



JenNet Home Sensor Demonstration Application Note

JN-AN-1065

Revision 3.0

24-Aug-2010

Contents

About this Manual	5
Organisation	5
Conventions	5
Acronyms and Abbreviations	5
Related Documents	6
Feedback Address	6
1 Operational Features	7
1.1 Functional Overview	7
1.2 Hardware Components	7
1.3 Network Formation	8
1.4 Sensor Monitoring	8
1.4.1 Network Screens	10
1.4.2 Node Screens	10
1.4.3 Node Control Screens	11
1.4.4 Settings Screen	12
1.5 Remote Control of LEDs	13
2 Application Design	15
2.1 Software Architecture	15
2.2 Context, Interrupts and Callbacks	15
3 Unpacking, Build and Download Instructions	17
3.1 Unpacking the Application	17
3.2 Building the Application	17
3.2.1 Using Eclipse (JN5148 Only)	18
3.2.2 Using Code::Blocks (JN5139 Only)	18
3.2.3 Using Makefiles	19
3.3 Downloading the Application	20
4 Code Description	21
4.1 Common Function Calls	21
4.1.1 Overview	21
4.1.2 Function Descriptions	21
4.2 Co-ordinator	23
4.2.1 Overview	23
4.2.2 Function Descriptions	24
4.3 Router	26
4.3.1 Overview	26
4.3.2 Function Descriptions	26
4.4 End Device	27
4.4.1 Overview	27
4.4.2 Function Descriptions	27

About this Manual

This document forms part of the Application Note *JenNet Home Sensor Demonstration (JN-AN-1065)* which contains the source code (and associated files) of the JenNet Home Sensor Demonstration. The document describes the demonstration application, which can be used as an example for your own JenNet application development.

This binary code for this example application is also supplied as the default Flash memory contents in the JN5148-EK010 Evaluation Kit. Use of this pre-loaded demonstration with the evaluation kit is fully described in the *JN5148-EK010 Evaluation Kit User Guide (JN-UG-3062)*.

Organisation

This document consists of four chapters, as follows:

- [Chapter 1](#) provides a high-level view of the application in terms of its hardware components and operation.
- [Chapter 2](#) outlines the application design in terms of the software used, and the use of callbacks and interrupts.
- [Chapter 3](#) describes how to build the application and download it to the hardware.
- [Chapter 4](#) describes the application code, including the functions used.

Conventions

Files, folders, functions and parameter types are represented in **bold** type.

Function parameters are represented in *italics* type.

Code fragments are represented in the `Courier` typeface.

Acronyms and Abbreviations

API	Application Programming Interface
IDE	Integrated Development Environment
SDK	Software Developer's Kit

Related Documents

JN-UG-3062	JN5148-EK010 Evaluation Kit User Guide
JN-UG-3041	JenNet Stack User Guide
JN-UG-3063	Eclipse IDE User Guide
JN-UG-3028	Code::Blocks IDE User Guide
JN-UG-3007	JN51xx Flash Programmer Application User Guide

Feedback Address

If you wish to comment on this manual, or any other Jennic user documentation, please provide your feedback by writing to us (quoting the manual reference number and version) at the following postal address or e-mail address:

Applications
Jennic
Furnival Street
Sheffield S1 4QT
United Kingdom

doc@jennic.com

1 Operational Features

The JenNet Home Sensor Demonstration application is intended as an aid to understanding how an application can be built on top of the JenNet stack, and how to use the sensors and LCD panel of an evaluation kit for the JN5148 or JN5139 device.



Note: This software is supplied as the pre-loaded demonstration of the JN5148-EK010 Evaluation Kit. Use of this demonstration with the evaluation kit is fully described in the *JN5148-EK010 Evaluation Kit User Guide (JN-UG-3062)*.

1.1 Functional Overview

The demonstration uses a Controller board (featuring an LCD panel) and a number of Sensor boards from the evaluation kit. All the boards measure temperature, humidity and light levels. The Sensor boards periodically send their measurements to the Controller board through a beacon mechanism. The Controller board displays the received data on its built-in LCD panel. Thus, the Controller board allows each Sensor board to be monitored, and also allows alarms to be set for temperature and light levels on the Sensor boards. In addition, LEDs on the Controller board can be controlled using buttons on the Sensor boards.

In this application, the Controller board acts as the Co-ordinator of a JenNet network. The Sensor boards act as Routers (boards with SMA connectors) and as End Devices (boards with ceramic antennae). Separate code is provided for the Co-ordinator, Routers and End Devices.

The Controller board can be configured to operate on a specific channel to avoid interference on busy frequencies, and the Sensor boards automatically scan for the controller and synchronise with it.

