EB358

# ZigBee Teachers Course Notes

**Contents**

## About this course

**Aims:** The principal aim of this course is to introduce you to the concepts involved in ZigBee networks.

On completing this course you will have learned about:

- the relationship between IEEE 802.15.4 and ZigBee protocols;
- the electrical principles behind ZigBee transmissions;
- the components that make up a ZigBee network;
- the topologies available to ZigBee networks;
- the addressing schemes;
- ZigBee routing;
- Sleep modes and packet buffering;
- ZigBee security options;
- common AT commands and syntax.

**What you will need:**

To complete this course you will need the following equipment:

- Flowcode software
- E-blocks including:
    - four Multiprogrammers (EB006) with a microcontroller device
    - one ZigBee E-Block ( EB051 C) –co-ordinator;
    - three ZigBee E-Blocks (EB051 R) – router / end-node;
    - an LED E-Block (EB004)
    - a Graphical Colour LCD E-Block (EB043)
    - a Keypad (EB014)
    - two Sensor E-Blocks (EB003)
    - a Switch E-Block (EB007)
    - a USB232 E_Block (EB039)

**Using this course:**

This course presents you with a number of tasks listed in the exercises in the following text. All the information you need to complete the labs is contained in the notes.

Before starting any exercises, you are advised to spend some time familiarising yourself with the material on this course so that you know where to look when you get stuck.

Time: If you undertake all of the exercises on this course then it will take you around twelve hours.

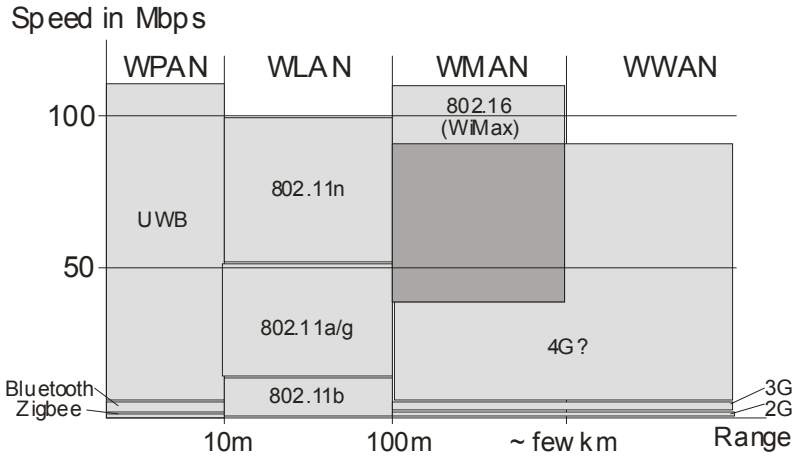
**Important note**: Information presented here is correct at the time this document was produced. Please check the Matrix Multimedia web site http://www.matrixmultimedia.com for the latest E-Blocks documentation.

# 1. Introduction to ZigBee

## 1.1 Overview

ZigBee is the name given to a specific suite of high level communication protocols using low power digital radios, based on the IEEE 802.15.4 standard for Wireless Personal Area Networks (WPANs).

The following diagram relates a number of wireless technologies used in WPANs, WLANs (Wireless Local Area Networks,) WMANs (Wireless Metropolitan Areas) and WWNAs (Wireless Wide Area Networks.) The speeds shown are guides only. WWANs are dominated by mobile phone (cell phone) technologies, known as 2G, 3G and, forthcoming, 4G. In the WPAN field, UWB (Ultra-WideBand radio technology) is a rapidly developing area, used to transmit high data rates over very short distances, opening up application such as video and audio streaming wirelessly around the home between a base device and subsidiary devices.




WPANs cover a radius of about 10m around a person or object. The core aim is to design systems offering low cost, low power, and compact size. The IEEE 802.15 working group has defined three classes of WPANs, differentiated by data rate, power requirements and level of performance. The high data rate WPAN technology, UWB, is suitable for multi-media applications that require very high performance levels. Medium rate WPANs (IEEE 802.15.1/Blueetooth) handle a variety of tasks ranging from mobile phones to PDA communications. The low data rate WPAN standard, ZigBee, is intended to serve a set of industrial, residential and medical applications with very low power consumption and cost requirement and with much lower requirements in terms of data rate and performance.

A ZigBee network links a number of electronic devices (nodes). Each node in the network forms part of the transmission chain, receiving messages, deciding if the messages are for local use, and re-transmitting them to other nodes in the network if not.

A common use of ZigBee is to form ‘sensor area networks’. For example in a factory environment many ZigBee nodes can be quickly installed to provide complete low power wireless coverage of the many sensors needed in a factory for fire and burglar alarm systems.

## 1.2 Comparison of Wireless Technologies

Comparing ZigBee with other wireless technologies:

**ZigBee:**

- was formally adopted in December 2004
- is targeting control applications in industry, which do not require high data rates, but must have low power demand, low cost and offer ease of use (remote controls, home automation, etc.);
- offers data rates of 250 Kbits at 2.4 GHz, 40 Kpbs at 915 Mhz, and 20 Kpbs at 868 Mhz with a range of 10-100m;
- currently offers three levels of security;
- costs around half that of Bluetooth;
- can network up to 256 devices;
- has power requirements much less than Bluetooth;
- uses star, tree or mesh topology.

**Bluetooth:**

- is designed for voice and higher data-rate applications;
- also operates in the 2.4 GHz spectrum;
- operates typically over a distance of 10 metres;
- allows for three modes of security;
- has a range of ~10 metres;
- has power requirements of ~40 to 100mW per device;
- can network up to 8 devices;
- costs around £2 per chip.

**Ultra-Wideband (UWB):**

- transmits digital data over a wide frequency spectrum using very low power;
- can transmit data at very high rates (for wireless local area network applications) over distances of up to 10m;
- has two competing UWB standards currently - one based on direct sequence spread spectrum techniques, (DS-UWB), the other based on Multi-band Orthogonal Frequency Division Modulation (OFDM), with each standard offering data rates around 500 Mbps at a range of 2 metres;
- has power demands typically twice that of Bluetooth;
- is typically twice as expensive as Bluetooth implementations.

**Wi-Fi (IEEE 802.11.x technologies)**

- is typically three times more expensive than Bluetooth implementations;
- uses around five times the power consumption of Bluetooth devices;
- is certified by the Wi-Fi Alliance;
- 802.11a uses OFDM, in the 5GHz band with data rates up to 54Mbps;
- 802.11b uses DSSS, in the 2.4GHz band with data rates up to 11 Mbps;
- 802.11g uses OFDM, in the 2.4GHz band with data rates up to 54Mbps;
- 802.11n is likely to operate in the 5GHz band with data rates over 100Mbps.

**WiMAX (Worldwide Interoperability for Microwave Access)**

- is a wireless metropolitan area network technology;
- has a range of 50 km with data rates of 70 Mbps;
- operates in the 10-66 GHz frequency bands, but requires line of sight;
- supports vehicle mobility of 20 to 100+ km/hr;
- was created to compete with DSL and cable modem access for rural, hard-to-wire areas.

**In summary:**

| | ZigBee | *Bluetooth* | 802.11a | 802.11b | 802.11g | 802.11n | UWB |
|---|---|---|---|---|---|---|---|
| **Throughput in Mbps** | 0.03 | 1-3 | 54 | 11 | 54 | 200 | 200 |
| **Max. range in m** | 30 | 10 | 45 | 60 | 60 | 150 | 10 |
| **Power in mW** | 30 | 100 | 1500 | 750 | 1000 | 2000 | 400 |
| **Bandwidth in MHz** | 0.3 | 1 | 20 | 22 | 20 | 40 | 500 |
| **Price/device in £** | 1 | 1.5 | 6 | 2.5 | 4.5 | 10 | 3.5 |

Reasons for choosing ZigBee include:

- low cost
- high reliability
- very long battery life
- high security
- self-healing properties
- large number of nodes supported
- ease of deployment
- guaranteed delivery
- route optimisation.

## 1.3 ZigBee Applications

ZigBee technologies offer a way to realise 'Smart' buildings, whether at work or at home. These can be either pre-programmed or operated by remote control.

A ZigBee-enabled control system offers:

- automatic control of lighting systems to create adaptable workplaces;
- versatile control of heating and ventilation to personalise the environment;
- security systems, both inside and outside the building;
- automatic environmental control of grounds and gardens.

At work, while staff and visitors can walk freely in and out of ZigBee-secured smart building, intruders trigger alarms. Health and Safety considerations are enhanced because workers no longer enter or leave darkened buildings or car parks. They trigger preset, personalised levels of heating and lighting in their work space, removing the need to contact building maintenance personnel. This results in savings in energy consumption, staffing costs and installation costs, which are reduced because wireless networks do not require the same levels of cable installation.

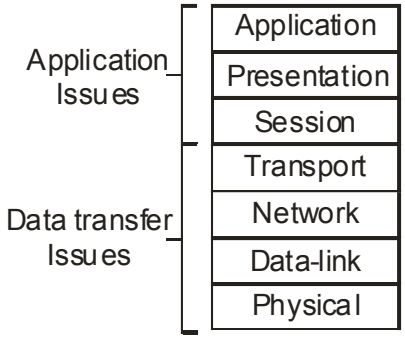
At home, the user has a digital key to unlock the door. The control system then identifies the user and adjusts the lighting, heat, and curtains accordingly. While the house is empty, the control system records the output of security cameras around the property. It monitors sensors looking for water or gas leaks, for freezing pipes, and for fire. It monitors appliances such as the fridge and freezer.

When the user goes to bed, as well as taking care of temperature and light levels, the ZigBee control system scans sensors that monitor the user's health. Should a wireless smoke detector detect fire, the central-heating system immediately turns off, lights come up to a low level, to help people find their way out of the house and the outside lights flash to make the it easier for the fire brigade to find the house.

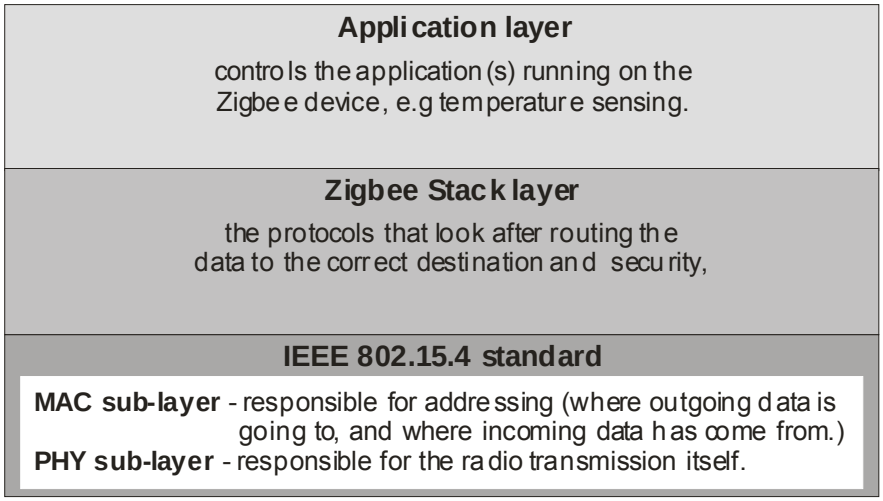
## 1.4 ZigBee protocol outline

Computer and computer-related networking is often simplified by using the OSI (Open System Interconnection) model. This divides the tasks involved with moving information between networked devices into smaller, more manageable groups. Each layer of the model is responsible for one of these task groups. Each layer is reasonably self-contained so that the tasks assigned to each layer can be implemented independently of other layers. This enables the solutions offered by one layer to be updated without adversely affecting the other layers.

The next diagram shows the seven layers of the full OSI model. It shows that these layers can be grouped into two areas of action, application issues, which are usually implemented in software, and data-transfer issues, which are realised in hardware and software. The latter includes the Network layer, which deals with planning the route that the data will take between source and destination devices, the Data-link layer, which controls when a device can transmit over the medium being used (copper cable, air, glass fibre etc., and the Physical layer, which includes the electrical and signalling standards used by the transmission.




The ZigBee protocol can be viewed as three separate layers, each with its own specific functions. The Application layer and some of the functions of the ZigBee Stack layer are usually controlled by a microcontroller, such as a PIC chip. In a typical system, the IC contains circuitry to meet the requirements of the physical layer (PHY) and a portion of the media access controller (hardware-MAC). The remaining MAC functions (software-MAC) may also be executed on the microcontroller.



## 1.5 The IEEE 802.15.4 Standard

The IEEE802.15.4 standard for wireless communication was developed by the IEEE (Institute for Electrical and Electronics Engineers,) an American technical / professional association responsible for a range of communication technologies.

For example, the IEEE 802.11 standard covers communications in wireless LANs, while the IEEE 802.16 standard embraces broadband wireless Metropolitan Area Network communications.

Whereas these two standards are concerned with high data rate communication, the IEEE 802.15.4 standard offers low data rate applications where simple connectivity and low power consumption are important. It defines both the "physical layer" and the "medium access layer".

### 1.5.1 The IEEE 802.15.4 PHY

The physical layer specifications (PHY) define three low-power unlicensed radios. These include a radio operating at 2.4 GHz with a basic bit rate of 250 kilobits per second, a radio for the US market operating at 915 MHz, and one for Europe and Japan, operating at 868 MHz. These latter frequencies have data rates of 40 kb/s and 20 kb/s, respectively.

The 868MHz band offers only one communication channel, (channel 0), while the 902MHz band offers ten (channels 1 to 10).

In the 2.4 GHz band, there are sixteen communication channels, each 5 MHz wide, numbered from 11 to 26. Although a maximum data rate of 250 kbps is specified, the actual throughput is approximately half of that due to the overhead of the protocol, (additional bytes needed for addressing, security and error-checking.) The transmission uses DSSS, (Direct Sequence Spread Spectrum,) and O-QPSK (Offset Quadrature Phase Shift Keying ) modulation.

Other responsibilities of the PHY layer include detecting transmissions from new nodes, and assessing the quality of links with other nodes.

### 1.5.2 The IEEE 802.15.4 MAC

The medium access layer (MAC) specification defines how multiple 802.15.4 radios operating in the same area will share the airwaves. It employs CSMA-CA (Carrier Sensing Multiple Access with Collision Avoidance) to control when a node is allowed to transmit. (A collision occurs when two nodes transmit at the same time.)

The MAC layer also governs any security protocols applied to the link.

The maximum length of an IEEE 802.15.4 packet is 127 bytes. This includes a two byte CRC (cyclic redundancy check) value, used for error-checking.

There is an optional mechanism to ensure that data is transmitted successfully, using an Acknowledgement bit. When this bit is set, the receiver must acknowledge that the data has been received, by sending back an acknowledgement transmission. If this acknowledgement is not received within a certain time, the transmitter will retransmit the data, and do this for a fixed number of times before declaring that there is a transmission error. However, receiving an acknowledgement simply indicates that a frame was properly received by the receiver’s MAC layer. It does not, however, indicate that the frame was processed correctly. It is possible that the MAC layer of the receiving node received and acknowledged a frame correctly, but due to the lack of processing resources, a frame might be discarded by upper layers.

### 1.5.3 Modulation techniques

In all radio communication, the signal (the information) is modulated onto a carrier wave, by using it to change slightly (modulate) a property of the carrier wave. The three main types of modulation are:

- amplitude modulation (AM) – the information signal changes the amplitude (height) of the carrier wave;
- frequency modulation (FM) – the signal changes the frequency of the carrier wave slightly;
- phase modulation (PM) – the signal causes slight changes in the phase of the carrier wave. This is the basis of the modulation used in ZigBee and it will be further examined shortly.

In the case of ZigBee, the signal is digital, either logic 0 or logic 1. The term 'keying' is used to describe digital modulation, referring back to early telegraphy when a Morse key was tapped to send a signal.

With PSK (phase-shift keying), the phase is changed by $180^0$ when the signal changes from logic 0 to logic 1, or back again. The next diagram illustrates this:

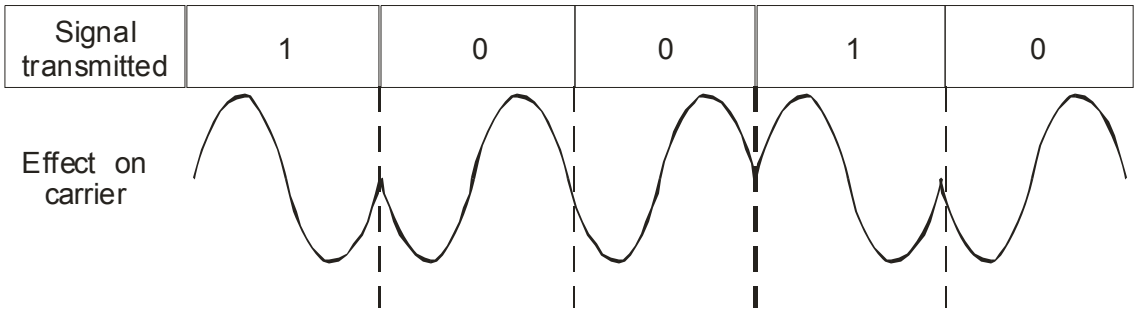
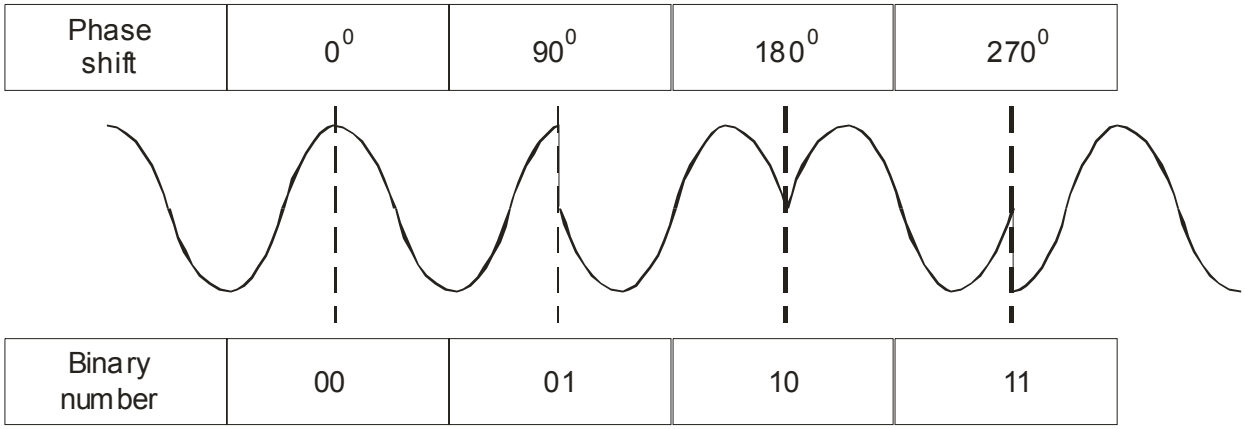


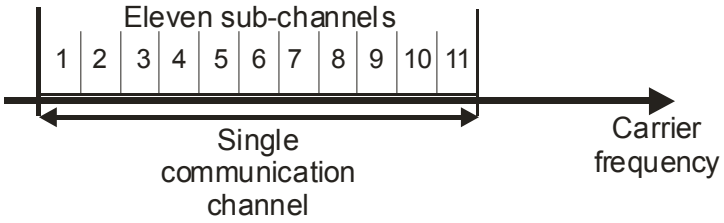

Illustration of PSK

QPSK (quadrature phase-shift keying) uses phase-shifts of $90^0$ instead of $180^0$. This allows four changes in phase instead of only two. As a result, the phase of the carrier can represent two binary digits, 00, 01, 10 and 11, effectively doubling the bandwidth of the carrier.




