp.com