1.2 Hardware Components

The evaluation kit consists of:

- One Controller board with an LCD panel, a temperature sensor, humidity sensor, light sensor, four control buttons (SW1-SW4) and four LEDs (D1-D4)
- Four Sensor boards, each with a temperature sensor, humidity sensor, light sensor, two control buttons (SW1 and SW2) and two LEDs (D1 and D2)

The Controller board and two of the Sensor boards have SMA connectors for antennae. The two remaining Sensor boards have integrated ceramic antennae.

For more detailed information on the Controller board and Sensor boards of the JN5148-EK010 Evaluation Kit, refer to the *JN5148-EK010 Evaluation Kit User Guide (JN-UG-3062)*.

1.3 Network Formation

The Controller board should be powered on first. The start-up screen on the LCD panel allows the radio channel to be selected using the '+' and '-' buttons (default is channel 18) and the network to be started by pressing the 'Done' button.

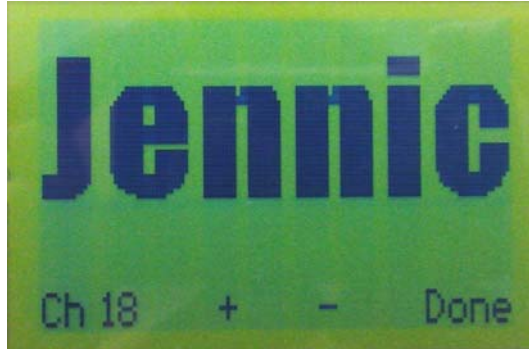


Figure 1: Start-up Screen

The Sensor boards can then be powered on. When a Sensor board is switched on, it automatically enters scanning mode:

1. First the Sensor board performs a channel scan. It repeatedly tries until it finds the channel on which the network is operating.
2. The board then associates with a suitable parent that is operating on this network. Again, this will be repeated until successful. During initial association, the Sensor board is assigned a role by the controller. The roles are given in the order in which each board associates, in the order Hall, Bedroom, Lounge and Bathroom.
3. After a first successful association, the Controller board remembers the role it assigned to a particular Sensor board, so if that Sensor board loses the communication link and has to re-associate, it will be assigned the same role again.
4. Once associated, the Sensor board moves into operating mode.

While in operating mode, the Sensor board periodically transmits its sensor values to the Controller board, which can then display this information. The Controller board can also display its own sensor values.

1.4 Sensor Monitoring

The application which runs on the Controller board provides a user interface consisting of a set of information screens that display the sensor data from the Sensor boards in different ways. The following types of screen are available:

- Network screens, described in Section 1.4.1
- Node screens, described in Section 1.4.2
- Node Control screens, described in Section 1.4.3
- Settings screen, described in Section 1.4.4

JenNet Home Sensor Demonstration Application Note

Screen navigation is performed on the Controller board using the four buttons SW1-SW4, which are given on-screen labels. This navigation is illustrated in the figure below.