O-QPSK (Offset QPSK) is a variation which uses different values of phase-shift to minimise the changes in amplitude that take place in the resulting signal.

The principle of DSSS is shown in the following diagram:




Although the signal is binary, either logic 0 or logic 1, it is not transmitted as a single change in amplitude, frequency or phase. Instead, these two possible states are transmitted as different series of bits, called 'chips'. Each chip is transmitted on its own sub-channel, simultaneously with the other chips.

For example, eleven chips could be used to distinguish between logic 0 and 1:

Logic 1 = 00110011011 Logic 0 = 11001100100

These sequences are so different that even if substantial corruption occurs during transmission, logic 0 and 1 can still be distinguished.

### 1.5.4 IEEE 802.15.4 Device Types

The standard defines two types of device:

- a full-function-device (FFD) - These typically perform network management functions such as routing, coordination, networking formation, and other management functions;
- a reduced-function-device (RFD) - These typically interact directly with the application processes and sensors. These nodes (often referred to as leaf nodes) contain the firmware and hardware that perform data capture, control functions, and other application specific functions. They can also be mobile, depending on application, and therefore have stringent requirements for low power and memory space.

FFD devices usually have higher power requirements to permit "always on" operation to facilitate network routing, data analysis/aggregation, and other more demanding operations. An FFD can talk to RFDs and other FFDs, whereas a RFD can only talk to a FFD.

## 2. The ZigBee network

As we have seen, the MAC layer of the IEEE 802.15.4 introduced two types of **physical** device, the full-function-device, FFD and the reduced-function-device, RFD. The main differences are summarised in the following table, which introduces new concepts which will be developed in this section:

| Full Function Device | Reduced Function Device |
|---|---|
| Found in any topology | Found only in star topology |
| Can be a network co-ordinator | Cannot be a network co-ordinator |
| Can talk to any type of device (FFD or RFD) | Talks only to the network co-ordinator (FFD) |
| Usually mains powered | Usually battered powered |

- The RFD requires only limited RAM and ROM resources and is thus less expensive.
- ZigBee RFDs can search for available networks, transfer data from its application, request data from the network co-ordinator, and sleep for extended periods of time to increase battery life.
- RFDs can talk only to a FFD.
- The FFD can serve as a network co-ordinator, a router or as an end device.
- Any FFD can discover and talk to other FFD and RFDs.

### 2.1. ZigBee Logical Devices

At the Network layer of the ZigBee stack, the software is able to recognise three types of **logical** device:

- ZigBee network **co-ordinator**;
- ZigBee **router**;
- ZigBee **end device**.

**Co-ordinator:**

- The co-ordinator is a FFD device, responsible for creating the network, assigning network channel, assigning network addresses and adding other nodes to the network.
- There can be only one co-ordinator node for a given ZigBee network.
- The co-ordinator is normally powered via the electricity mains, and so is usually located in a fixed position. As a result, it is often convenient to use this node to access the network.
- New child nodes can usually join the network at any time if the co-ordinator is switched on,
- If the co-ordinator is switched off, new nodes are unable to join but nodes that already have an address on the network, such as sleeping end devices and roaming devices, will still run correctly.

**Router:**

- The router is a FFD device, responsible for buffering packets, and routing signals between nodes.
- It can be used to extend the range of the network.
- It is also normally mains powered so that it can maintain routing, and also buffer any data sent to a sleeping node.
- Since router nodes are also usually located in a fixed position, due to the power requirements, they too can be used to access the network - useful if the co-ordinator has been disabled to stop any new nodes joining the network.

**End Device:**

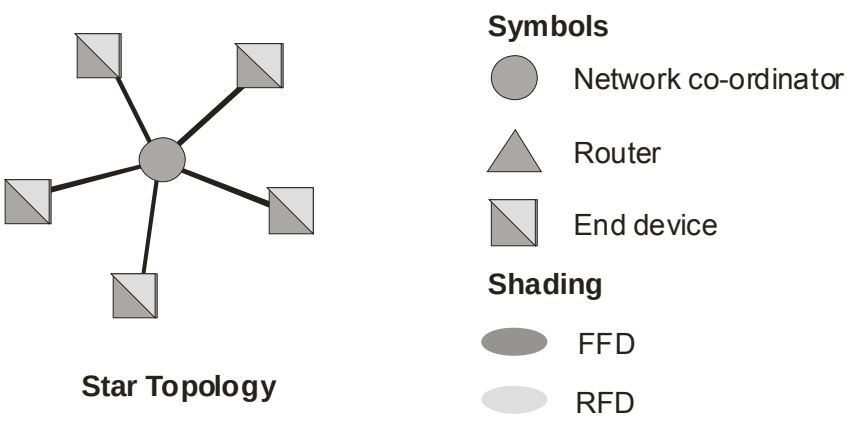
- This is used to communicate between the network and the real world, and is either a FFD or a RFD device. For example end devices can be placed into sensors, switches, displays, actuators etc.
- End devices are low in power consumption, and are highly suited to battery operation.
- They can be placed in sleep mode to further reduce power consumption.
- End device nodes cannot communicate directly with other end device nodes. Instead, they communicate through routers or the network co-ordinator.

## 2.2. ZigBee Network Topologies

The Network layer of the ZigBee stack recognises three topologies:

- Star topology
- Cluster tree topology
- Mesh topology

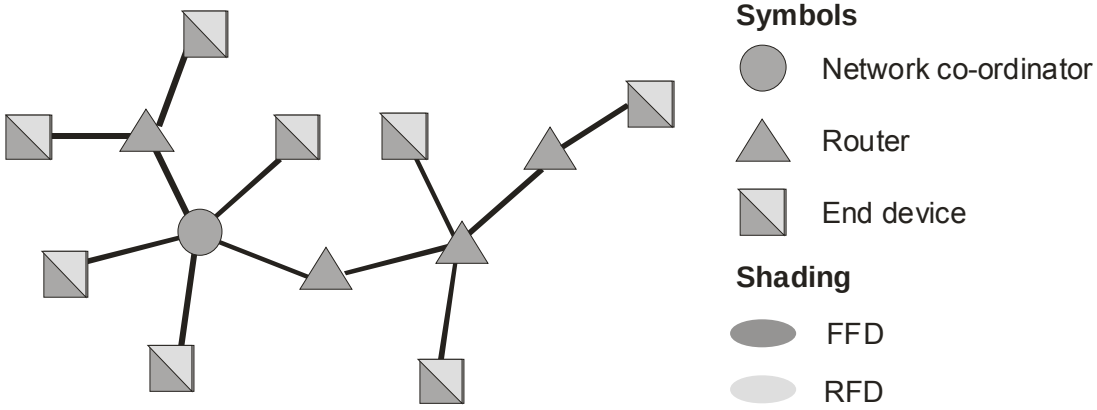
### 2.2.1 Star Topology




**Star Topology**

A star network consists of one network co-ordinator and one or more end devices. All end devices communicate only with the co-ordinator, so that when an end device needs to transfer data to another end device, it sends its data via the co-ordinator. The star network is termed a single-hop network, as the co-ordinator is only a single hop (link) away. The reliability of this type of network is reduced because of the single point of failure, the co-ordinator.

### 2.2.2 Cluster Tree Topology



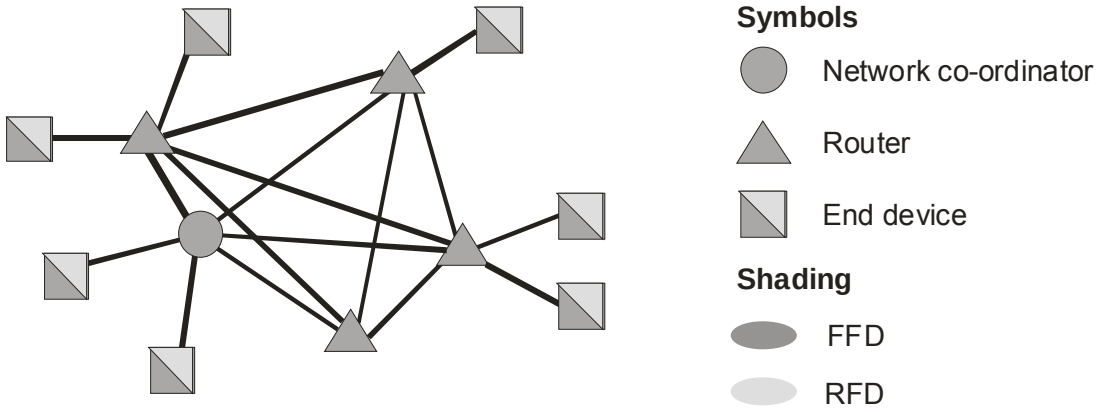

**Cluster Tree Topology**

In the cluster tree topology, the co-ordinator, the parent, is linked to a number of routers and end devices, its children.
In turn, a router may then be linked to more routers and end devices, its children.
Routers serve two functions:

- They increase the number of nodes that can join the network.
- They extend the physical range of the network, so that an end device does not have to be within radio range of the co-ordinator.

In a cluster tree topology, a child can directly communicate only with its parent (and with no other node). A parent can directly communicate with only its children and with its own parent. As in the star topology, end devices cannot communicate directly with each other. Messages are routed through a router or through the co-ordinator. There may be a number of possible paths between source and destination. The router chooses an appropriate route from these. This type of topology is called multi-hop.

### 2.2.3 Mesh Topology




**Mesh Topology**

The mesh network is a modification of the cluster tree topology:

- The co-ordinator is linked to a set of routers and end devices, its children.
- A router may be linked to more routers and end devices as its children.

However, communication is more flexible. RFD end devices are still unable to communicate with each other directly. FFD end devices, however, can communicate directly, without having to follow the tree structure. Messages to RFDs must still go through the RFD's parent node.

The advantages of this topology are:

- message latency (delay) can be reduced by route optimisation;
- reliability is increased, as there are alternative routes available, should a link go down.

This is also a multi-hop network.

## 2.3. Multi-Access Networks

All three topologies allow a ZigBee node equal access to the network. As a result, these networks are known as multi-access. There are two types of multi-access mechanisms, beacon and non-beacon.

### 2.3.1 Non-Beacon Access

Nodes in a non-beacon network are allowed to transmit any time that the radio channel is open and idle. This creates a 'free-for-all' environment in which collisions occur regularly when two or more devices try to transmit at the same time. In this mode, the co-ordinator and routers must be active at all times, and so it is best suited to mains-powered devices.

### 2.3.2 Beacon Access

A node in a beacon enabled network can transmit only in its designated time slot. This regulates transmissions making collisions less likely. The co-ordinator periodically generates a superframe, identified as a beacon frame. All nodes in the network are expected to synchronize their on-board clocks to this frame. Each node is allocated a specific time-slot within this superframe during which it, and only it, is allowed to transmit and receive its data. Usually, in a beacon-based network, an end device node will wake up just before this superframe is generated, will transmit / receive data at the appropriate point and then go back to sleep until just before the next superframe. A superframe may also contain a common time slot during which all nodes compete to access the channel. In this mode, the co-ordinator can be battery-powered, as it and end devices can sleep for most of the time, extending battery life.

## 2.4 Creating a ZigBee network

A new ZigBee network is started by a would-be co-ordinator which searches for other co-ordinators transmitting on its allowed frequency channels. If possible, it identifies a quiet frequency channel, establishes its own network and selects a unique 16-bit PAN (Personal Area Network) ID for that network.

After that, other devices can join the network as routers or end devices. Both routers and the co-ordinator have the capability to allow other nodes to join the network. The chain of events is as follows:

- The new device scans all available channels, looking for active networks.
- If the device detects both routers and co-ordinator from the same network, it usually chooses the strongest signal and tries to connect to that device.
- The device then sends a 'Can I join?' message to the relevant router or co-ordinator.
- The prospective parent allows the device to join if the network configuration permits it. The parent then allocates an address for the new child device.

The co-ordinator is configured to set a maximum number of permitted associations (children) per parent. To be more precise, it specifies the maximum number of child devices allowed per router, and directs how many of these children may be routers themselves.

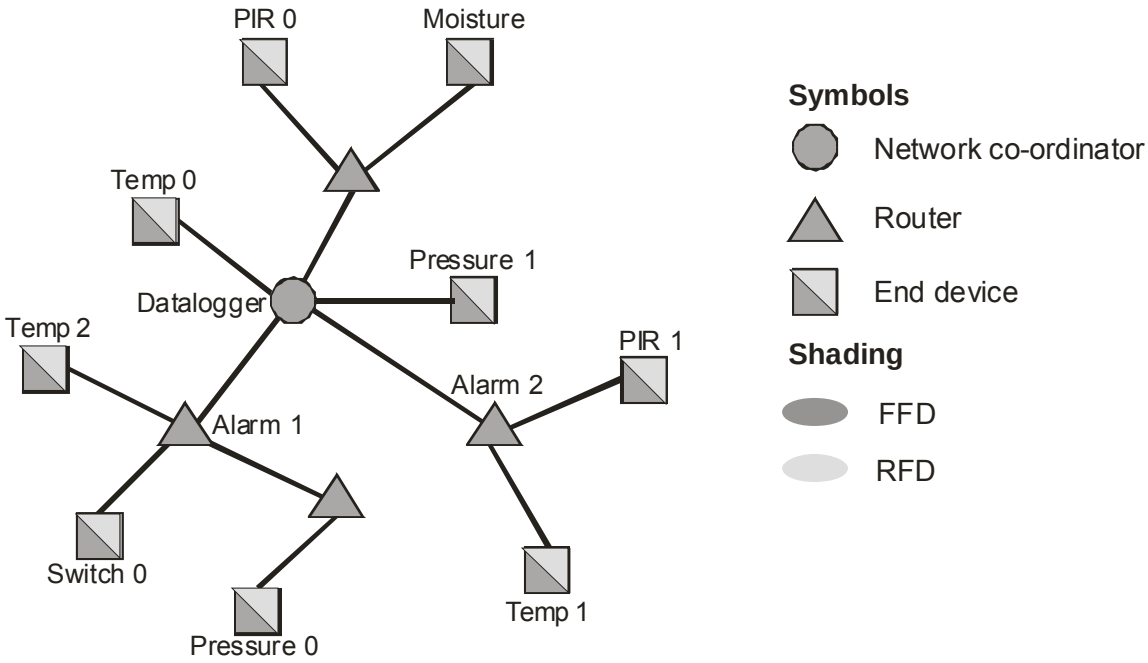
It also sets the maximum depth of the network (maximum number of hops from the co-ordinator to the most distant device.)

These limits may prevent a node from joining a network if its potential parents have already associated the maximum allowed number of children, or would breach the maximum network depth if the association was allowed. Such nodes are called orphan nodes.

The network devices store information about other nodes in the network, including parent and child nodes, in an area of non-volatile memory called a neighbour table.

Once in a network, a device can disassociate from the network either by being requested to leave the network by its parent or by requesting disassociation itself.

### 2.4.1 A ZigBee network example




Here, the co-ordinator is the overall parent as it was responsible for starting the network and allowing other nodes to join. In this case, the co-ordinator has five child nodes, three configured as routers and two as end devices. Two of the routers are also end devices. If the co-ordinator has child end devices directly connected to it that are configured to sleep, then the co-ordinator is responsible for buffering incoming messages for those sleeping nodes. Router nodes are also capable of having child nodes that can be either other router nodes or end device nodes. Again, if a router has child end devices directly connected to it that are configured to sleep, then that router is responsible for buffering incoming messages for those sleeping nodes.

## 2.5 ZigBee Addressing Scheme

The MatrixMultimedia ZigBee kit uses Version 2 XBEE ZigBee modules which are configured to associate up to seven child nodes.

Every node on a ZigBee network can have up to three different types of address assigned to it:

- a MAC address;
- a network address;
- a name.

### 2.5.1 MAC Addresses

Each ZigBee node comes complete with its own unique 64-bit MAC address, assigned by the IEEE. As a result, it is also known as the IEEE address or extended address. This can be used to secure the network against unwanted communication. By scanning the MAC address of a node wanting to join a network and referring to a list of permitted MAC addresses, the co-ordinator can allow or deny access to that node..

### 2.5.2 Network Addresses

This is a sixteen bit address which identifies the node to the rest of the network. It is not globally unique, so that two nodes on separate networks may have the same network address. The network address is also called the short address and is allocated by the parent node, router or co-ordinator, when the node joins the network. The co-ordinator always takes the network address 0x0000. Having sixteen bits gives a possible $2^{16}$ nodes (65536) on the network.

