

E BLOCKS2

ZigBee Communications



CP3924

MATRIX
www.matrixsl.com

Copyright © 2018 Matrix Technology Solutions Limited

CP3924

ZigBee Training Instructor Guide

Contents

About this course	4
Scheme of work	8
1. Introduction to ZigBee	8
2. The ZigBee Network	9
3. The Matrix ZigBee Training Solution	11
4. The ZigBee assignments	11
5. Exercise 1 – Moulding the network	12
6. Exercise 2 – Adding a Node	12
7. Exercise 3 - Expanding the network	13
8. Exercise 4 - Reducing power consumption	13
9. Exercise 5 – Dynamic networks	14
10. Exercise 6 – Message Routing	14
11. Exercise 7 – Datalogging Gateway.	15
12. Exercise 8 – Modular fire and burglar alarm system	16
13. Exercise 9 – Improving network security	17
Solutions to Exercises	18
Exercise 1	18
Exercise 2	20
Exercise 3	24
Exercise 4	30
Exercise 5	32
Exercise 6	37
Exercise 7	44
Exercise 8	51
Exercise 9	61
The OSI Model	64
The RS232 Protocol	65
AT commands	66
Further information	67

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:

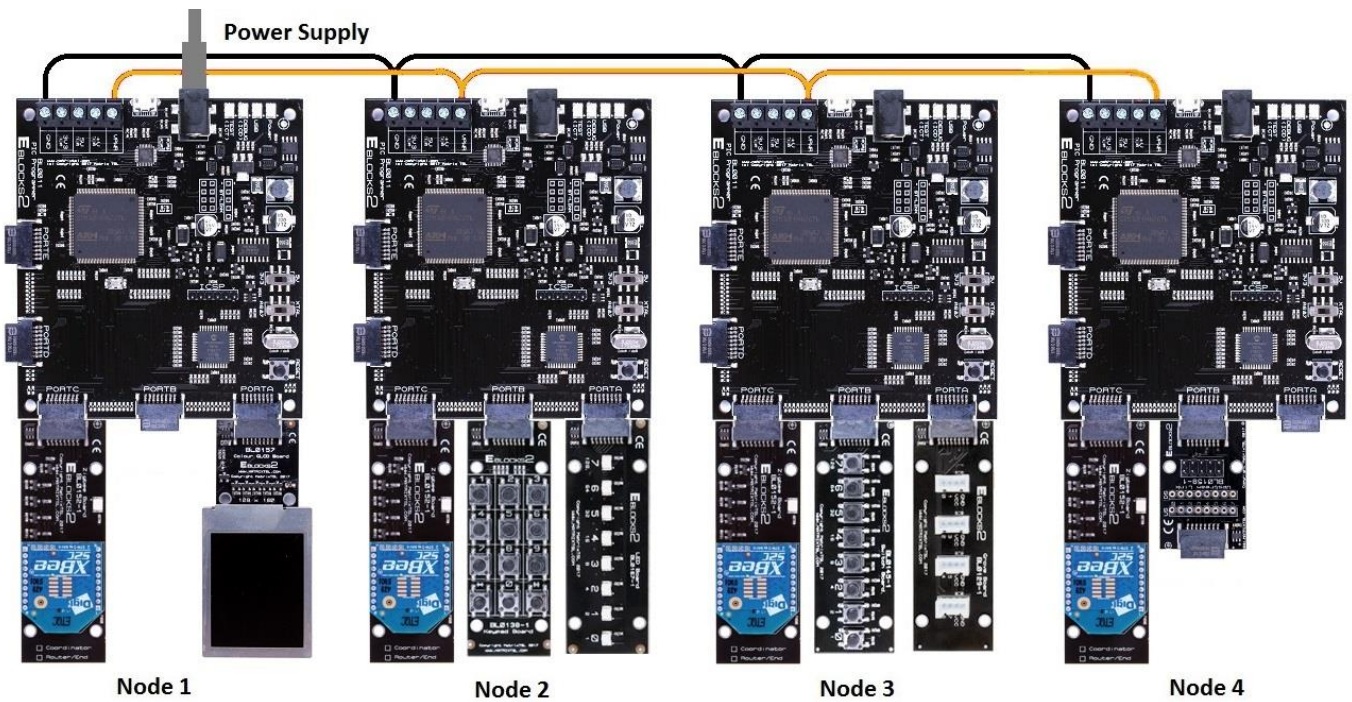
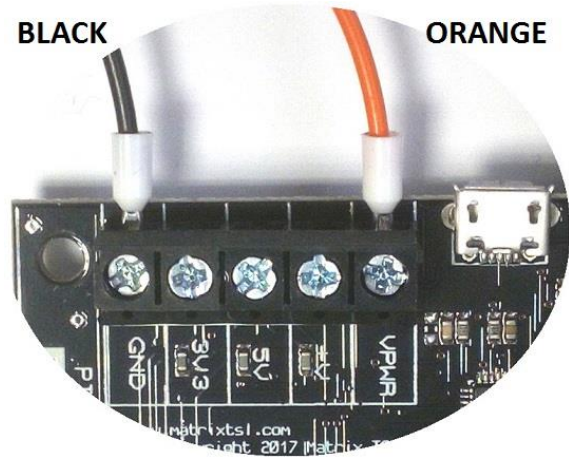
- the relationship between IEEE 802.15.4 and ZigBee protocols;
- the electrical principles behind ZigBee transmissions;
- the components that make up a ZigBee network;
- the topologies available to ZigBee networks;
- the addressing schemes;
- ZigBee routing;
- Sleep modes and packet buffering;
- ZigBee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