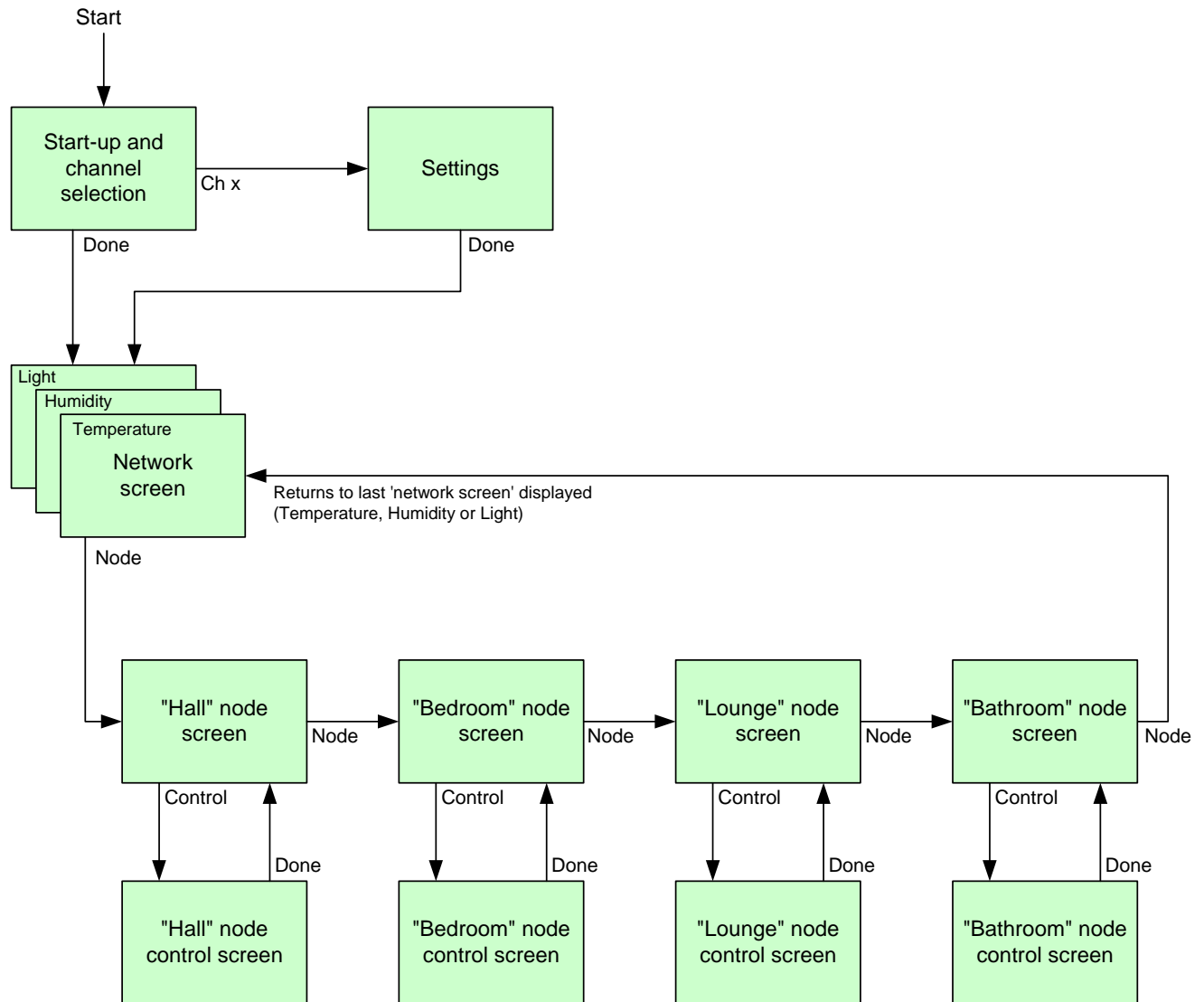


Figure 2: Screen Navigation

1.4.1 Network Screens

Each network screen displays a particular type of sensor data for all network nodes:

- **Temperature screen:** This screen displays the temperature readings from all nodes, in degrees Celsius. It is the first screen displayed following the start-up screen (once the 'Done' button has been pressed).
- **Humidity screen:** This screen displays the relative humidity readings from all nodes, as percentages.
- **Light screen:** This screen displays the light-level readings from all nodes. A light-level reading is represented pictorially by a circle which is partially filled according to the light level - a completely filled circle indicates total darkness and a completely unfilled circle indicates maximum brightness.

Each sensor reading is also displayed as a graph as well as a value/picture.

A network screen also displays the number of messages from each node that have been missed - this count will increase rapidly if the network has lost the node. Note that this figure is not shown for the Controller board since its sensor measurements are not transmitted over the air.

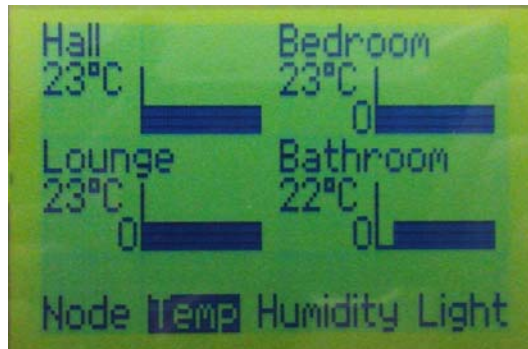


Figure 3: Example Network Screen (Temperature)

You can move directly from one network screen to another network screen by pressing the appropriate button on the Controller board (Temp, Humidity or Light). You can also move to the first 'node screen' (see Section 1.4.2) by pressing the 'Node' button.

1.4.2 Node Screens

Each node screen displays all the sensor readings from a particular node (Hall, Bedroom, Lounge or Bathroom). The node screens are accessed by pressing the 'Node' button on any of the network screens. This takes you to the first node screen, which is 'Hall'. Pressing the 'Node' button repeatedly then takes you through the node screens in sequence: first to the 'Bedroom' screen, then to the 'Lounge' screen, followed by the 'Bathroom' screen and finally back to the network screen where you started.

The node screens also display (below the room name) the number of messages from the relevant node that have been missed.

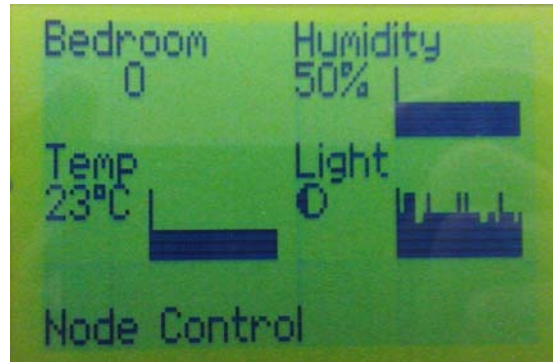


Figure 4: Example Node Screen (Bedroom)

1.4.3 Node Control Screens

There is a node control screen for each node. This screen is accessed by pressing the 'Control' button on the corresponding node screen. The node control screen is used to configure the alarms that can be generated for the node.