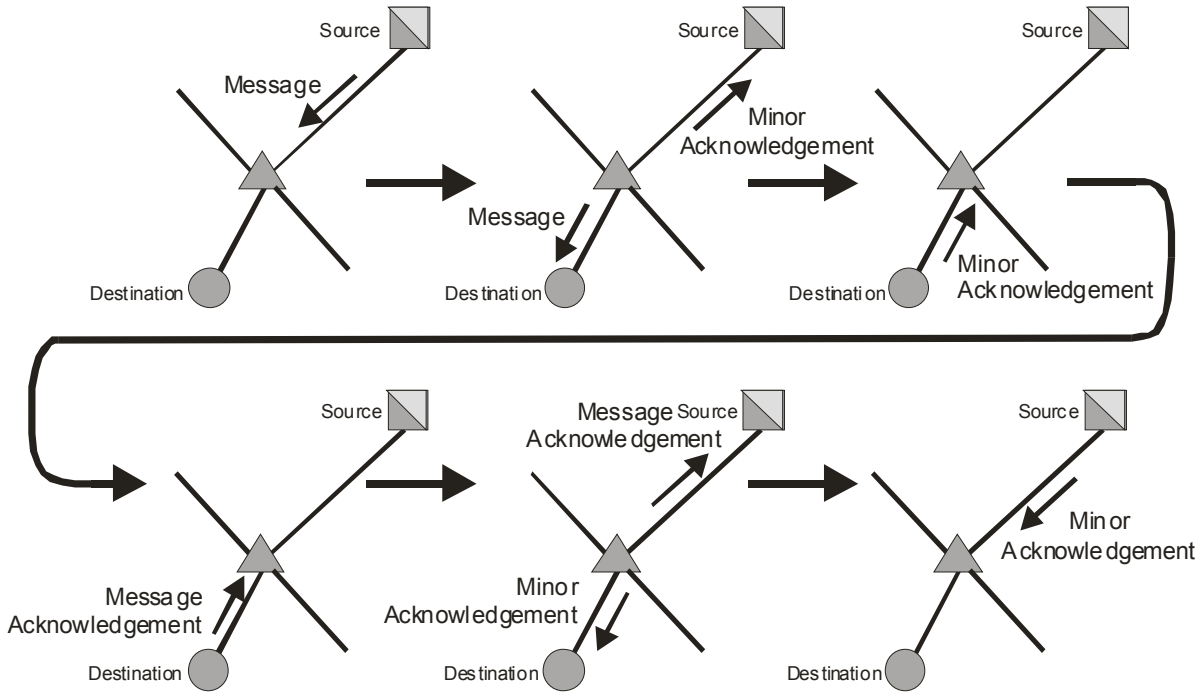
Under some circumstances, the network address may change. For example, the node may be mobile, or the parent router may become disconnected. As a result, it is not good practice to address nodes directly using the network address.

### 2.5.3 Unicasting and Broadcasting

Devices use their extended addresses to communicate with network devices while in the process of joining the network. Once the device has joined, it is assigned a sixteen bit network address, which is then used to communicate with the other network devices.

The 802.15.4 specification uses two forms of communication between devices in the network, broadcast and unicast messages.

- Unicast
  In a unicast message, the network address of the destination node is provided as the destination address in the MAC layer header of the message. Only the device that has this address accepts the message. The destination device may then transmit an acknowledgment, as a unicast message back to the source node, to confirm reception of the message. The sequence of acknowledgements is shown in the next diagram:



- Broadcast

  In a broadcast message, the MAC layer destination address is 0xFFFF. All active devices in the network receive and analyse the message. As other nodes detect the broadcast, they retransmit the message to extend the range of the broadcast.
  This form of addressing is used when:
  - joining or rejoining a network;
  - discovering routes in the network;

  Broadcasts cause substantial disruption to the operation of the network, as all nodes must take action, adding to the delays present in transmission times. As a result, the number of broadcasts should be kept to a minimum especially in larger networks.
  Although broadcast messages are not acknowledged in the formal sense, the ZigBee protocol implements a passive acknowledgement of broadcasts. When a device originates or retransmits a broadcast packet, it listens to check that all its neighbours retransmit the broadcast within a configurable period of time. If they do not, it will re-transmit the broadcast.

Unicast messages take much less time, as only the source and destination nodes, plus any intermediate stepping stone nodes are involved. The other advantage is that the transmitting node receives an acknowledgement that the message has been received by the remote node.

### 2.5.4 Name address

Each node can be assigned an optional name address string of up to twenty ASCII characters. The name address is useful for creating dynamic networks that are capable of scanning for particular node types. For example, all nodes which include sensors can be given a name starting with SN. When the co-ordinator is adding nodes to the network it can scan the node names and if the scanned name begins with SN, then it knows that the node has sensors available. This allows for a network to be formed and then at a later date extra sensor nodes can be added to the network without having to edit the existing code.

## 2.6 Routing and route discovery

In order for a message to travel from source node to destination node, there must be a mechanism that allows one node to locate and send data to another.

In a simple star network, this is not a complex task, as all end devices communicate directly through the central co-ordinator. Other topologies demand a more multifaceted approach, and routes are learned and established in a number of ways.

In a cluster tree network, there are two options, tree routing and route discovery. For small, static networks, tree routing is the best option. If there is a risk that nodes might stop working, or if reliable data delivery is essential, tree routing may be ineffective, and it may be better to use the more complex route discovery process.

In a mesh network, routing must rely on the route discovery process to build and maintain routing tables in FFD devices.

### 2.6.1 Tree routing overview

The plan is to use the tree structure to route packets from source to destination. The first decision the source device must make is which way, up or down the tree, should the packet be sent. The answer is found by examining the addressing structure. Basically, if the destination node is a descendant, the device sends the packet to one of its children; otherwise, it sends it to its parent.

When a device receives the packet, it checks to see whether it, or one of its child devices, is the destination. If so, the device either accepts the packet itself, or passes it to the appropriate child device. If not, it passes the packet to its parent

The problem with this is that the path taken by the packet may be longer than necessary because it followed the tree structure, rather than looking for the shortest possible path.

Routing efficiency is improved when ZigBee allows routers to discover shortcuts, using the path discovery process. Each router involved must maintain a routing table mapping destination addresses to next hop device addresses.

### 2.6.2 AODV algorithm overview

The protocol involved in path discovery is part of the Network layer of the ZigBee stack. The process often involves the use of broadcasts, - the transmitting node does not know where to send the message, and so it uses a broadcast that all nearby devices will pick up. The ideal is to use these broadcasts only when necessary, in order to limit the disturbance they cause to the network.

The techniques used are based on the AODV (Ad hoc On demand Distance Vector) algorithm. End devices do not store routing information. Only when they need to communicate do they use the path discovery process to establish a route between them, (hence the ‘On demand’ part of the AODV title.)

This process starts when the sending node, or its parent, broadcasts a Route Request Packet. This contains information such as the desired destination network address, (obviously), the source network address (so that the sending node can eventually receive the required information,) and a Route Request ID, so that the receiving nodes know whether they have seen this request before.

When a node receives this request, it uses the Route Request ID to check whether it has already received it. If so, it simply drops the packet. Otherwise, it increases the hop count (measure of how good/long the route to the destination is) and re-broadcasts it.

As the request travels from node to node, it automatically sets up the reverse path back to the source, because each node en route records the address of the neighbour from which it first received the request.

All routers eventually see the broadcast including the destination node or its parent, which then sends back a reply addressed to the source device. As the reply travels back through the network, each router in the path can build a routing table entry containing the best path to the destination node.

## 2.7 Sleep mode

End device nodes use sleep mode to reduce their power consumption and thus increase the device's battery life. When in sleep mode, they still retain their network address and so are still joined to the network. On wakeup, they do not need to contact the co-ordinator to rejoin the network. This means that the co-ordinator can be switched off once the network is established even if some of the devices are in sleep mode.

If an end device is in sleep mode, it cannot communicate with the rest of the network and so its parent device has the job of buffering (storing) any messages that are sent to the sleeping node. When the sleeping node wakes up, it must check whether its parent has any stored messages.

The Version2 XBee module can use two different forms of sleep mode:

- cyclic sleep;
- pin sleep.

### 2.7.1 Cyclic sleep

Cyclic sleep is one way in which devices can conserve power by using periodic sleep and wake cycles. During the sleep cycles, a parent node will buffer messages for the sleeping device. When the device wakes, it sends a poll request to the parent. At this point, the device receives any messages stored in the parent, or can transmit a message. After doing so, the device returns to sleep mode.

This operation is governed by a timer which is reset every time the module enters sleep mode. When the timer times out, the module wakes up and sends poll requests to its parent device to check whether any unicast messages have been buffered while it was in sleep mode. (The parent does not store broadcast transmissions for the child device.)

The parent will accept a message addressed to the sleeping device's network or MAC address and hold on to it in a buffer until the device wakes and requests the information. These messages are referred to as indirect messages. The parent will only queue one indirect message at a time. Indirect messaging does not work with a broadcast message.

The timer duration is linked to a similar timer on the parent node called the buffer timeout period. When this time expires, any messages stored in the parent are cleared. If the message has not been collected within the timeout cycle then the parent device knows that there is a problem with the end device. This problem can then be logged and used to alert that a node has failed to wakeup.

### 2.7.2 Pin sleep

Pin sleep puts the module to sleep and wakes it from sleep by changing the voltage on the sleep-enable pin on the XBee chip. When this pin is taken to logic 1, the module will finish any transmit or receive operations, and then enter sleep mode. While asleep, the module will not respond to network activity. To wake a module from pin sleep, apply logic 0 to the sleep-enable pin.

When using this mode of sleep, packets destined for the sleeping node will only be buffered for a finite period of time. Therefore the node must be woken up at regular intervals to poll for any incoming data.

## 2.8 Security

ZigBee networks are highly secure, incorporating several measures to prevent unwanted intrusion from hackers or from neighbouring ZigBee networks. These measures include:

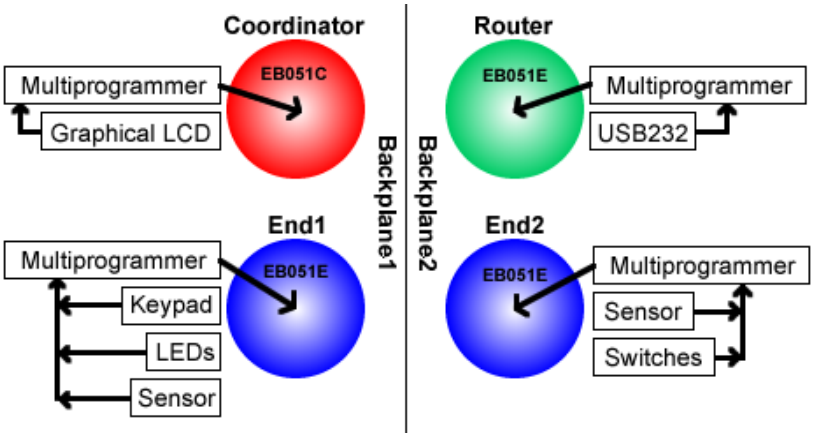
- Access Control Lists: Only pre-defined “friendly" nodes can join a network. The co-ordinator contains a table of permitted MAC addresses.
- Message Freshness Timers: Timed-out messages are rejected, preventing replay attacks on the network. A replay attack occurs when an attacker records a transmission and replays it later. For example, someone might record the command used to open a door controlled by a security system, and then retransmit it later to try to gain entry.
- 128-bit AES-based Encryption: The XBee modules have a built in 128-bit AES encryption algorithm that can be enabled to secure the ZigBee network. If this is enabled, the co-ordinator will start up using a 128-bit AES encryption key. Only devices that have the same key can communicate on the network. Routers and end devices wishing to join the network must either have been configured with that security key, or, as a less secure option, must obtain it over the air, as an unencrypted transmission, when they join.

  If encryption is enabled, all data to be transmitted will be encrypted using the sixteen byte encryption key. If any non-encrypted transmissions are received, the ZigBee node will ignore them.

  This security comes at a price. For multi-hop transmissions, each router along the route must decrypt and re-encrypt the data. As a result, latency will increase. In addition, enabling security adds several bytes of overhead to each packet.

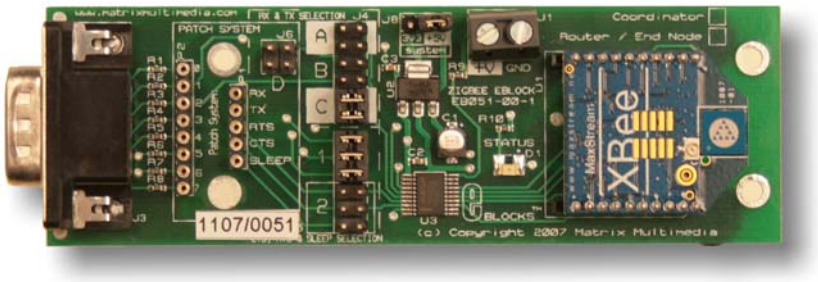
## 3. The Matrix Multimedia ZigBee training solution

This course is based around a four-node network, consisting of two sensor nodes, a monitoring/control node and a gateway node. Not all nodes are required for every task, and, for some, you will need to reconfigure some of the nodes. A fifth connection point is available, which is used in conjunction with the ZENA ZigBee Analyser to monitor network traffic.




The ZigBee training solution consists of two metal backplanes panels with two ZigBee nodes attached to each. The E-Blocks on each panel are connected in such a way that both nodes are powered from a single PSU attached to one of the programmer boards. One USB cable is needed for programming each of the microcontrollers in turn. A second USB port on the PC is needed for the ZigBee Analyser or the USB232 board gateway. If two USB ports are not available then the USB devices can be interchanged as needed.

### 3.1 The ZigBee E-Block




A 9 way D-type connector is used to connect the ZigBee E-Block to the input / output pins on the microcontroller. A jumper system is used to configure the pins for custom setups or for different microcontroller devices. At the other end of the ZigBee E-Block is the Version 2 XBee module, used to set up a wireless connection between the microcontroller and the ZigBee network.

The on-board LED displays the connection status of the node as follows:

- When the node is configured as part of a network, the LED will flash at 2Hz.
- When the node is configured as the network co-ordinator, the LED will flash at 1Hz.
- When the node is not connected to a network then the LED will remain off.

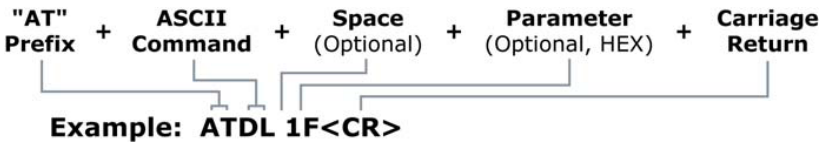
The XBee modules supplied with the ZigBee boards are configured in one of two modes, Co-ordinator or Router/End device. The Co-ordinator (EB051C) must be used to start the network. There can be only one co-ordinator node on the network. The Router/End Device node (EB051E) can be configured as a routing device or as an end device. There can be almost any number of these on the network.

### 3.1.1 AT Commands

The V2 XBee modules are controlled by a set of ASCII commands knows as AT commands.

The modules are pre-set in data transfer mode and must be changed into command mode by sending a specific sequence of commands. By default this sequence of commands comprises of a thirty millisecond pause, a string of three ASCII '+' characters and a second thirty millisecond pause. If the command sequence is entered correctly then the module returns an "OK" string.

The diagram shows the structure of a typical AT command. Some of the more commonly used AT commands are shown in the reference data section at the end of these notes. For a complete list of supported commands, please refer to pages 29 to 34 in the V2 XBee module datasheet in the Datasheet folder of the solution CD.




The command mode is ended by waiting for the default timeout period of ten seconds or by sending the exit command "ATCN".

Any data sent to the V2 XBee module while it is in data transfer mode will result in transmission to the destination address specified. By default no destination node is specified and so any data is transferred as a broadcast message to all nodes

## 3.2 Installation

- Install Flowcode from the Flowcode CD.
- Check the accompanying CD for any updates to Flowcode. The Readme.txt on the CD will explain any steps required to install and check the update.
- The ZigBee analyser requires the installation of a USB driver. See the ZENA folder on the accompanying CD for instructions and driver files. IMPORTANT – Do not plug in the USB ZigBee analyser until asked to do so by the installation routine.

### 3.2.1 Setting up the ZigBee system

The standard ZigBee node consists of a EB006 Multiprogrammer, containing a PIC16F877A, with a EB051 ZigBee E-Block attached to Port C. A 19.9906 MHz crystal is used to drive the PIC microcontroller. The ZigBee E-Block jumpers should be set to C and 1 respectively to suit the pins of the PIC16F877A microcontroller. For other microcontrollers, the connection port and jumper settings may need to be altered.

### 3.2.2 Setting up the ZigBee nodes

The four ZigBee nodes are set up as follows:

Node 1 – Co-ordinator and display node
Multiprogrammer
ZigBee E-Block - Port C
Graphical LCD E-Block - Port D

Node 2 – Keypad / Sensor end device node
Multiprogrammer
Sensor E-Block - Port A
LED E-Block - Port B
ZigBee E-Block - Port C
Keypad E-Block - Port D

Node 3 - Sensor end device node
Multiprogrammer
Sensor E-Block - Port A
ZigBee E-Block - Port C
Switches E-Block - Port D

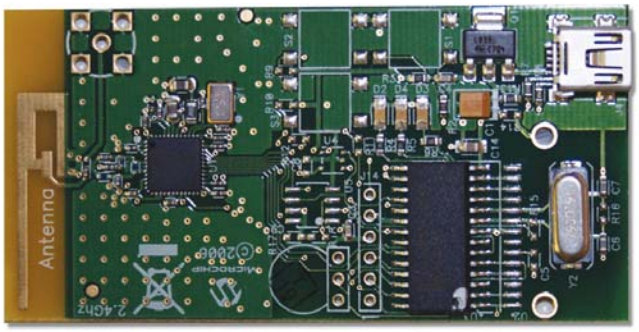
Node 4 - Router and gateway node
Multiprogrammer
ZigBee E-Block - Port C
USB232 E-Block - Port D.

### 3.2.3 Setting up the ZENA ZigBee analyser node

Any useful ZigBee network requires at least two nodes, and hence two separate embedded programs running at the same time. Flowcode cannot simulate them both simultaneously. To get around this problem a the Microchip ZENA ZigBee analyzer is included in the ZigBee solution pack.

The ZENA analyser, connected to a computer, allows the user to monitor and debug a ZigBee network. It uses a USB mini-B cable to connect to, and receive power from, the computer. A PCB trace antenna on the ZENA board receives transmissions on the specified channel and sends the information over USB to the computer.

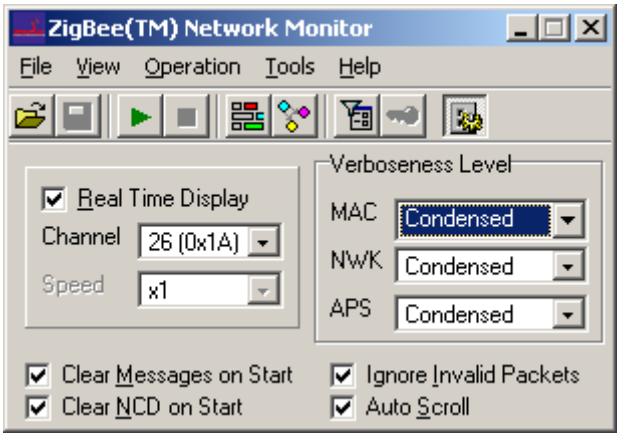
The analyser comes complete with set up instructions, documentation and software on the accompanying ZENA CD. Please refer to the documentation for details on installing and setting up the software.




The ZENA ZigBee analyser

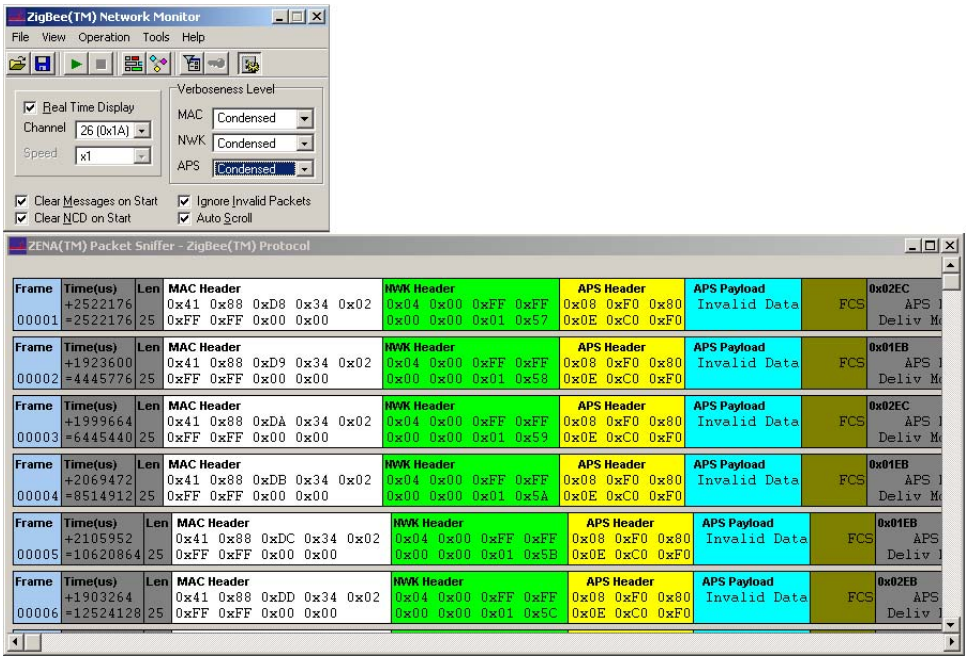
When the installation program prompts, connect the ZigBee analyser to the computer with the USB cable. Run the ZENA ZigBee analyser software, which should appear in your Start menu as “Microchip Software Stack for ZigBee”. The main part of the program is the network traffic monitor. To access this, click on the “ZigBee(TM) Tools” menu and then select “Network traffic monitor”.

In order to get the Analyser working correctly, select channel 26 (0x1A), or the channel used by the Flowcode ZigBee component. If you are enabling multiple channels in the Flowcode ZigBee component, you will have to let ZENA monitor the channels one at a time, or use the ZigBee LCD to print the current operating channel.




Notice the Start and Stop buttons and the “Packet Sniffer” screen. The Start and Stop buttons control the network analysis. The Output Window displays all the transmissions that have been sent via the ZigBee network and displays the MAC ID, header data, time and other information about each transmission. This makes troubleshooting much easier and is an aid in the understanding of ZigBee and other wireless networks.

The next diagram illustrates typical output from the ZENA analyser.




### 3.2.4 Testing the ZigBee system

Set up the ZigBee system and switch on power to the four nodes. Connect the ZENA ZigBee analyser to the computer. Open the Test folder on the CD and download the set of hex files to the appropriate nodes using PPP (installed as part of Flowcode)as follows:

- Start up PPP (Start →All Programs →Matrix Multimedia →PPP v3 →PPP v3).
- Go to File → Open.
- Navigate to the test folder.
- Open the hex file for the first node. (The file name matches that of the node).
- Check the Configure chip options are set to XTAL and Watchdog Off.
- Click on the SEND TO PIC button to program the microcontroller.
- Repeat the process for the other nodes.
    - Node 1 – Coordinator and display node
    - Node 2 – Sensors node A
    - Node 3 – Router and gateway node
    - Node 4 – Sensors node B

Once the four nodes have been programmed you can test Node 1 and Node 4 by moving the sensor Pot and watching for a corresponding change on the Node 1 display. Nodes 2 and 3 can be checked by pressing D0 on Node 2 and watching for a signal on LED D0 on Node 3.

## 4. The ZigBee assignments

The assignments included with the ZigBee training solution lead up to developing a four node ZigBee network in a step-by-step manner. Further features are added to the network at each stage. The end product is a fully operational, secure, dynamic, fire and burglar alarm system.

Here is an overview of that process..

Exercise 1:Moulding the network

- Configuring the co-ordinator node;
- Searching for an unused channel;
- Assigning a network ID;
- Starting up the ZigBee network.

Exercise 2:Adding a node

- Connecting an end device node to the network;
- Communicating on the network;
- Measuring transmission signal strength;
- Controlling the node join time.

Exercise 3:Expanding the network

- Connecting a router node and a second end device node to the network;
- Modifying message destinations.

Exercise 4:Reducing power consumption

- Configuring sleep mode for end devices;
- Configuring router and co-ordinator to buffer packets sent to sleeping nodes.

Exercise 5:Dynamic networks

- Creating and maintaining a list of active nodes connected to the network.

Exercise 6:Message routing

- Scanning for the best routes between nodes using broadcast packets.

Exercise 7:Data logging gateway

- Streaming ZigBee network messages via the router node and USB232 connection into a PC for control, logging or analysis.

Exercise 8:Modular fire and burglar alarm

- Building a fire and burglar alarm system capable of sensing new nodes and displaying node interaction.

Exercise 9:Improving network security

- Adding an encryption layer to prevent hackers from gaining access to the network without locking out any possible future nodes.

# 5. Exercise 1 – Moulding the network

## 5.1 Introduction

The aim of first exercise is to set up a ZigBee personal area network using the co-ordinator node and then to detect it using the Microchip ZENA ZigBee analyser.

The ZigBee E-Block uses a Version 2 XBee ZigBee controller module. A TTL level RS232 bus is used to communicate with the controller module, using a baud rate of 9600, 8 data bits, 1 start bit, 1 stop bit, no parity and hardware flow control.

Information sent and received is stored in a series of buffers, which are part of the Matrix Multimedia implementation of the ZigBee physical layer and not part of the ZigBee specification itself.

## 5.2 Objective

The objective of this exercise is to write a Flowcode program to create a ZigBee network that will allow other ZigBee devices to connect to it.

## 5.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C
- a Graphical LCD E-Block (EB043) connected to Port D
- a Microchip ZENA ZigBee analyser.

## 5.4 The Flowcode program in detail

The program will:

- assign properties to the ZigBee Flowcode component;
- configure the co-ordinator mode with:
  - a preset PAN ID;
  - a device name of Coord;
  - an infinite join time;
  - the channel set to 1A;
  - a scan duration of 3;
  - a maximum hop count of 4;
  - verbose mode enabled.
- display the current active channel on the graphical LCD ;
- display the signals transmitted by the co-ordinator, using the ZENA analyser.

### 5.4.1 Target PICmicro device

The course was written using the PIC16F877A device. Other microcontroller devices can be used instead, but the settings, programs and instructions may need to be adapted to match the new device.

The ZigBee E-Block is connected to the PIC16F877A on port C as this port contains the UART controller.

### 5.4.2 Flowcode ZigBee component

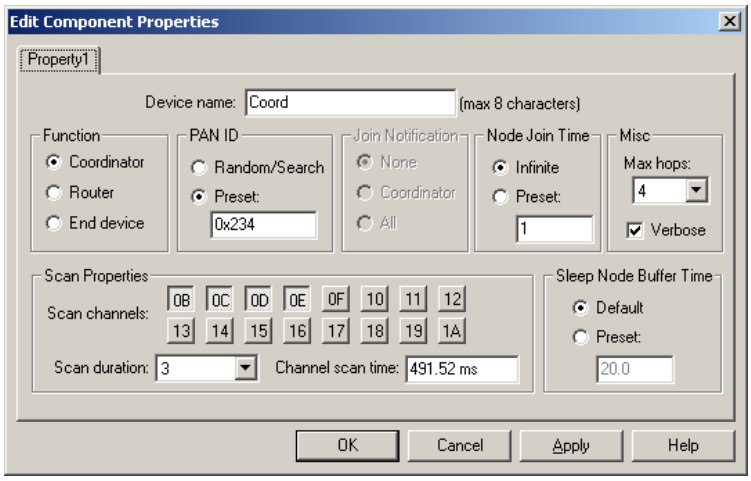
The Flowcode ZigBee component is added to a Flowcode program by clicking the ZigBee icon, shown opposite, found in the component toolbox.

Once the ZigBee component has been added to the program, the component window appears.


### 5.4.3 ZigBee component settings

Click on the down arrow in the title bar of the ZigBee Component to access the following dialogue boxes:

**Component Properties dialogue box:**
The ZigBee component properties are grouped into several categories as shown in the following diagram:




Here is an overview of what each category configures. Further details will be given as the course develops.

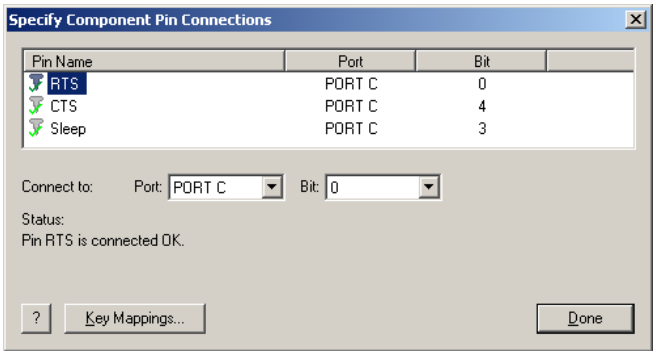
- Function – sets the mode in which the module operates.
- PAN ID – is either preset or obtained by scanning for an unused Pan ID.
- Scan Properties – configures which RF channels to scan, and for how long.
- Join Notification – specifies who to inform when connecting to the network.
- Join Time – sets the time over which other nodes will be permitted to join.
- Sleep Time – sets the time for which end nodes will sleep and parents will buffer data.
- Device name – allocates a character based (name) address.
- Misc – sets the maximum number of repeat broadcast transmissions and can enable verbose LCD logging.

**Component Connections dialogue box:**

In addition, the component connections may need to be configured. These are shown in the next diagram.

If the RTS and CTS pins are left disconnected then hardware flow control for the UART is automatically disabled.

The sleep pin is used to control the pin sleep mode available to the module.




**Hardware configuration settings:**

Other settings are shown in the following diagrams.

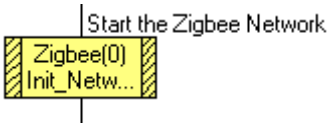
| Flowcode/PPP configuration settings | |
|---|---|
| Target | PIC16F877A |
| Clock Speed | 19.6608 MHz |
| Clock Mode | HS |
| Watchdog | Disabled |
| LVP | Disabled |
| All Other Options | Disabled |

| Development board/E-Block settings | |
|---|---|
| XT/RC | XT |
| Fast / Slow | Don't Care |
| PSU / USB | PSU |
| USB / ICD2 | USB |
| Device | PIC16F877A |
| Crystal Speed | 19.6608MHz |

| ZigBee board settings (PIC16F877A) | |
|---|---|
| Jumper J4 | C |
| Jumper J5 | 1 |
| Jumper J8 | +5V |

### 5.4.4 Init_Network function

The ZigBee component needs to be initialised before it is used in a program.
To do this, place the ZigBee Init_Network macro in the program before any other ZigBee macros are used, ideally right at the start of the program.




The Init_Network macro configures the ZigBee module to the profile specified in the component properties and component connections by sending appropriate AT commands.

**Note: If the ZigBee component has verbose mode enabled, then the Graphical LCD module must be initialised before running the ZigBee init macro.**

### 5.4.5 Configuring the co-ordinator

The co-ordinator node is responsible for creating the ZigBee network and assigning addresses to ZigBee nodes on the network.

Here are some of the AT commands for the coordinator that will be used in this exercise.

- All parameters sent with the AT commands are in hexadecimal format except for the character based address which is in ASCII character format.
- All AT commands are terminated with a carriage return, ASCII code 13.
- The Send AT command function is used to send out an AT command in the form of a string. Using this function, the carriage return is added to the end of the AT command string automatically.

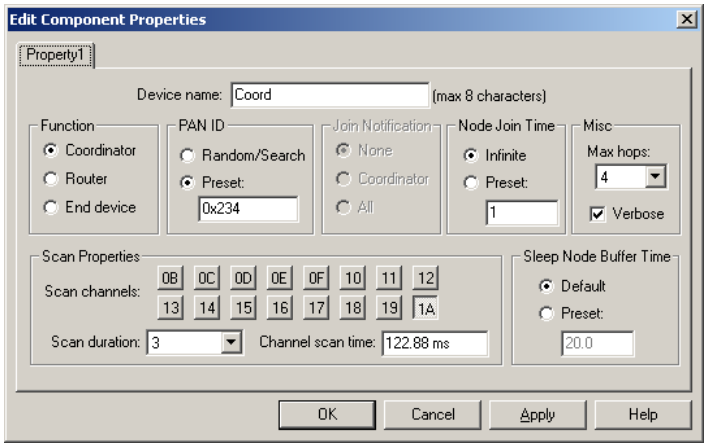
| | | | |
|---|---|---|---|
| ATID | XXXX | – | Assigns the PAN ID XXXX. |
| ATNI | XXXXXXXX | – | Assigns the character based node address XXXXXXXX. |
| ATSC | XXXX | – | Assigns the RF channel XXXX to be scannned. |
| ATSD | X | – | Sets the RF channel scan duration to X. |

The ZigBee Init_Network and other macros handle all of the actual AT commands that control the ZigBee module. The AT commands and responses are displayed on the LCD module when verbose mode is enabled.

## 5.5 What to do

1. Write the Flowcode program using the following steps as a guide:

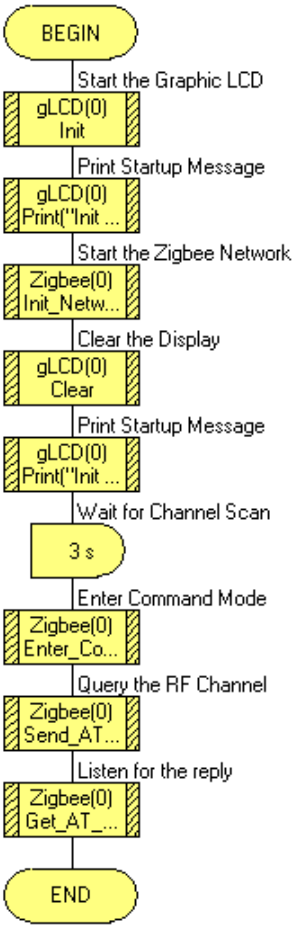
1. Load the ZigBee component into a new Flowcode flowchart;
2. Configure the ZigBee component properties as shown in the next diagram;



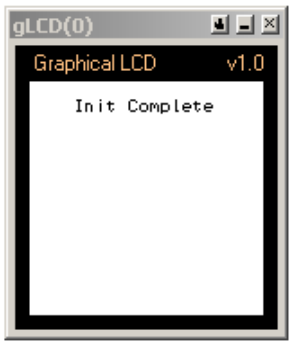

3. Load the Graphical LCD component into the flowchart;
4. Insert a Component Macro, and select the gLCD(0) component and the Init macro to initialise the Graphical LCD component;
5. Give the macro the Display Name “Start the Graphic LCD”;
6. Insert another Component Macro, and select the gLCD(0) component and the Print macro;
7. Insert the following in the Parameter box: "Init Started", 25, 10, 0, 0;
8. Give the macro the Display Name “Print Startup Message”;
9. Insert a Component Macro, and select the Zigbee0) component and the Init_Network macro to initialise the ZigBee co-ordinator;
10. Give the macro the Display Name “Start the ZigBee Network”;
11. Insert another Component Macro, and select the gLCD(0) component and the Clear macro to clear the Graphical LCD;
12. Give the macro the Display Name “Clear the Display”;
13. Insert another Component Macro, and select the gLCD(0) component and the Print macro;
14. Insert the following in the Parameter box: "Init Complete", 25, 10, 0, 0;
15. Give the macro the Display Name “Print Startup Message”;
16. Add a delay icon, set to 3 seconds with the Display Name “Wait for Channel Scan”;
17. Insert a Component Macro, and select the Zigbee0) component and the Enter_Command_Mode macro;
18. Give the macro the Display Name “Enter Command Mode”;
19. Insert a Component Macro, and select the Zigbee0) component and the Send_AT_Command macro;
20. Insert the following in the Parameter box:"ATCH"
21. Give the macro the Display Name “Query the RF Channel”;
22. Insert a Component Macro, and select the ZigBee(0) component and the Get_AT_Response macro;
23. Insert the number 0 as the logging parameter ;
24. Give the macro the Display Name “Listen for the Reply”;
25. End the program.

2. Compile the program and transfer it to the PIC chip on node 1.
3. Run and test the program by observing the LCD to see the interim messages while initialisation takes place, and the final channel display.
4. Do not delete this program as it can be modified for use in later exercises!

The resulting Flowcode program is shown in the next diagram.







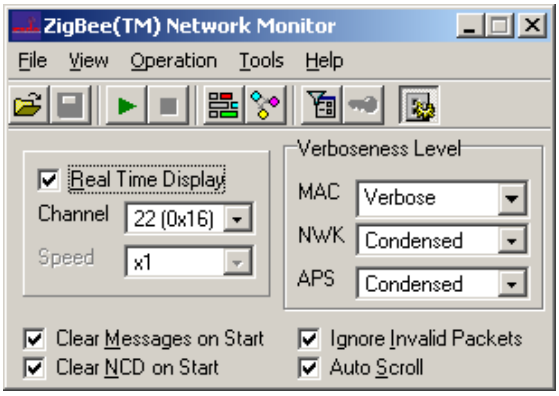

### 5.5.1 Analysing the network with ZENA

To monitor the network with the ZENA analyser, plug it into a spare USB port on the computer and then click on Start -> Microchip software stack for ZigBee™ -> ZENA.