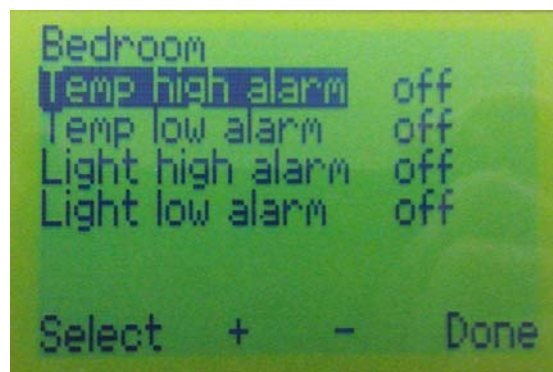


Figure 5: Example Node Control Screen (Bedroom)

Four alarms can be configured:

- **Temp high alarm:** This is triggered when the temperature measured by the node rises to a pre-defined value. This alarm is enabled by setting the required trigger temperature on this screen.
- **Temp low alarm:** This is triggered when the temperature measured by the node falls to a pre-defined value. This alarm is enabled by setting the required trigger temperature on this screen.
- **Light high alarm:** This is triggered when the light level measured by the node increases to a pre-defined level. This alarm is enabled by setting the required trigger level on this screen.
- **Light low alarm:** This is triggered when the light level measured by the node decreases to a pre-defined level. This alarm is enabled by setting the required trigger level on this screen.

By default, all the alarms are set to 'off'.

The alarm to be configured is selected using the 'Select' button and the trigger level for the alarm is selected using the '+' and '-' buttons.

The alarm settings are saved and the control screen is exited by pressing the 'Done' button, which returns you to the node screen where you started.

If an alarm has been enabled and is triggered, the corresponding text (Temp or Light) is inverted (dark background) on the relevant node screen.

1.4.4 Settings Screen

The settings screen can be accessed only from the start-up screen (before the network is formed) by pressing the 'Ch x' button (where x=18 be default).

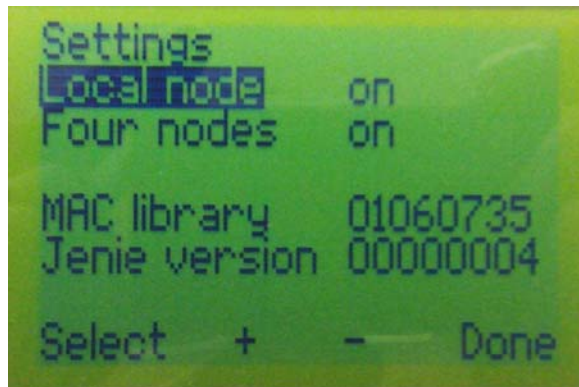


Figure 6: Settings Screen

This screen provides access to two configuration parameters, 'Local node' and 'Four nodes', as well as indicating software version information. The parameters are described in the table below.

Parameter	Possible Values	Description
Local node	off, on (default)	Allows the Controller board sensor readings to be included in or excluded from the sensor readings displayed: <ul style="list-style-type: none">• on: Include Controller board readings• off: Exclude Controller board readings
Four nodes	off, on (default)	Allows the sensor readings from 3 or 4 nodes to be displayed: <ul style="list-style-type: none">• on: Readings from 4 nodes displayed• off: Readings from 3 nodes displayed

The parameter to be configured is selected using the 'Select' button and the parameter setting is toggled using the '+' or '-' button. Pressing the 'Done' button will start the network with the configured settings (the next screen is the 'Temperature' screen).

This screen also shows the version numbers of the 802.15.4 Stack API and Jenie API used in the application.

1.5 Remote Control of LEDs

The Home Sensor Demonstration illustrates wireless lighting control by using a button on a Sensor board to control an LED on the Controller board, thereby providing a wireless lightswitch.

Buttons SW1 and SW2 on any Sensor board can be used to control one of the four LEDs on the Controller board. The LEDs controlled from the different Sensor boards are detailed in the table below.

Sensor Board	LED on Controller Board
Hall	D1
Bedroom	D2
Lounge	D3
Bathroom	D4

The buttons SW1 and SW2 on the Sensor boards are used as follows:

- Pressing SW2 illuminates the relevant LED on the Controller board.
- Pressing SW1 extinguishes the relevant LED on the Controller board.

There is no visible effect on the Sensor board itself.