Open up the Network Traffic Monitor by clicking on the ZigBee Tools menu.

Next select the appropriate channel for your ZigBee network. This correct channel is the number that is shown on the graphical LCD at the end of the exercise 1.

Also set the MAC verboseness level to verbose and then click the Start button.




ZENA should now capture a signal sent out regularly from the ZigBee co-ordinator node similar to that shown below.

| Frame | Time(us) | Len | MAC Frame Control | | | | | Seq Num | Dest PAN | Dest Addr | Source Addr | NWK Frame Control | | | | Dest Addr | Source Addr | Radius | Seq Num |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | +1651712 | | Type | Sec | Pend | ACK | IPAN | | | | | Type | Ver | Route | Sec | | | | |
| 00001 | =1651712 | 25 | DATA | N | N | N | Y | 0x37 | 0x0234 | 0xFFFF | 0x0000 | DAT | 0x1 | SUP | N | 0xFFFF | 0x0000 | 0x01 | 0x6D |

Interpretation:

**Dest PAN**
0x0234

The Personal Area Network Identifier.

**Dest Addr**
0xFFFF

Destination address
(0xFFFF means broadcast to all nodes on the PAN)

**Source Addr**
0x0000

Source address
(0x0000 is always the address of the co-ordinator node)

## 5.6 Further work

- Increase the number of channels that are scanned during start-up.
- Does the co-ordinator always pick the same channel?
- What happens if you disable this channel?
- What happens if you increased the scan times?

# 6. Exercise 2 – Adding a Node

## 6.1 Introduction

In this exercise, the plan is to add an end device node onto the ZigBee network and see if it can communicate with the co-ordinator node.

## 6.2 Objective

The objective of this exercise is to write a Flowcode program to create a functional two-node ZigBee network that will allow data to be sent and received

## 6.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Microchip ZENA ZigBee analyser.

## 6.4 The Flowcode program in detail

For node 1, the program will:

- configure the co-ordinator node as in exercise 1;
- wait for a new device to join the network;
- wait to receive a character sent from the keypad attached to the new node;
- monitor and display the signal strength of the received transmission;
- display the current activity on the graphical LCD ;

For node 2 the program will:

- configure the second node as an end device
- connect to the network;
- display the current status of the node on the LED board as it joins the network ;
- wait until a key on the keypad is pressed and then transmit the character;

For both nodes, the ZENA analyser will be used to display the signals transmitted by the co-ordinator, using the ZENA analyser.

### 6.4.1 End device node

The end device node is assigned an address by the co-ordinator as it joins the network. For that to happen, the end device must be configured with settings compatible to those of the co-ordinator. In particular, the PAN ID and RF scan channel parameters must be identical to those of the co-ordinator.

The end device node is configured in the same way as the co-ordinator node. Instead of the Graphical LCD E-Block, the LED E-Block will display the status of the operation.

- LED D0 will light up at the start of the program to show that the microcontroller is running.
- LED D1 will light up when the ZigBee initialisation routine is complete.
- LED D2 will light up if the node has successfully found a co-ordinator node.
- LEDs D0 and D1 will both light up if no co-ordinator node can be found.

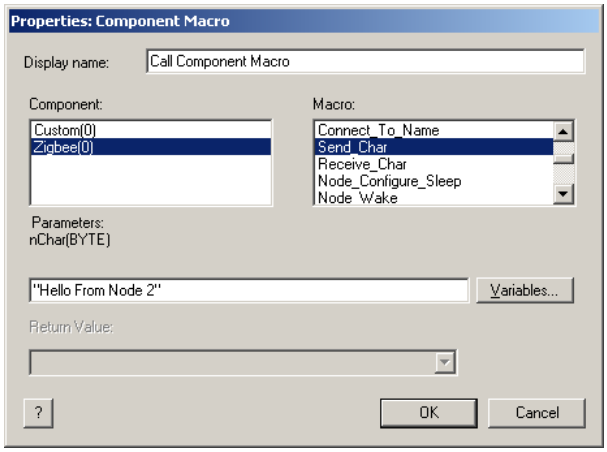
Here are some of the AT commands for the end device node that will be used in this exercise:

| | | |
|---|---|---|
| ATDN XXXXXXXX | - | Assigns the character based destination node. |
| ATJN X | - | Configures the network join notifications. |
| ATDB | - | Collects the last received signal strength |

Again, in Flowcode, most of these AT commands are handled by the ZigBee macros.

### 6.4.2 Talking on the network

Once the nodes are connected to each other as a ZigBee network, the Send_Char and Receive_Char macros can be used to communicate across the network.




Both macros can handle single characters, character strings or data contained in variables. If sending single characters, then use single commas around it, e.g.'1'.
If sending a character string, enclose it in double commas, e.g. "Hello From Node 2".
If sending data as a variable, then enter the variable name directly, e.g. keypad_num.

The ZigBee coordinator will send the ASCII code '1' to represent a node poll command (an "Is anyone there?" command). The end device will then respond with ASCII code '2' to represent a node present response (i.e. "Yes").

Once it has been established that both nodes can communicate correctly, node 2 will stream data from the keypad to the LCD on node 1.

### 6.4.3 Specifying the destination for the message

Since it is so easy to send and receive data on the ZigBee network it is useful to know which device or devices the node wishes to talk to. This is done by using the Flowcode ZigBee component Connect_To macros, which include:

- Connect_To_Coordinator;
- Connect_To_Address (to connect to a specific network address);
- Connect_To_Name (to connect to a specific name address);
- Connect_To_All

If the node manages to connect successfully, then the Connect_To macro will return value zero. Otherwise it will return a value of one, if the remote node cannot be found.

### 6.4.4 Monitoring the signal strength

In exercise 1, node 1 used verbose mode to check that the Init_Network macro completed correctly.

In this exercise, that is still a useful function, but then the LCD is needed to display the received signal strength. There needs to be a method of disabling the LCD verbose mode mid-program. This is done by using a C code icon to run the following line of code.

```
LCD_VERB_ZIG = 0;        //Disable LCD Verbose Mode
```

To re-enable the LCD verbose mode during program execution, the C code icon needs to run the following line of code..

```
LCD_VERB_ZIG = 1;        //Re-enable LCD Verbose Mode
```

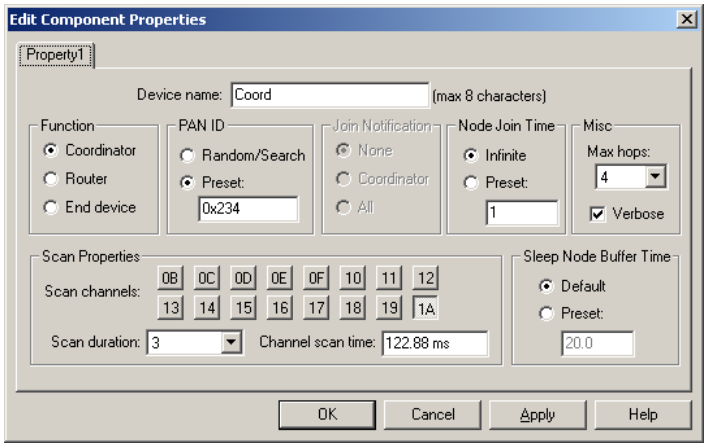
(Note - For this to work, the LCD verbose mode must have been enabled in the Flowcode ZigBee component properties.)

## 6.5 What to do

**For Node 1:**

1. Write the Flowcode program using the following steps as a guide:

*1.* Load the ZigBee component into a new Flowcode flowchart;
*2.* Configure the ZigBee component properties as shown in the next diagram;




*3.* Load the Graphical LCD component into the flowchart;
*4.* Insert a Component Macro, and select the gLCD(0) component and the Init macro to initialise the Graphical LCD component;
*5.* Give the macro the Display Name “Start the Graphic LCD”;

6. Insert a Component Macro, and select the Zigbee0) component and the Init_Network macro to initialise the ZigBee co-ordinator;
7. Give the macro the Display Name “Start the ZigBee Network”;
8. Insert another Component Macro, and select the gLCD(0) component and the Clear macro to clear the Graphical LCD;
9. Give the macro the Display Name “Clear the Display”;
10. Insert another Component Macro, and select the gLCD(0) component and the Print macro;
11. Insert the following in the Parameter box: "Waiting for Node 2", 2, 2, 0, 0;
12. Give the macro the Display Name “Print Waiting for Node 2";
13. Insert a Calculation icon, create a variable called Devices and set the value of Devices to zero.
14. Add a loop having the Display Name “While nodes not connected”, tested at the start of the loop, with the looping condition “Devices < 1”;
15. Insert a Component Macro, and select the Zigbee0) component and the Send_Char macro with a parameter of ‘1’ (to see whether node 2 is active,);
16. Give the macro the Display Name “Poll for Node 2”;
17. Add a delay icon, set to 100 milliseconds with the Display Name “Delay”;
18. Insert a Component Macro, and select the Zigbee0) component and the Receive_Char macro with a nTimeout(BYTE) parameter of 200;
19. Create a new variable “Incoming”, and select it for use as the Return Value;
20. Give the macro the Display Name “Check for Response”;
21. Insert a Decision box, called “Response received?”, with the condition ‘Incoming = 2’ (meaning that the second node has replied,);
22. On its ‘Yes’ loop add a Calculation icon, to carry out the calculation ‘Devices = Devices + 1’ and a name ‘Increment device count’;
23. Its ‘No’ arm continues to the next icon, which is the end of the “While nodes not connected” loop ;
24. Insert another Component Macro, and select the gLCD(0) component and the Clear macro to clear the Graphical LCD;
25. Give the macro the Display Name “Clear the Display”;
26. Insert another Component Macro, and select the gLCD(0) component and the Print macro;
27. Insert the following in the Parameter box: "Node 2 Connected", 2, 2, 0, 0;
28. Give the macro the Display Name “Print Node Connected";
29. Insert a C Code icon, called “Disable Verbose LCD Messages”;
30. Add the code LCD_VERB_ZIG = 0; in the C Code dialogue box;
31. Add another loop called ‘Main’, tested at the start of the loop, with a Loop While value of 1 (i.e. forever,);
32. Insert a Component Macro, and select the Zigbee0) component and the Receive_Char macro with a nTimeout(BYTE) parameter of 20;
33. Use the variable “Incoming” as the Return Value;
34. Give the macro the Display Name “Receive Keypad data”;
35. Insert a Decision box, called “If Data Received?”, with the condition ‘Incoming <255’ (meaning that data has arrived,);
36. On its ‘Yes’ loop add a String Manipulation icon, called ‘Convert Keypad number to String’, to carry out the function ‘String = "" + Incoming’ ;
37. Then insert a Component Macro, and select the gLCD(0) component and the Print macro;
38. Insert the following in the Parameter box: ‘ String, 60, 60, 2, 0’;
39. Give the macro the Display Name “Print Keypad Data to LCD";

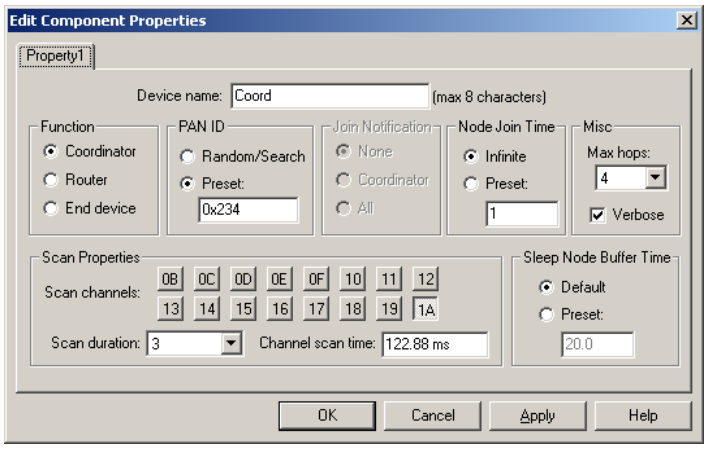
40. Its ‘No’ arm continues to the next icon;
41. Insert a Component Macro, and select the Zigbee0) component and the Get_Signal_Level macro with a nTimeout(BYTE) parameter of 20;
42. Create a new variable called sig_lev and use it as the Return Value;
43. Give the macro the Display Name “Read the last Signal Level”;
44. Add a String Manipulation icon, called ‘Convert Signal Level to String’, to carry out the function ‘String = ToString$(sig_lev)’;
45. Insert a Component Macro, and select the gLCD(0) component and the Print macro;
46. Insert the following in the Parameter box: "Signal Level", 2, 12, 0, 0;
47. Give the macro the Display Name “Print Signal Level";
48. Insert a Component Macro, and select the gLCD(0) component and the Print macro;
49. Insert the following in the Parameter box: "  ", 100, 12, 0, 0;
50. Give the macro the Display Name “Overwrite the Previous Value with Spaces";
51. Insert a Component Macro, and select the gLCD(0) component and the Print macro;
52. Insert the following in the Parameter box: String, 100, 12, 0, 0;
53. Give the macro the Display Name “Print Signal Level Data";
54. The next icon is the end of the Main loop;
55. That is followed by the End icon.

2. Compile the program and transfer it to the PIC chip on node 1.
3. Press the reset button on the microcontroller on node 1.

**For Node 2:**

1. Write the Flowcode program using the following steps as a guide:

1. Load the ZigBee component into a new Flowcode flowchart;
2. Configure the ZigBee component properties as shown in the next diagram;




3. Insert an Output icon, called ‘Program Started’ which outputs the number 1 to all of Port B (the LED E-Block);
4. Insert a Component Macro, and select the Zigbee0) component and the Init_Network macro to initialise the ZigBee co-ordinator;
5. Give the macro the Display Name “Start the ZigBee Network”;
6. Insert another Output icon, called ‘Init Complete which outputs the number 2 to all of Port B;
7. Insert a Component Macro, and select the Zigbee0) component and the Connect_To_Coordinator macro;

8. Create a new variable called retval and use it as the Return Value;
9. Give the macro the Display Name “Connect to Coordinator”;
10. Insert a Decision box, called “If Problem Connecting”, with the condition ‘retval’;
11. On its ‘Yes’ loop add an Output icon, called ‘Coordinator not found’ which outputs the number 3 to all of Port B;
12. On its ‘No’ arm add an Output icon, called ‘Coordinator Connected which outputs the number 4 to all of Port B;
13. Start a loop called ‘Main’, tested at the start of the loop, with a Loop While value of 1 (i.e. forever,);
14. Insert a Component Macro, and select the Zigbee0) component and the Receive_Char macro with a nTimeout(BYTE) parameter of 2;
15. Create a new variable “in”, and select it for use as the Return Value;
16. Give the macro the Display Name “Check for node poll command”;
17. Insert a Decision box, called “If command received”, with the condition in = ‘1’;
18. On its ‘Yes’ loop insert a Component Macro, and select the Zigbee0) component and the Send_Char macro with a parameter of ‘2’;
19. Give the macro the Display Name “Send node present command”;
20. Is ‘No’ arm continues to the next icon;
21. Insert a Component Macro, and select the Keypad(0) component and the GetKeypadAscii macro;
22. Create a new variable “keypad_num”, and select it for use as the Return Value
23. Give the macro the Display Name “Scan for Keypad press”;
24. Insert a Decision box, called “If Keypad pressed”, with the condition keypad_num < ‘255’;
25. On its ‘Yes’ loop insert a Component Macro, and select the Zigbee0) component and the Send_Char macro with a nChar(BYTE) parameter of keypad_num;
26. Give the macro the Display Name “Transmit keypress”;
27. Add a delay icon, called Delay, set to a value of 100 milliseconds;
28. Its ‘No’ arm continues to the next icon which is the end of the Main loop;
29. That is followed by the End icon.

2. Compile the program and transfer it to the PIC chip on node 2.
3. Press the reset button on the microcontroller on node 1.
4. Run and test the program by:
   - observing the LEDs and waiting until LED D2 lights, to show that the end device is connected to the network;
   - then watch for the “Node 2 connected” messages on the LCD;
   - then press one of the keys on the keypad;
   - watch for its value displayed on the LCD.
5. Do not delete these programs as they can be modified for use in later exercises!

### 6.5.1 Analysing the network with ZENA

To monitor the network with the ZENA analyser, connect it to the PC and run the software in the same way as in exercise 1.

The output should show the transmission of data between the end node and the co-ordinator.

- When the keypad on node 2 is pressed the data is send through from the end device to the co-ordinator.
- That is followed by the automatic minor acknowledgement, and then by a message received acknowledgement from the co-ordinator to the end node.
- This again generates the automatic minor acknowledgement to confirm reception of the message received acknowledgement.

| Frame | Time(us) | Len | MAC Frame Control | Seq Num | Dest PAN | Dest Addr | Source Addr | NWK Header | APS Header | APS Payload | FCS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | +1645232 | | | | | | | 0x44 0x00 0x00 0x00 | 0x00 0xE8 0x11 | 0xE1 0x96 0x00 0x6D 0x03 0x0A 0x40 0x00 0xA2 0x13 | RSSI Corr CRC |
| 00001 | =1645232 | 45 | 0x8861 | 0x05 | 0x0234 | 0x0000 | 0xD875 | 0x75 0xD8 0x0F 0xA9 | 0x05 0xC1 0xE8 | 0x00 0x00 0x13 0xA2 0x00 0x40 0x0A 0x04 0x77 0x39 | 0xF6 0x6A 1 |

| Frame | Time(us) | Len | MAC Frame Control | Seq Num | FCS |
|---|---|---|---|---|---|
| | +2560 | | | | RSSI Corr CRC |
| 00002 | =1647792 | 5 | 0x0002 | 0x05 | 0x00 0x6C 1 |

| Frame | Time(us) | Len | MAC Frame Control | Seq Num | Dest PAN | Dest Addr | Source Addr | NWK Header | APS Header | APS Payload | FCS |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | +2704 | | | | | | | 0x04 0x00 0x75 0xD8 | 0x00 0xE8 0x00 | 0xE1 0x96 | RSSI Corr CRC |
| 00003 | =1650496 | 28 | 0x8861 | 0xEC | 0x0234 | 0xD875 | 0x0000 | 0x00 0x00 0x0F 0x4D | 0x05 0xC1 0xE8 | 0x04 | 0x00 0x6B 1 |

| Frame | Time(us) | Len | MAC Frame Control | Seq Num | FCS |
|---|---|---|---|---|---|
| | +1680 | | | | RSSI Corr CRC |
| 00004 | =1652176 | 5 | 0x0002 | 0xEC | 0xF6 0x68 1 |

## 6.6 Further work

- How would you scan for more then one node?
- Does the receiving node know which node has communicated with it?
- How does range affect the received signal level?