2 Application Design

This chapter describes architectural and certain operational aspects of the demonstration application.

2.1 Software Architecture

The JenNet software stack is illustrated in the figure below. The application interacts with JenNet by means of the Jenie API.

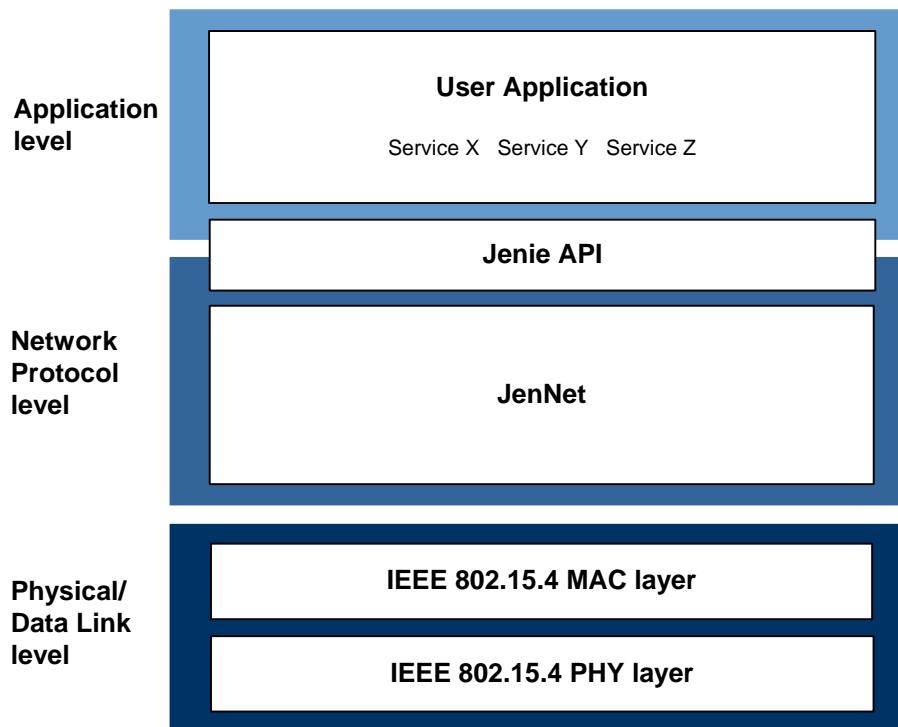


Figure 7: JenNet Software Stack

The JenNet stack is provided with defined entry points to request actions and also expects specific entry points to be provided by the application for callbacks.

The Integrated Peripherals API and Board API (from the Software Developer's Kit) sit logically to the side of the JenNet stack and are independent of it.

2.2 Context, Interrupts and Callbacks

The JenNet stack and the application operate within the processor's application context. Callbacks due to hardware events are passed to the application in the processor's interrupt context. It is therefore vital that code to implement the hardware event callback is kept as small and as efficient as possible.

3 Unpacking, Build and Download Instructions

This chapter describes how to build and download the demonstration application.



Note: This Home Sensor Demonstration application is pre-programmed into Flash memory of the boards provided with the JN5148-EK010 Evaluation Kit. You can therefore run the demonstration on this kit without needing to build and download it to the boards.

3.1 Unpacking the Application

You must first unzip the JN-AN-1065 Application Note into the following directory:

- For JN5148: `<JN51xx_SDK_ROOT>\Application`
- For JN5139: `<JN51xx_SDK_ROOT>\cygwin\jennic\SDK\Application`

where `<JN51xx_SDK_ROOT>` is the path into which the JN5148 or JN5139 Software Developer's Kit (SDK) was installed (by default, this is `C:\Jennic`). The **Application** directory is automatically created when you install the SDK.

3.2 Building the Application

The software provided with the Application Note can be built for the JN5148 or JN5139 device, but the available build methods differ between these chip types:

- JN5148 applications can be built using the Eclipse IDE or makefiles
- JN5139 applications can be built using the Code::Blocks IDE or makefiles

Note that different makefiles are provided for JN5148 and JN5139.

In all cases, you must build the separate applications for the Co-ordinator, Routers and End Devices (**AN1065_JN_HomeSensorCoord.c**, **AN1065_JN_HomeSensorRtr.c** and **AN1065_JN_HomeSensorEndD.c**) to produce the binary files to be downloaded to the relevant network nodes.

All files are located in the directory

...\Application\JN-AN-1065-JenNet-Home-Sensor-Demo

The source files are located in a **Source** sub-directory for each application.

If you are building the application for a high-power module, you must first ensure that the following is set in the header file **HomeSensorConfig.h**:

```
#define HIGH_POWER TRUE
```

Alternatively, this option can be set via the makefile.