# 7 Exercise 3 – Expanding the network

## 7.1 Introduction

In this exercise, the plan is to add two more nodes onto the ZigBee network and to see how communication takes place across the new network. As there is no simple means of displaying the status of nodes 3 and 4 on the nodes themselves, we will use the co-ordinator node (node 1) to monitor the status of the other nodes.

## 7.2 Objective

The objective of this exercise is to write Flowcode programs to add two more nodes, a router node and another end device, to the two-node ZigBee network, by modifying and extending the Flowcode programs set up in exercise 2. A message on the LCD display will indicate that all nodes have joined successfully.

## 7.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Multiprogrammer with a microcontroller device to control node 3
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 3
- a LED E-Block (EB007) connected to Port D of node 3
- a Multiprogrammer with a microcontroller device to control node 4
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 4
- a Microchip ZENA ZigBee analyser.

## 7.4 The Flowcode program in detail

For node 1, the program will:

- configure the co-ordinator node as in exercise 2;
- attempt to connect to all other nodes;
- wait for new devices to join the network;
- display the name of each new device that joins the network;
- activate the new nodes.

For node 2 the program will:

- configure the second node as an end device;
- connect to co-ordinator in an attempt to join the network;
- display the current status of the process on the LED board as it joins the network ;
- transmit confirmation to the co-ordinator when it has been successful in joining the network;
- wait to receive a transmission from node 3, and use the data in it to light the corresponding LEDs on the LED E-Block .

For node 3 the program will:

- configure the node as an end device;
- connect to the co-ordinator in an attempt to join the network;
- transmit a variable indicating the state of the switches on the Switch E-Block to node 2 .

For node 4 the program will:

- configure the node as a router;
- connect to the co-ordinator in an attempt to join the network;
- transmit confirmation to the co-ordinator when it has been successful in joining the network.

For all four nodes, the ZENA analyser will be used to display the signals transmitted by the co-ordinator, using the ZENA analyser.

### 7.4.1 Configuring the router node

The router node, like the co-ordinator, is capable of forwarding messages to remote nodes. It is also similar in that it too can act as a parent to sleeping end device nodes.

It is configured in the same way as the co-ordinator and end device nodes, by selecting properties for the Flowcode ZigBee component, and then to using the Flowcode Init_Network macro to transfer these choices to the device itself.

Here are two AT commands for the router node that will be used in this exercise.

ATSP X - sets the time X to buffer packets for sleeping children.
ATNJ X - sets the time X during which nodes can join the network as children of this node.

Again most of the AT commands are handled by the Flowcode ZigBee macros.

### 7.4.2 Parent/child association

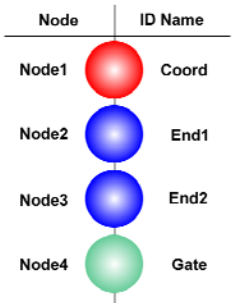
Each node that joins the network by connecting directly to another node is referred to as a child of that node. Both co-ordinator and router nodes, but not end devices, are capable of being parent devices. A single parent node can have up to eight associated child nodes. For more than eight child nodes one of the children must itself be a router.

For example a router node has eight children, seven of the children are end device nodes but one of the children is a router node. The child router node can have its own set of eight children. In this way, large networks can be built up.

### 7.4.3 Talking to specific nodes

In this exercise, each node in the network is assigned a name, given opposite.

The naming is arbitrary but it is recommended that these names are used in the exercise, to maintain compatibility with the course notes.



## 7.5 What to do

1. The programs will check for a node poll command (ASCII '1'), and on receipt will return the correct response (ASCII '2' or '3'). To ensure that the responses come back to the co-ordinator in the correct order, a delay has been added to the programs running on node 3 and node 4.

   In the case of nodes 1 and 2, the programs developed in exercise 2 can be modified.

   For node 1:
   - Initialise the LCD display;
   - Initialise the ZigBee component;
   - Command the co-ordinator to connect to all devices;
   - Clear the LCD;
   - Disable verbose messaging;
   - Print the message "Waiting for nodes"
   - Set the value of the Devices variable to zero;
   - Create a loop that continues while the value of Devices is less than 7;
   - Set the value of the Devices variable to zero again;
   - Poll for nodes by transmitting the character '1';
   - Clear any nodes displayed on the LCD by printing spaces in those locations;
   - Create a loop that will run 150 times;
   - Use the Receive_Char macro to check for incoming messages;
   - Use a Decision icon to see if the response is from node 2 (Incoming = 2?);
   - If so, print "2 – OK" to the LCD;
   - In the same way, check if a response has been received from nodes 3 and 4, and print appropriate messages;
   - Return to the 150 loop;
   - When this is completed, return to the loop while Devices < 7;
   - Clear the LCD;
   - Print the message "Nodes connected";
   - Activate the other nodes by transmitting the character '0'.

   For node 2:
   - Light LED D0 to show that the attempt to join the network has started;
   - Initialise the ZigBee component;
   - Light LED D1 to show that the initialisation is complete;
   - Command the node to connect to the co-ordinator;
   - Use the same process, a Decision box looking at the variable retval, to light both LED D0 and D1 if the co-ordinator is not found, or LED D2 if it has connected to the co-ordinator;
   - Set the value of the variable 'in' to zero;
   - Create a Node Check loop that continues if the value of 'in' is not zero;
   - Check for the poll node command, as in exercise 2, using the variable 'in';
   - When this is received (i.e. 'in' = 1,) transmit the 'node present' command, by sending back the character '2';
   - That completes the Node Check loop;
   - Switch off all LEDs;
   - Create an infinite loop that looks for new received values for the variable 'switch_num';
   - Display its value on the LED E-Block;

- Loop back and keep repeating this check.

For node 3 (an end device):

- Initialise the ZigBee component;
- Create the variables ‘switch_num’, ‘old_switch_num, and ‘in’ and set all values to zero;
- Command the node to connect to the co-ordinator;
- Create a Main Loop that continues if the value of ‘in’ is not zero;
- Use the Receive_Char macro to look for the node poll command, using the variable ‘in’;
- When ‘in’ = ‘1’, delay for 5 milliseconds and then send the ‘node present’ command by sending back the character ‘2’;
- Loop back to the Main Loop;
- Use the Connect_To_Name macro to connect to the device called “End1”;
- Create an infinite loop that looks for new values for the variable ‘switch_num’, by reading Port D;
- If the switch state has changed (switch_num != old_switch_num), transmit the new switch value;
- Add a 50ms delay;
- Update the value of old_switch_num by using a Calculation icon with ‘old_switch_num = switch_num’;
- Loop back to the infinite loop..

For node 4 (a router):

- Initialise the ZigBee component;
- Command the node to connect to the co-ordinator;
- Create an infinite loop that uses the Receive_Char macro to look for the node poll command, using the variable ‘in’;
- When ‘in’ = ‘1’, delay for 10 milliseconds and then send the ‘node present’ command by sending back the character ‘2’;
- Loop back and keep repeating this check.

2. Compile the program for node 1 and transfer it to the appropriate node.
3. Press reset on the node 1 multiprogrammer and, the ZigBee co-ordinator should start up the ZigBee network. Eventually, the graphical LCD should display the message “Waiting for Nodes”.
4. Compile the program for node 2 and transfer it to the appropriate node.
5. Press reset on the node 2 multiprogrammer and the ZigBee end device should join the network. The graphical LCD on Node 1 should then display the message “2-OK”.
6. Repeat the above steps for the other two remaining nodes.
7. Run and test the program by observing the LCD to see the interim messages and the final ‘Nodes Connected’ message.
8. Change the settings of the switches on the switch E-Block, and observe the LEDs to see whether the changes are transmitted to this node.
9. Do not delete these programs as they can be modified for use in later exercises!

### 7.5.1 Using ZENA to analyse End device to end device communication

Once the four nodes have successfully established communications with each other, we can begin monitoring the exchanges between the two end device nodes.

The diagram shows typical output from the analyser. As can be seen, each message passes along the chain via the router or via the coordinator node. Each time the message is received by the next node in the chain, an acknowledgement is transmitted. Once the message has reached its destination the message acknowledgement is bounced back along the chain with minor acknowledges being sent out by all the intermediate nodes.

| Frame | Time(us) | Len | MAC Frame Control | Seq Num | Dest PAN | Dest Addr | Source Addr | |
|---|---|---|---|---|---|---|---|---|
| 00626 | +83904<br>=75166112 | 56 | 0x8841 | 0xC7 | 0x0234 | 0xFFFF | 0x0000 | **Coordinator (0x000) Sends '1'** |
| 00627 | +5536<br>=75171648 | 56 | 0x8841 | 0x2F | 0x0234 | 0xFFFF | 0x1B2D | **Node 2 (0x1B2D) Sends '2'** |
| 00628 | +15280<br>=75186928 | 56 | 0x8841 | 0x16 | 0x0234 | 0xFFFF | 0x579C | **Node 3 (0x579C) Sends '3'** |
| 00629 | +26240<br>=75213168 | 56 | 0x8841 | 0xB0 | 0x0234 | 0xFFFF | 0x0B52 | **Node 4 (0x0B52) Sends '4'** |

## 7.6 Further work

- Once all nodes have been detected, try turning all 8 LEDs on.
- Can you think of a reason why this does not work correctly?
- Can you think of a way to get around the problem?

# 8 Exercise 4 – Reducing power consumption

## 8.1 Introduction

Scenario - node 2 is a hand-held battery-powered device with a set of 8 LEDs representing 8 specific conditions. It has a keypad for entering data. To lengthen the battery life the device will be placed in sleep mode whenever the it is not being used.

The solution will be modelled by combining the programs from exercises 2 and 3.

## 8.2 Objective

The objective of this exercise is to reduce the power consumption of the end device nodes by introducing sleep mode operation. This also introduces the concept of packet buffering. When end device sleep mode is combined with microcontroller sleep mode, the overall power saving can be significant. However in this exercise only the sleeping operation of the XBee device will be explored.

A small ZigBee network will be created that will allow for the sleep operation to be analysed.

## 8.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Multiprogrammer with a microcontroller device to control node 3
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 3
- a LED E-Block (EB007) connected to Port D of node 3
- a Multiprogrammer with a microcontroller device to control node 4
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 4
- a Microchip ZENA ZigBee analyser.

## 8.4 The Flowcode program in detail

For node 1, the program will:

- configure the co-ordinator node as in exercise 2;
- attempt to connect to all other nodes;
- wait for new devices to join the network;
- display the name of each new device that joins the network;
- activate the new nodes.

For node 2, the program will:

- configure the second node as an end device;
- use the LEDs to display the stages of joining the network;
- enter sleep mode, waking periodically to check for transmissions;
- use the LEDs to display the state of the switches on node 3.

For node 3 the program will:

- configure the node as an end device;
- connect to the co-ordinator in an attempt to join the network;
- transmit a variable indicating the state of the E-Block switches to node 2 .

For node 4 the program will:

- configure the node as a router;
- connect to the co-ordinator in an attempt to join the network;
- transmit confirmation to the co-ordinator when it has been successful in joining the network.

For all four nodes, the ZENA analyser will be used to display the signals transmitted by the co-ordinator, using the ZENA analyser.

### 8.4.1 Sleep modes

The ZigBee XBee modules are capable of two distinct types of sleep operation. One, pin sleep, is controlled directly from a digital input pin connected to the microcontroller and the other, cyclic sleep, is controlled by a low power timer.

The following table summarises the sleep modes used by the Node_Configure_Sleep macro.

| Mode | Timeout | Operation | Wake Up Time | Power Demand |
|---|---|---|---|---|
| 0 | N/A | Sleep Disabled | N/A | < 50mA |
| 1 | N/A | Pin Hibernate | 13.2ms | < 10uA |
| 2 | N/A | Pin Doze | 2ms | < 50uA |
| 3 | N/A | Reserved (do not use) | 2ms | N/A |
| 4 | Time in ms of inactivity to wait before going into sleep mode | Cyclic Sleep | 2ms | < 50uA |
| 5 | | Cyclic Sleep with Pin Wake Up | 2ms | < 50uA |

### 8.4.2 Packet buffering

When a node is asleep, it cannot communicate with, or be contacted by the network.

If a parent node receives a message for a sleeping child node, it will acknowledge receipt of the message, and then it will buffer the message for a specific length of time. This time is defined in the properties of the Flowcode ZigBee component.

When a sleeping node wakes up, the first thing that it must do is to check whether its parent node has any buffered messages. If the child node does not check for messages within the timeout period then the message will be discarded.

This means that to maintain guaranteed delivery, the child nodes must poll their parents before a timeout can occur. In practise this is achieved by assigning the same sleep timeout to all nodes on the network.

### 8.4.3 Polling for buffered data

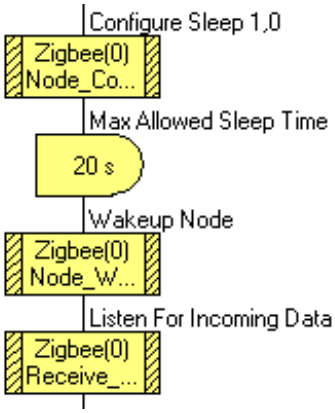
Data is collected from the parent node by issuing the 'force poll' AT command.

The Flowcode Node_Wake macro is responsible for waking a node up from pin sleep and / or issuing the 'force poll' command.

A node in cyclic sleep must be left for the correct length of time before issuing the Node_Wake command. Otherwise it will remain in sleep mode and any buffered messages will not be collected.

After issuing the 'force poll' command, any incoming messages can be collected by using the standard Receive_Char macro.

The diagram shows code for a typical pin sleep operation cycle.




## 8.5 What to do

The aim is to design a program that will allow sleep operation to be used alongside the keypad and LED controller functionality. If the keypad on node 2 is pressed then node 2 is woken up immediately and the switch data is read before the keypad data is sent.

The programs designed in exercise 3 can be modified so that node 2 now goes to sleep in between receiving commands from node 3.

For node 1:

1. Use exactly the same program as in exercise 3.
2. Compile the program and transfer it to node1.
3. Press reset on the node 1 multiprogrammer and, the ZigBee co-ordinator should start up the network. Eventually, the graphical LCD should display the message "Waiting for Nodes".

For node 2:

1. Use exactly the same program as in exercise 3 up to the point where all the LEDs are switched off. Then:
   - Create an infinite loop;
   - Create a second loop that runs 50 times;
   - Use a Receive_Char macro to look for new received values for the variable 'in';
   - If there is a new value, display it on the LED E-Block;
   - Delay for 100 milliseconds;
   - Loop back to the 50 times loop;
   - Insert a Component Macro that calls the ZigBee(0) Node_Configure_Sleep macro, with Mode(BYTE), Timeout(INT) parameters of 2, 0;
   - Add a 5 second delay;
   - Insert a Component Macro that calls the ZigBee(0) Node_Wake macro;
   - Loop back to the infinite loop
2. Compile the program and transfer it to node2.
3. Press reset on the node 2 multiprogrammer. Once the network has been established node 2 will enter its sleep mode. Every 5 seconds node 2 will wake up and poll for any incoming data. The incoming data is provided from the switches connected to node 3. If the keypad connected to node 2 is pressed then the node will wake up immediately to send out the keypad data and retrieve the LED data.

For node 3:

1. Use exactly the same program as in exercise 3.
2. Compile the program and transfer it to node 3.
3. Press reset on the node 3 multiprogrammer.

For node 4:

1. Use exactly the same program as in exercise 3.
2. Compile the program and transfer it to node 4.
3. Press reset on the node 4 multiprogrammer.

## 8.6 Further work

- Node 3 is also an end device node that is capable of sleeping. Try to implement a pin sleep operation on this device whilst keeping the nodes functionality running correctly.
- Next try to implement a cyclic sleep operation on the node 2 end device.

# 9 Exercise 5 – Dynamic networks

## 9.1 Introduction

For this fifth exercise, the programs will be designed so that the network becomes more dynamic. Nodes will come and go on the network as needed.

## 9.2 Objective

The objective of this exercise is to select nodes based on their name, which, in turn, can be chosen to reflect their function, or the peripheral devices connected to them.

A small ZigBee network will be created that will allow for the dynamic operation to be analysed.

## 9.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Multiprogrammer with a microcontroller device to control node 3
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 3
- a LED E-Block (EB007) connected to Port D of node 3
- a Multiprogrammer with a microcontroller device to control node 4
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 4
- a Microchip ZENA ZigBee analyser.

## 9.4 The Flowcode program in detail

For node 1, the program will:

- configure the co-ordinator node as in exercise 2;
- establish the network the network;
- scan for nodes whose name starts with the string “END”;
- will display a list of nodes connected to the network, and highlight those whose name starts with “END”.

For node 2, the program will:

- configure the second node as an end device;
- display the stages of joining the network on the LEDs;
- enter sleep mode, waking periodically to check for transmissions;
- use the LEDs to display the state of the switches on node 3.

For node 3 the program will:

- configure the node as an end device;
- connect to the co-ordinator in an attempt to join the network;
- transmit a variable indicating the state of the E-Block switches to node 2 .

For node 4 the program will:

- configure the node as a router;
- connect to the co-ordinator in an attempt to join the network;
- transmit confirmation to the co-ordinator when it has been successful in joining the network.

For all four nodes, the ZENA analyser will be used to display the signals transmitted by the co-ordinator, using the ZENA analyser.

### 9.4.1 MAC addresses

Each ZigBee node has a unique and fixed 64-bit MAC address. These addresses can be used to secure a network of known devices. Nodes will not be able to join the network unless they have an authorised MAC address. One disadvantage of this approach is that networks generally will need to alter over time. Secondly, the MAC addressing scheme is ‘flat’. There is no hierarchy indicated by it, and so it provides no information about what or where the node is connected to.

### 9.4.2 Identifier addresses

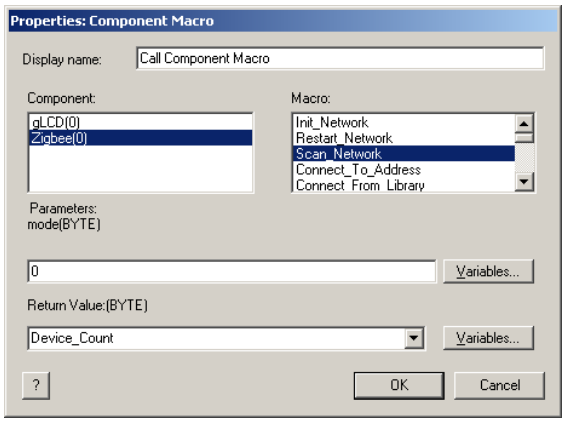
An identifier address is a character string of up to 8 characters. It could be just an arbitrary name or it could indicate the functionality of the node

For example, a burglar alarm system may contain a large number of PIR sensors. If these nodes were given names starting with the string PIR, (and so were named PIR1, PIR15, etc,) then by scanning for the string “PIR” in the identifier address, all of these nodes could be contacted and controlled.

This means that exactly the same program could be used for all of these nodes, apart from the need to complete the devices’ identifier addresses as no two devices can have identical addresses.

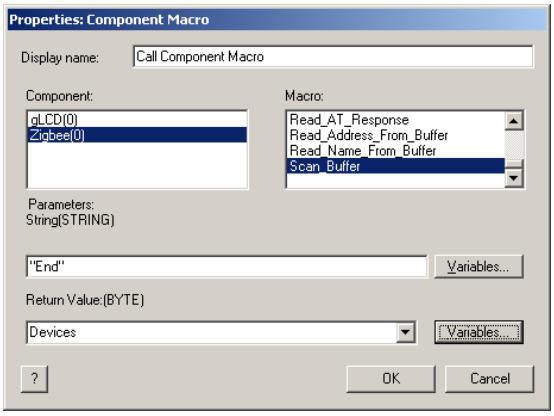
### 9.4.3 Polling for nodes

The Flowcode Scan_Network macro retrieves up to eight identifier addresses or network addresses. This maximum of eight addresses is the result of a limitation on the number that can be held in the microcontroller memory, which, in turn, stems from the format of the Matrix Multimedia ZigBee driver. (This can be upgraded for any nodes that need to scan more then eight devices.)

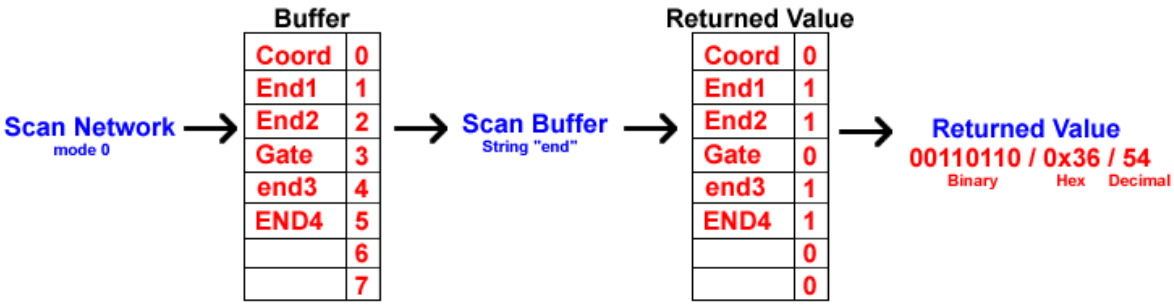



The Scan_Network macro has a mode parameter that controls whether a detected device will be identified by its (numerical) address (mode1) or by its identifier address (mode 0). In the diagram above, the node will scan the network and store the identifier addresses of detected devices in a variable named Device_Count.

Once the scan is complete, there will be a local list of up to eight identifier addresses of nodes connected to the network. This list can then be analysed by the Flowcode Scan_Buffer macro. This enables the list of identifier addresses to be compared to a string constant. The result of this comparison is returned as a binary number, representing the resulting buffer values.




The next diagram shows how this binary number relates to the buffer values. Notice that the comparison is not case sensitive, and so does not distinguish between “END” and “eNd”. However, the comparison string must be at the beginning of the identifier to be correctly identified. For example a search using a parameter “the” would find the identifier “theory” and “them” but not the identifier “another”.




In this case, the list on the left is the result of scanning the network in mode 0. Running the Scan_Buffer macro, with a comparison string “end” modifies the second column of the table, as shown. That column now contains value 1 when the identifier starts with “end”, and value 0 if it does not. The Return Value of this macro is a read-out of the second column, from bottom to top. Hence in this case, the Return Value is 00110110. Now it is simply a case of scanning through the Return Value one bit at a time to identify the nodes that match the scan string.

## 9.5 What to do

The aim is to create a Node 1 co-ordinator program that will scan for any device identifier addresses beginning with the string “END”. The program then prints a list of the detected device identifiers and uses the scan return value to colour the names that match the searched string.

For node 1:

1. Write a Flowcode program using the following steps as a guide:
    1. Initialise the LCD display;
    2. Initialise the ZigBee component;
    3. Print the message “Init Complete”;
    4. Add a two second delay;
    5. Disable verbose messaging;
    6. Start an infinite loop, called ‘Main Program Loop’;
    7. Clear the LCD;
    8. Print the message “Scanning for nodes”;
    9. Add a Scan_Network Component macro, called ‘Scan for nodes’ using mode 0, and using the variable Device_Count as the Return Value;
    10. Print the message “Scan complete”;
    11. Print the message “Nodes detected” in position X2, Y15;
    12. Add a gLCD(0) PrintNumber Component macro, called ‘Printnumber nodes detected’ in position X110, Y15, using Device_Count as the number;
    13. Add a Scan_Buffer Component macro, called ‘Scan for “END” in ID address’ using “END” as the comparison string, and length 3 bytes, with the variable Devices as the Return Value;
    14. Print the message “End Devices” in position X2, Y28;
    15. Add a gLCD(0) PrintNumber Component macro, called ‘Printnumber end devices’ in position X110, Y28, using Devices as the number;
    16. Create a variable called idx, and set its value to zero using a Calculation icon called ‘Initialise index’ ;
    17. Create a loop called ‘Scan through Device Table’ that continues while the value of idx is less than 8;
    18. Create a variable called Flag and add a calculation icon called ‘Mask one device at a time’ to perform the calculation Flag = Devices & 0x01;
    19. Add a Decision icon to test if the variable Flag is true;
    20. In the ‘Yes’ loop, add a gLCD(0) SetForeColor macro, with parameters 7,0,0 for red, green and blue;

21. In the ‘No’ loop, add a gLCD(0) SetForeColor macro, with parameters 0,0,0 for red, green and blue;
22. Create a variable called idx2, and set its value to zero using a Calculation icon called ‘Initialise index 2’;
23. Create a loop called ‘Scan through Device Name’ that continues while the value of idx2 is less than 8;
24. Add a Read_Name_From_Buffer Component macro, called ‘Read name from buffer’ using idx and idx2 as device and index parameters respectively, and String[idx2] as the Return Value;
25. Add a Calculation icon called ‘Increment index 2’ to increment idx2;
26. End the loop ‘Scan through Device Name’;
27. Create a variable called Ycoord and add a Calculation icon called ‘Calculate Y Coord for LCD’ to carry out the calculation Ycoord = ( idx * 10 ) + 40;
28. Add a gLCD(0) PrintNumber Component macro, called ‘Print device index’ in position X2, Y= Ycoord, using idx as the number;
29. Add a gLCD(0) PrintNumber Component macro, called ‘Print device flag’ in position X10, Y= Ycoord, using Flag as the number;
30. Print the message “Print Device ID Name” in position X20, Y=Ycoord, using String as the string parameter;
31. Add a Calculation icon called ‘Move to next Device’ to increment the variable Devices;
32. Add a Calculation icon called ‘Increment index’ to increment idx;
33. End the loop ‘Scan through Device Table;
34. Add a ten second delay;
35. End the loop Main Program Loop
36. End the program.

2. Compile the program and transfer it to node1.
3. Press reset on the node 1 multiprogrammer and, the ZigBee co-ordinator should start up the ZigBee network. Eventually, the graphical LCD should display the table of devices starting with the string “END”.

For node 2:

1. Use the same program as in **exercise 3** modified as follows:
   - In the ‘Yes’ loop of the ‘If command received’ Decision box, add a delay of 100 milliseconds after the ‘Send node present’ command.
2. Compile the program and transfer it to node2.
3. Press reset on the node 2 multiprogrammer.

For node 3:

1. Use the same program as in exercise 4, modified as follows:
   - In the ‘Yes’ loop of the ‘If command received’ Decision box, increase the delay, before the ‘Send node present command’ to 100 milliseconds, and add another delay of 100 milliseconds after that command.
   - Move the ‘Connect to End Device 1’ Component macro into the ‘Yes’ loop of the ‘If switch state changed’ Decision box, before the ‘Transmit switch value’ Component macro.
2. Compile the program and transfer it to node 3.
3. Press reset on the node 3 multiprogrammer.

For node 4:

1. Use the same program as in exercise 3 or 4, modified as follows:
   - In the ‘Yes’ loop of the ‘If command received’ Decision box, increase the delay, before the ‘Send node present command’ to 200 milliseconds, and add another delay of 100 milliseconds after that command.
2. Compile the program and transfer it to node 4.
3. Press reset on the node 4 multiprogrammer.

## 9.6 Further work

- Node 4 has an identifier name beginning with “GATE”. Perform a scan to determine the position of this node in the address buffer.
- Once you have determined the position, use the Connect_From_Library component macro to connect to the device.

# 10 Exercise 6 – Message Routing

## 10.1 Introduction

This exercise looks at programs designed to route data along specific paths through the network. It will also add an address layer to the data so that receiving node knows where incoming messages originated and where they are destined for.

The problems to be addressed occur in situations where, for example, a sensor node has to report an event to both an alarm node and a logging gateway node. This raises questions like:
Which node will receive the notification first?
How can we ensure that both nodes receive the signal?

## 10.2 Objective

The objective of this exercise is to create a transmission chain so that all data will end up at the same place. This, coupled with the techniques developed in the previous exercise, will allow ZigBee network data to be exported to a specific device or another network.

A small ZigBee network will be created that will allow for the transmission chain operation to be analysed.

## 10.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Multiprogrammer with a microcontroller device to control node 3
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 3
- a LED E-Block (EB007) connected to Port D of node 3
- a Multiprogrammer with a microcontroller device to control node 4
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 4
- a Microchip ZENA ZigBee analyser.

## 10.4 The Flowcode program in detail

For node 1, the program will:

- configure the co-ordinator node;
- set up the ZigBee network;
- display the source network ID, and two data bytes, for each transmission received from nodes 2 and 3;
- check that the parity value is correct for each of these transmissions, and display any incorrect values in red.

For node 2, the program will:

- configure the second node as an end device;
- display the stages of joining the network on the LEDs;
- locate and identify the gateway node;
- sample the potentiometer reading periodically and convert the value to a digital number;
- display this value on the LEDs;
- calculate a parity value for each sample;
- transmit a four-byte message containing the node address, data and parity value to the gateway node.

For node 3 the program will:

- configure the second node as an end device;
- locate and identify the gateway node;
- sample the potentiometer reading periodically and convert the value to a digital number;
- calculate a parity value for each sample;
- transmit a four-byte message containing the node address, data and parity value to the gateway node.

For node 4 the program will:

- configure the node as a router;
- connect to the co-ordinator in an attempt to join the network;
- check for messages from nodes 2 and node 3;
- re-transmit these messages to node 1.

For all four nodes, the ZENA analyser will be used to display the signals transmitted by the co-ordinator, using the ZENA analyser.

### 10.4.1 Route calibration

Specifying the pathway that data follows ensures that data is not missed as it travels across the network. In the case of the sensor reporting to both a logging gateway and an alarm, the solution is to send the data from the end device nodes straight to the gateway node, so that it is logged, and from there to the endpoint alarm node. This makes it impossible for the alarm to be triggered without it first being logged. This increases accountability and reliability for the system.

### 10.4.2 Mobile nodes and beacon signals

The task of ensuring that a signal reaches specific nodes becomes even harder when those nodes are mobile within the network. A strategy for routing messages should avoid such problems of mobility or of signal attenuation that may result. The best strategy is always to send the signal directly to the device that must respond first.

In the present exercise, node 4 will be contacted first as it contains the gateway device that logs the alarm condition. After that, either the originating node or node 4 can retransmit the message. For this exercise, node 4 will repeat any incoming messages to node 1.

### 10.4.3 Message Structure

One element to include in a message structure is the identification of the source node which originated the message. This is particularly useful for applications such as fire alarms, where fire-fighters need to know which zone has triggered the alarm, in order to locate the fire quickly.

Here is the proposed message structure for the current exercise.

- Each message will consist of four data bytes.
- The first data byte denotes the origin of the message, (i.e. a specific node on the network.)
- The next two bytes are data bytes that contain usable sensor data.
- The final byte is a parity byte to make sure that all four bytes of data came from the same node. In this case, the parity byte is calculated by adding together the node number byte and the data bytes, and then dividing the total value by 3, i.e. shows the average value of the first three bytes.The table illustrates this idea.

| Node Number Byte | Data Byte 1 | Data Byte 2 | Parity Byte |
|---|---|---|---|
| 03 | 200 | 12 | (3 + 200 + 12) / 3 = 71 |
| 02 | 56 | 87 | (2 + 56 + 87) / 3 = 48 |

## 10.5 What to do

The aim is to create three programs:

- a program, to run on nodes 2 and 3, that will locate the gateway node and transmit data from the potentiometers on the analogue sensor boards, using the messaging structure shown above.
- a program written for node 4, that will constantly look for messages, run the parity check on messages it receives, and then forward them to node 1.
- a program for node 1, which checks the validity of the messages and outputs them to the graphical LCD. Any invalid messages arriving at node 1 should be flagged on the LCD by modifying the colour of the text.

The program structures are complex and so should be copied from the Matrix Multimedia ZigBee course CD-ROM.

- Download the Exercise 6 programs into the correct nodes;
- Allow the network to boot up.
- Once the network has been established nodes 2 and 3 will begin by searching for a device with the name identifier “GATE”.
- Once each of these has found the correct address for the gateway, it begins sampling the voltage on the potentiometer connected to analogue channel AN1.
- After each sample, a packet will be generated and transmitted to node 4, the gateway node.
- Once the gateway node has received a complete transmission, it forwards the packet to node 1.
- Node 1 uses the parity byte to check the transmission for errors. Any incorrect parity bytes are highlighted in red on the graphical LCD.

## 10.6 Further work

- On nodes 2 and 3, use the second data byte to send the data from the LDR on the sensor board. The LDR is connected to analogue channel AN0.
- At the moment, only two bytes of data can be sent packaged with the two bytes of overhead (address and parity bytes). This makes the system only 50% efficient, since only two in every four bytes sent over the network is valid data. How could you increase this efficiency? How could you send more then two data bytes without having to resend the address and parity information?
- At the moment, the parity byte does not show actual parity, but carries the average of the data bytes. Write a program to calculate the actual parity bit for the data bytes.

  (Use odd parity, which means that the parity bit value is chosen so that the total number of logic 1s in the data and parity bit combined is an odd number. For example, if the data bytes were 10011000 and 01111000, there is a total of seven logic 1s. Seven is an odd number so the parity bit will be a logic 0, keeping the total number of logic 1s as an odd number.)

# 11 Exercise 7 – Datalogging Gateway

## 11.1 Introduction

This exercise will examine the use of the router's USB232 board as a means of exporting and storing the data as it travels over the network.

## 11.2 Objective

The objective of this exercise is to take the data from a transmission chain and feed it over the gateway connection into a PC. This completes the process of exporting the ZigBee network data onto another type of device or network. A small ZigBee network will be created that will allow for the gateway operation to be analysed.

## 11.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Multiprogrammer with a microcontroller device to control node 3
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 3
- a LED E-Block (EB007) connected to Port D of node 3
- a Multiprogrammer with a microcontroller device to control node 4
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 4
- a USB232 E-Block (EB039) connected to Port D of node 4
- a Microchip ZENA ZigBee analyser.

## 11.4 The Flowcode program in detail

For node 1, the program will:

- configure the co-ordinator node;
- set up the ZigBee network;
- display the source network ID, and two data bytes, for each transmission received from nodes 2 and 3;
- will check that the parity value correct for each of these transmissions, and display any incorrect values in red.

For node 2, the program will:

- configure the second node as an end device;
- display the stages of joining the network on the LEDs;
- locate and identify the gateway node;
- sample the potentiometer reading periodically and convert the value to a digital number;
- display this value on the LEDs;
- calculate a parity value for each sample;
- transmit a four-byte message containing the node address, data and parity value to the gateway node.

For node 3 the program will:

- configure the second node as an end device;
- locate and identify the gateway node;
- sample the potentiometer reading periodically and convert the value to a digital number;
- calculate a parity value for each sample;
- transmit a four-byte message containing the node address, data and parity value to the gateway node.

For node 4 the program will:

- configure the node as a router;
- connect to the co-ordinator in an attempt to join the network;
- check for messages from nodes 2 and node 3;
- re-transmit these messages to node 1.
- convert the data into a form compatible with HyperTerminal so that the data can be relayed to a computer.

For all four nodes, the ZENA analyser will be used to display the signals transmitted by the co-ordinator, using the ZENA analyser.

### 11.4.1 Collecting the data

Each end device node now identifies and sends its data to node 4 the Router node. This means that all messages propagating through the network will arrive at the Router node. They can then be forwarded to the USB gateway.

Thereafter, the messages are forwarded to node 1, the co-ordinator node, where they can be displayed.

### 11.4.2 Logging the data

The data is logged by the PC as it comes in on the USB emulated COM port. To monitor this we will use the Windows communications software program, HyperTerminal.

HyperTerminal will provide a snapshot of what is happening on the network but will not be able to log the data over a period of time. Other COM port data logging software will log directly to a text or csv file to allow data to be archived.

### 11.4.3 Bytes and ASCII

Up to this point, data has travelled over the ZigBee network in the form of byte variables. These are each 8 bits long and so can represent $2^8$ or 256 different values of pure numbers, ranging from 0 to 255.