This setting should not be applied when building for a standard-power module.

Build the application as described in the appropriate section below, depending on whether you intend to use Eclipse, Code::Blocks or makefiles.


3.2.1 Using Eclipse (JN5148 Only)

To build the applications and load them into JN5148 boards, follow the instructions below:

1. Ensure that the project directory is located in

<JN5148_SDK_ROOT>\Application

where **<JN5148_SDK_ROOT>** is the path into which the JN5148 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 applications.

The binary files will be created in the relevant **Build** directories, the resulting filenames indicating both the chip type (**JN5148**) and networking stack (**JN** for JenNet) for which they were built.

3.2.2 Using Code::Blocks (JN5139 Only)

To build each application and load it into a JN5139 board, follow the instructions below:

1. Ensure that the project directory is located in

<JN5139_SDK_ROOT>\cygwin\jennic\SDK\Application

where **<JN5139_SDK_ROOT>** is the path into which the JN5139 SDK was installed.

2. Open the appropriate Code::Blocks project file (**.cbp** file in the **CodeBlocksProject** directory) and build.

The project files are named according to both the chip type (**JN5139**) and networking stack (**JN** for JenNet) for which the binaries are to be built.

The binary file will be created in the **5139_Build** directory.



Caution: *If problems occur when using your current Code::Blocks version, download the latest version along with the latest version of this Application Note from the Support area of the Jennic web site (www.jennic.com/support).*

3.2.3 Using Makefiles

Each application has its own **Build** directory, which contains the makefiles for the application.

Different makefiles are provided for JN5148 and JN5139 – a JN5148 makefile is simply called **Makefile** while a JN5139 makefile is called **Makefile_JN5139.mk**.

To build each application and load it into a JN5148/JN5139 board, follow the instructions below:

1. Ensure that the project directory is located in

<JN51xx_SDK_ROOT>\Application for JN5148

<JN51xx_SDK_ROOT>\cygwin\jennic\SDK\Application for JN5139

where **<JN51xx_SDK_ROOT>** is the path into which the JN5148 or JN5139 SDK was installed.

2. Navigate to the **Build** directory for the application to be built and follow the instructions below for your chip type:

For JN5148:

At the command prompt, enter:

```
make clean all
```

Note that for the JN5148, you can alternatively enter the above command from the top level of the project directory, which will build the binaries for both applications.

For JN5139:

At the command prompt, enter:

```
make -f Makefile_JN5139.mk clean all
```

In all the above cases, the binary file will be created in the relevant **Build** directory, the resulting filename indicating both the chip type (**JN5148** or **JN5139**) and networking stack (**JN** for JenNet) for which the application was built.

3.3 Downloading the Application

The build process produces three binary files, one for each device type (Co-ordinator, Router, End Device). You must download each of these files to the relevant device(s):

- the Co-ordinator binary file to the Controller board
- the Router binary file to the Sensor boards with SMA connectors
- the End Device binary file to the Sensor boards with ceramic antennae

The following download methods are available:

- If you are using Eclipse (with the Flash Programmer external tool installed), you can download your **.bin** file from within the IDE – refer to the *Eclipse IDE User Guide (JN-UG-3063)*.
- If you are using Code::Blocks (the version available from the Jennic web site), you can download your **.bin** file from within the IDE – refer to the *Code::Blocks IDE User Guide (JN-UG-3028)*.
- Otherwise, you must directly use the JN51xx Flash Programmer application to download your **.bin** file – refer to the *JN51xx Flash Programmer Application User Guide (JN-UG-3007)*.

4 Code Description

This chapter provides details of the code used in the demonstration application. This should help you understand the code sufficiently to adapt the application.

4.1 Common Function Calls

4.1.1 Overview

The demonstration system consists of a Co-ordinator, which is also the system controller, and several Sensor boards that are configured as Routers and End Devices. The general structure of the code in each device is the same, with an initialisation followed by a main loop.

The Co-ordinator is initialised and starts a non-beaconing network. The Routers and End Devices then join the network and start to send regular sensor readings. There is no specific binding operation, since the network is pre-configured. The sensor devices send readings once per second, timed by a wake timer.

The Co-ordinator uses the source address of the received sensor readings to determine if it has received data from that device before, and displays the information as appropriate. As each device joins the network, a new graph is displayed for it.