ASCII data has a different meaning. Using the ASCII standard, each digit or character is translated into a byte of data. For example, the letter 'A' is represented by the decimal number 65, whereas 'a' is represented by the number 97. A full list of ASCII characters can be found at http://www.asciitable.com/

Here, then, are two distinct varieties of data, both represented by eight bit numbers. The first is the pure number form that has been used in the exercises so far and the other relates to a character or control command.

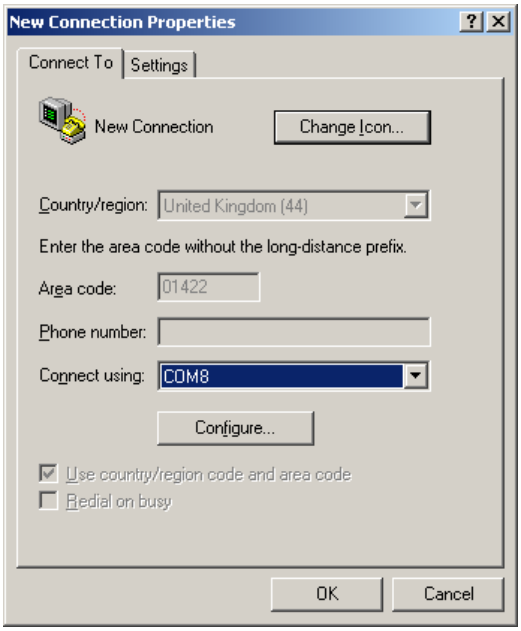
HyperTerminal accepts bytes in ASCII form so there needs to be a way of converting from pure number format to a form that can be used in HyperTerminal. This is the function of a Flowcode macro named "BYTE_2_NUMBER", which can be imported into a Flowcode program to this conversion to take place.

### 11.4.4 Connecting the USB232 board

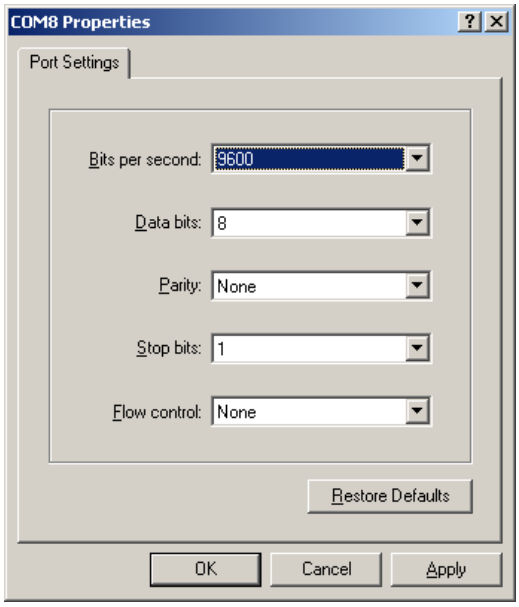
The drivers for the USB232 board can be found on the ZigBee course CD or can be downloaded from the Matrix Multimedia website. When the USB232 board is plugged in and installed, the associated COM port can be found by looking in the Device Manager under the heading Ports (COM & LPT).


That done, HyperTerminal can be configured to use this hardware by following the steps outlined below:

1. Open the HyperTerminal application, by going to Start / All Programs / Accessories / Communications / HyperTerminal.
2. Give the new connection a name, such as 'ZigBee'.
3. Select the COM port associated with the USB232 board from the list given after 'Connect using'.
4. Open the properties page by clicking the 'Configure...' button located underneath the COM port selection box.




5. Use the settings shown in the next diagram to configure the connection, and then click on the 'OK' button.




6. The HyperTerminal console window will then open.

### 11.4.5 Bit Banging Serial Communications

In order to talk to the computer via the USB232 board, a serial communications port must be configured and initialised. The microcontrollers used in the ZigBee nodes have such a serial communications port, using hardware to do the pin toggling automatically. Unfortunately, this is occupied in sending and receiving data to and from the ZigBee boards.

This means that the USB232 board has to be driven directly, using software to toggle the associated pin output levels. This is known as 'bit banging'.

The task is made easier because node 4 only has to send data to the computer and does not have to deal with data coming back from it.

Each byte of data sent over the serial connection is sandwiched between two extra bits, a start bit ( a logic 0) and a stop bit (a logic 1).

A Flowcode macro, named “NUMBER_2_USB232”, created to simplify this bit banging process, can be imported into Flowcode programs.

## 11.5 What to do

The aim is to modify the program running on node 4, from the previous exercise, so that after it has forwarded its packets to the co-ordinator node, it bit bangs the data to the USB232 E-Block.

The data is then in a form which allows it to be relayed to the computer by HyperTerminal, as well as being displayed on the graphical LCD. The latter will be used to make sure that the system is running as expected.

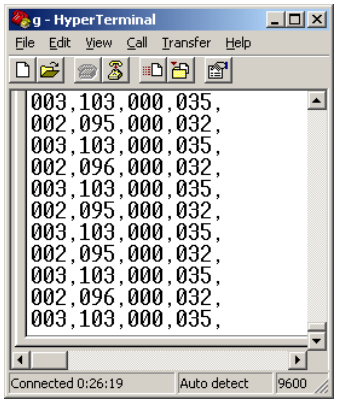
The programs for nodes 1, 2 and 3 are unchanged from exercise 6.

The program for node 4 is modified by the addition of:

- two Number_2_USB232 macros - one to send a Line Feed character (using a parameter of 10, the ASCII code for a Line Feed character,) and the other to send a Carriage Return ( with parameter 13, ASCII code for Carriage Return.)
- four Byte_2_Number macros – one to convert the source node number, two to convert the two bytes of data, and one to convert the parity byte.

This program should be copied from the ZigBee course CD-ROM and downloaded to the node 4 microcontroller.

- Allow the network to become established.
- Nodes 2 and 3 begin by searching for a device with the name identifier “GATE”.
- Then they begin sampling the potentiometer voltages and convertingthem to digital signals.
- These are then transmitted to node 4, the gateway node.
- This node then forwards the packet to node 1 and also sends the data to the computer via the bit banged USB232 connection, where it is visible on the HyperTerminal console screen, as illustrated below. The first column shows the identity of the source node. The second and third columns are the data bytes, while the last column gives the parity byte.




## 11.6 Further work

- Add a macro to read a byte from the bit banged USB232 connection.
- Use this to enable HyperTerminal to switch the PC gateway on and off. (For example, send an ASCII ‘1’ to switch on the gateway output and send an ASCII ‘0’ to switch the gateway off again.)

# 12 Exercise 8 – Modular fire and burglar alarm system

## 12.1 Introduction

This exercise develops a fully integrated fire and alarm system using dynamic nodes, structured routing and a USB gateway.

## 12.2 Objective

The objective of this exercise is to adapt the ZigBee network to perform a useful function. In doing so, the exercise will examine the basic elements of data capture as well as the more complex rule tables that will control the overall operation of the system.

A small ZigBee network will be created that will emulate a domestic combined fire and burglar alarm. This system will be modular and will demonstrate the advantages of using ZigBee over standard wired systems.

## 12.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Multiprogrammer with a microcontroller device to control node 3
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 3
- a LED E-Block (EB007) connected to Port D of node 3
- a Multiprogrammer with a microcontroller device to control node 4
- a ZigBee Co-ordinator E-Block (EB051R) connected to Port C of node 4
- a USB232 E-Block (EB039) connected to Port D of node 4
- a Microchip ZENA ZigBee analyser.

## 12.4 The Flowcode program in detail

Node 1 is used to start the ZigBee network and display data on the LCD.
The analogue sensor boards on Nodes 2 and 3 act as temperature sensors.
The Switch board on Node 3 acts as a PIR sensor
Node 4 is the gateway to receive all packets and then forward them to Node 1 and the computer.

Once the network has been established nodes 2 and 3 begin searching for a device with the name identifier “GATE”, and then sample the output of their on-board potentiometers. After each sample, a packet is transmitted to node 4, the gateway node, which will forward it to node 1 and to the computer via the bit banged USB232 connection.

A four digit code can be entered on the keypad connected to node 2 by pressing and holding the ‘#’ button until the LEDs all light up. When this happens, the ‘#’ key is released, and the code can be entered. A code of “1234” will set the alarm and a code of “4321” will deactivate the alarm.

### 12.4.1 Sensors and data inputs

The end device nodes (nodes 2 and 3) both have an analogue sensor E-Block connected to them equipped with a linear potentiometer. These will be used to emulate the temperature present at the end device nodes. Node 3 has a switch E-Block which will be used to represent a proximity infrared (PIR) sensor.

(Real PIR sensors could be connected to the end nodes via the analogue boards, though the program code would need modification in that case).

Each end node device will transmit a packet of data through to the gateway (router) node, from where it will be forwarded to the computer and to the alarm siren, situated on the co-ordinator node.

Each packet sent from an end node device consists of 5 bytes of data. The first byte is the node number and is used to track which node created the packet. The second and third bytes are sensor information detailing temperature level and PIR activity local to the node. The fourth byte is the control code byte, which controls several key aspects of the system, e.g. under what conditions the alarm is to sound. The final byte is the parity byte, used to make sure that the data received is the same as the data transmitted. The next diagram illustrates this packet structure.

| Node Number | Temperature Sensor Byte | PIR Sensor Byte | Control Code Byte | Parity Byte |
|---|---|---|---|---|
| 03 | 200 | 12 | 31 | (3 + 200 + 12 + 31) / 4 = 61 |
| 02 | 56 | 87 | 25 | (2 + 56 + 87 + 25) / 4 = 42 |
| 99 | 25 | 63 | 05 | (99 + 25 + 63 + 05) / 4 = 48 |

### 12.4.2 Data output

The output of the system will be available both via the graphical LCD and on a computer, via the USB HyperTerminal connection. The graphic LCD will act as the alarm or siren, and will also act as the central control point, where faults and data are displayed graphically.

The computer / USB HyperTerminal connection will be used to display the workings of the system in real time, as well as any errors that may occur while the system is active.

To aid in monitoring the system a macro named “String_2_USB232” has been added.to output a string directly to the bit banged USB port.

The diagram shows a typical output from the system. The data is arranged in five columns, following the packer structure outlined earlier.

```
002,096,000,000,024,
003,087,000,000,022,
002,234,000,128,091,Temp Too High,
003,239,000,128,092,Temp Too High,
002,080,000,000,020,
003,251,002,192,112,Temp Too High,PIR Triggered
002,079,000,000,020,
003,097,002,064,041,PIR Triggered
002,080,000,000,020,
```

### 12.4.3 Controlling the operation of the alarm

In a fully functioning alarm system, it should be possible to activate and deactivate the PIR sensors. This is the purpose of another macro, named “COLLECT_KEYCODE”, which reads a four digit number from the keypad attached to node 2.

The user enters this number by first pressing the ‘#’ button, and then the four number keys.

The program checks if the ‘#’ key is pressed every time it cycles through the main loop. If the ‘#’ button is pressed, then all of the LEDs on PortB will light to signify the start of the key code sequence. Once the ‘#’ button is released the LEDs on PortB will all switch off, These LEDs now show the number of the key as it is pressed.

The correct codes programmed into the system are “1234” to activate the alarm and “4321” to deactivate it.

## 12.5 What to do

The program structures are again complex and should be copied from the Matrix Multimedia ZigBee course CD-ROM.

- Download the Exercise 8 programs into the correct nodes;
- Allow the network to boot up.
- Once the network has been established nodes 2 and 3 will begin by searching for a device with the name identifier “GATE”.
- Once each of these has found the correct address for the gateway, it begins sampling the voltage on the potentiometer connected to analogue channel AN1.
- After each sample, a packet will be generated and transmitted to node 4, the gateway node.
- Once the gateway node has received a complete transmission, it forwards the packet to node 1 and to the computer via the bit banged USB232 connection.
- Node 1 uses the parity byte to check the transmission for errors. Any incorrect parity bytes are highlighted in red on the graphical LCD.
- A four digit code can be entered on the keypad connected to node 2 by pressing and holding the ‘#’ button until the LEDs all light up. Once this has happened the code can be entered. A code of “1234” will set the alarm and a code of “4321” will deactivate the alarm

## 12.6 Further work

- Add a function to node 1 to display a flashing icon when the alarm is triggered.
- Modify the program so that the same key code enables and then disables the alarm.

# 13 Exercise 9 – Improving network security

## 13.1 Introduction

This exercise focuses on the 128-bit AES encryption capability of the XBee modules.

It will demonstrate how easy it is to create a fully encrypted hacker-proof network, in which all data travelling between nodes is encrypted with a private key. Any nodes that do not have this key cannot communicate successfully with the other nodes.

The encryption key cannot be recovered from a module even if the module is accessed locally using AT commands. This makes the ZigBee network very secure and an ideal environment for control and embedded functions spanning a large area.

## 13.2 Objective

The objective of this exercise is to improve the security of the ZigBee network.

In the networks developed so far, there has been no restriction on which nodes can join the network. As a result, any data sent over these networks is open to scrutiny, modification and even removal.

For example, with the alarm system developed in exercise 8, a hacker may be able to join another node to the network, and use it to deactivate the alarm.

This exercise shows how to use the 128-bit AES encryption, a built in feature of the XBEE modules, tol prevent hackers from gaining access to the network, while still allowing future expansion by adding legitimate nodes.

## 13.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a Multiprogrammer with a microcontroller device to control node 1
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 1
- a Graphical LCD E-Block (EB043) connected to Port D of node 1
- a Multiprogrammer with a microcontroller device to control node 2
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 2
- a LED E-Block (EB004) connected to Port B of node 2
- a Keypad E-Block (EB014) connected to Port D of node 2
- a Multiprogrammer with a microcontroller device to control node 3
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 3
- a LED E-Block (EB007) connected to Port D of node 3
- a Multiprogrammer with a microcontroller device to control node 4
- a ZigBee Co-ordinator E-Block (EB051C) connected to Port C of node 4
- a USB232 E-Block (EB039) connected to Port D of node 4
- a Microchip ZENA ZigBee analyser.

## 13.4 The Flowcode program in detail

As in the last exercise:

- Node 1 is used to start the ZigBee network and display data on the LCD.
- Nodes 2 and 3 act as temperature sensors.
- Node 3 uses the Switch E-Block as a PIR sensor.
- The analogue sensor boards on Nodes 2 and 3 act as temperature sensors.
- Node 4 is the gateway to receive all packets and then forward them to Node 1 and the computer.
- A four digit code can be entered on the keypad connected to node 2 to activate and de-activate the alarm.

The programs used in exercise 8 are modified so that nodes 1, 2 and 4 use the 128-bit AES encryption protocol, with the same encryption key. Transmissions from node 3, however, are not encrypted

### 13.4.1 Encryption commands

All commands sent to the XBee ZigBee module from the microcontroller are in the form of AT commands. These commands comprise an ASCII string starting with the characters “AT”, (standing for attention), plus command characters plus data value characters.

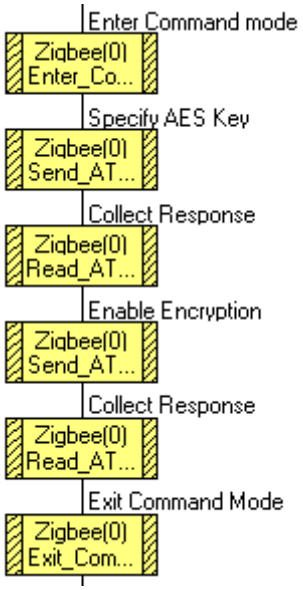
The following two AT commands are used to control the operation of the AES encryption onboard the XBee modules.

ATEE - Encryption Control ( 0 = off, 1 = on)
ATKY - AES Encryption Key from 0 to FFFFFFFFFFFFFFFF

For example the following command sequence configures the encryption key and then enables encryption:

Enter Command Mode
Send AT Command “ATKY FFAA55BBC3”
Get AT Response 0
Send AT Command “ATEE 1”
Get AT Response 0
Exit Command Mode

Shown below is the same code repeated in Flowcode ZigBee component macro form.



### 13.5 What to do

The programs should be copied from the Matrix Multimedia ZigBee course CD-ROM.

- Download the Exercise 9 programs into the correct nodes;
- Allow the network to boot up.
- The network functions in the same way as in Exercise 8 for nodes 1, 2 and 4.
- What effect does the unencrypted node 3 have on the system?

Congratulations – you have just completed the course!

# 14 Reference data

This document was designed to teach the basics of ZigBee, the theory and concepts behind it, and included practical exercises to increase knowledge and skill in working with ZigBee. But that's not the whole story of ZigBee. Should you wish to take your study of ZigBee further, or to move into areas of industry that use ZigBee, there is much more to know, and much more to study!

## IEEE 802.xx Networks

The complete range of IEEE 802 networking specifications can be found at the IEEE 802 standards committee website.
http://www.ieee802.org/

## ZigBee protocol

ZigBee is an evolving specification. It has already advanced from the original 2004 specification and is currently available as a reduced functionality version (ZigBee 2006) and a full functionality version (ZigBee Pro - 2007). The next release of the ZigBee stack specification is planned for release in 2008. The system represented here is a full functionality ZigBee Pro (2007) system.

The complete ZigBee specifications can be found at the ZigBee alliance
http://www.ZigBee.org/en/index.asp

## Acronyms and abbreviations

AES – Advanced Encryption Standard
AT – Attention command
MAC – Medium Access Control
PAN – Personal Area Network
RF – Radio Frequency
WPAN – Wireless Personal Area Network
ZENA – ZigBee Enabled Network Analyser

## Useful AT Commands

ATRE – Restores defaults into the non-volatile memory.
ATWR – Writes current parameter set into non-volatile memory.
ATCN – Exits the command mode.
ATFR – Performs software reset.
ATID – Sets/Reads the PAN ID of the network.
ATMY – Reads the 16-bit address of the module.
ATSH – Reads the high 32-bits of the MAC address.
ATSL – Reads the low 32-bits of the MAC address.
ATNI – Sets/Reads the ASCII node identifier.
ATDH – Sets/Reads the high 32-bits of the destination node's MAC address.
ATDL – Sets/Reads the low 32-bits of the destination node's MAC address.
ATDN – Collects the 16-bit address for the destination node's ASCII identifier.
ATDB – Reads the signal strength of the last reception.
ATND – Performs a node discovery broadcast.
ATSM – Sets/Reads the sleep mode.
ATSP – Sets/Reads the cyclic sleep period.
ATEE – Switches the 128-bit encryption on or off