4.1.2 Function Descriptions

The function descriptions in this section are intended to assist in creating a JenNet application using the Jenie API.

vJenie_CbConfigureNetwork()

This function is the first callback of an application and is called before the stack initialises itself, providing the application with the opportunity to initialise/override default stack parameters. The function is only called during a cold start.

vJenie_CbInit()

This function is called after the stack has initialised itself. It provides the application with the opportunity to perform any additional hardware or software initialisation that may be required.

vJenie_CbMain()

This function is the main application task. It is called many times per second by the stack and provides the opportunity for the application to perform any processing that is required. This function should be non-blocking.

vJenie_CbStackMgmtEvent ()

This function is called by the stack to inform the application that one of a number of stack management events has occurred. For example, the node may have received a service request response from a remote node.

vJenie_CbStackDataEvent()

This function is called by the stack to inform the application that one of a number of stack data events has occurred. For example, the node may have received a message from a remote node or a response to one of its own messages.

vJenie_CbHwEvent()

This function is called by the stack to inform the application that a hardware event has occurred - that is, an event has been generated by an on-chip peripheral of the JN5148 or JN5139 device.

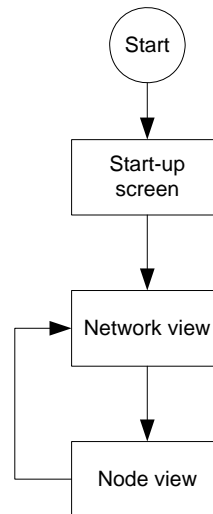
4.2 Co-ordinator

The Co-ordinator code (for the Controller board) is in the file **AN1065_JN_HomeSensorCoord.c** in the directory:

JN-AP-1065-JenNet-Home-Sensor-Demo\AN1065_JN_HomeSensorCoord \Source

4.2.1 Overview

The general operation of the Co-ordinator is described in Chapter 1. It goes through a series of states that reflect which screen is being displayed at any time, as shown below.



For each screen, there are associated functions in the code to create the screen, update the screen and handle any button presses.

Upon being started, the first action of the application is the initialisation of the hardware, stack and application variables. Once this has completed, the first screen is displayed and the main control loop is entered. This loop runs for the duration of the demonstration.

The main loop iterates 20 times per second, driven by events from a wake timer. The buttons are checked once per iteration of the loop, which avoids the need for any key de-bounce software algorithm without adding any perceived operating delay. Any events from the JenNet stack are processed as they occur, handling received frames from other nodes.

Once per second, the Co-ordinator updates the screen. At this time, it also keeps track of received frames from each node from which it has previously seen data. In this way, it can produce a count of the number frames it has missed from each node.

4.2.2 Function Descriptions

The functions described in this section are defined in the application code (in terms of supplied API functions) and are intended to assist in creating a JenNet application. As such, some functions are not mentioned, as they are not directly relevant to this goal.

vInitCoord()

Data relating to the sensor information is initialised here, as is the configuration of the demonstration.

vSetTimer()

This function uses the Integrated Peripherals API to start the wake timer for 1600 cycles (approximately 1/20th of a second).

vProcessCurrentTimeBlock()

This function implements the state machine. There is only one action – that is, to update the display.

vProcessKeys()

This function relays button presses to the appropriate function, depending on which screen is being displayed. The buttons are all 'soft' – that is, their function is dependent on the screen.

u8UpdateTimeBlock()

If the Co-ordinator is in the correct state, this function increments the state machine and returns the next time-block. The state machine is effectively disabled when some screens are displayed.

vProcessUpdateBlock()

This function updates the graphs for all nodes, scaling the data to fit, and then updates the LCD panel, if the appropriate screen is being shown.

Keypresses

vProcessSetChannelKeyPress

vProcessNetworkKeyPress

vProcessNodeKeyPress

These functions all operate in much the same way, responding to button presses to adjust values, set data or move to another screen.

vUpdateNetworkSensor()

This function is called to refresh the 'network' screen, which shows the same sensor type from all nodes at the same time.

LCD screens

vBuildNetworkScreen()
vBuildNodeScreen()

These functions all demonstrate the creation of a fresh screen on the LCD panel, using the Board API. Note that the font definition requires some odd characters to be used. For instance, ‘\’ is used for a ‘+’ and ‘]’ is used for ‘-’. Normally, a call to **vLcdRefreshAll()** would be used to put the new display onto the LCD panel, but in all cases an associated update function is called instead, which then calls **vLcdRefreshAll()**.

Extensive use is made of **vLcdWriteText()**, and **vBuildSetChannelScreen()** also uses **vLcdWriteBitmap()**, using the bitmap that is defined in **JennicLogo.c**.

vUpdateSetChannelScreen()
vUpdateNetworkScreen()
vLcdUpdateElement()
vUpdateNodeScreen()
vUpdateNodeControlScreen()
vUpdateSetupScreen()

These functions all update an existing LCD screen whilst retaining what was there before. Note the use of space characters to delete unwanted text. Since the font is proportional, each space is only 3 pixels wide, while most characters are 5 pixels wide. It is therefore necessary to use additional space characters to successfully erase a series of text characters.

vDrawGraph()

This function takes an array of 32 values, in the range 0 to 13, and makes a bitmap graph from them. This is an alternative to using a pre-defined bitmap and allows simple graphics to be created without the complication of a line-drawing algorithm.

An array is created for the two rows of data used by each graph, and a constant array is defined containing the values required for each value in the range 0 to 13. The array is then filled with the appropriate data for each item in the series of values.

A structure defines the bitmap in terms of the array, as being 33 columns wide and 2 character rows high. Finally, a call to the Board API writes the bitmap to the LCD shadow memory. A call to **vLcdRefreshAll()** must be made after calling this function for the bitmap to appear on the LCD panel.

Miscellaneous

vStringCopy()
vValToDec()

These functions do not make use of the stack or API, so are not described here.

4.3 Router

The Router code is in the **AN1065_JN_HomeSensorRtr.c** file in the directory:
JN-AP-1065-JenNet-Home-Sensor-Demo\AN1065_JN_HomeSensorRtr\Source

4.3.1 Overview

Like the Co-ordinator, the Router code uses a wake timer to determine the system timing. Once per second, the Router code reads the sensors and sends the read information to the Co-ordinator. It also uses hardware events on the DIO pins to detect when keys are pressed and when the sensor readings have completed.

4.3.2 Function Descriptions

The functions described in this section are defined in the application code (in terms of supplied API functions) and are intended to assist in creating a JenNet application. As such, some functions are not mentioned, as they are not directly relevant to this goal.

vProcessTxData()

This function creates and requests the transmission of the sensor information and to control an LED on the Co-ordinator (Controller board).

vTxRegister()

This function requests the transmission of a registration message to the Co-ordinator.

vProcessRead()

This function gets the latest readings from each sensor. If the light level causes the low-light alarm to be triggered, an LED is illuminated.

vInitEndpoint()

This function sets certain internal variables.

u8FindMin()

This function returns the minimum of two values.

4.4 End Device

The End Device code is in the **AN1065_JN_HomeSensorEndD.c** file in the directory:

JN-AP-1065-JenNet-Home-Sensor-Demo\AN1065_JN_HomeSensorEndD\Source

4.4.1 Overview

Once per second, the End Device code reads the sensors and passes the read information to the Co-ordinator. It also uses hardware events on the DIO pins to detect when keys are pressed and when the sensor readings have completed.

4.4.2 Function Descriptions

The functions described in this section are defined in the application code (in terms of supplied API functions) and are intended to assist in creating a JenNet application. As such, some functions are not mentioned, as they are not directly relevant to this goal.

vProcessTxData()

This function creates and requests the transmission of the sensor information and to control an LED on the Co-ordinator (Controller board).

vTxRegister()

This function requests the transmission of a registration message to the Co-ordinator.

vProcessRead()

This function gets the current readings from each sensor. If the light level causes the low-light alarm to be triggered, an LED is illuminated.

vInitEndpoint()

This function sets certain internal variables.

u8FindMin()

This function returns the minimum of two values.

Revision History

Version	Date	Description
1.0	28-Nov-2007	First release
1.2	07-Mar-2008	Updated for JN5139 device
1.3	28-May-2008	Updated to become built-in demo for JN5139-EK000 evaluation kit.
2.0	23-Feb-2010	Support for JN5148 and Eclipse added
3.0	24-Aug-2010	Application Note (JN-AN-1065) and User Guide (JN-UG-3046) merged

Important Notice

Jennic reserves the right to make corrections, modifications, enhancements, improvements and other changes to its products and services at any time, and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders, and should verify that such information is current and complete. All products are sold subject to Jennic's terms and conditions of sale, supplied at the time of order acknowledgment. Information relating to device applications, and the like, is intended as suggestion only and may be superseded by updates. It is the customer's responsibility to ensure that their application meets their own specifications. Jennic makes no representation and gives no warranty relating to advice, support or customer product design.

Jennic assumes no responsibility or liability for the use of any of its products, conveys no license or title under any patent, copyright or mask work rights to these products, and makes no representations or warranties that these products are free from patent, copyright or mask work infringement, unless otherwise specified.

Jennic products are not intended for use in life support systems/appliances or any systems where product malfunction can reasonably be expected to result in personal injury, death, severe property damage or environmental damage. Jennic customers using or selling Jennic products for use in such applications do so at their own risk and agree to fully indemnify Jennic for any damages resulting from such use.

All trademarks are the property of their respective owners.

Jennic
(An NXP Company)
Furnival Street
Sheffield
S1 4QT
United Kingdom

Tel: +44 (0)114 281 2655
Fax: +44 (0)114 281 2951
E-mail: info@jennic.com

For the contact details of your local Jennic office or distributor, refer to the Jennic web site:

www.Jennic.com
TECHNOLOGY FOR A CHANGING WORLD