What you will need:

To complete this course you will need the following equipment:

- Flowcode version 8 or higher software
- Power supply
- USB programming lead
- E-blocks2 boards including:
 - Four BL0011 or BL0055 (with Arduino Uno fitted)
 - One BL0152C ZigBee co-ordinator
 - Three BL0152R ZigBee router / end-node
 - One BL0157 Graphical Colour LCD
 - One BL0145 Switches
 - One BL0138 Keypad
 - Two BL0167 LEDs
 - One BL0158 Connector
 - One BL0129 Grove interface with light and rotary sensors

Configuration of the base panels:



Note that “Node 1” has the Zigbee Coordinator board (BL0152C), all the others have a Zigbee Router/End board (BL0152R)

Configuration of the downstream boards:

Exercise 1

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C

Exercise 2

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167	BL0138	BL0152R	BL0167	BL0138	BL0152R

Exercise 3, 4, 5

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167		BL0152R	BL0167		BL0152R
Node 3	BL0145		BL0152R	BL0145		BL0152R
Node 4			BL0152R			BL0152R

Exercise 6

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167		BL0152R	BL0167		BL0152R
Node 3	BL0129		BL0152R	BL0129		BL0152R
Node 4	BL0167		BL0152R	BL0167		BL0152R

Exercise 7

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167		BL0152R	BL0167		BL0152R
Node 3	BL0129		BL0152R	BL0129		BL0152R
Node 4		BL0158	BL0152R		BL0158	BL0152R

Exercise 8, 9

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167	BL0138	BL0152R	BL0167	BL0138	BL0152R
Node 3	BL0129	BL0145	BL0152R	BL0129	BL0145	BL0152R
Node 4		BL0158	BL0152R		BL0158	BL0152R

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 web site www.matrixtsl.com for the latest E-Blocks2 documentation.

Scheme of work

Section	Notes for instructors	Timing (minutes)
1. Introduction to ZigBee		
1.1 Overview	Students familiarise themselves with various wireless technologies for WPANs, WLANs and WWANs. They can use websites such as www.ieee802.org/15/ , or en.wikipedia.org/wiki/WPAN for more information.	10 - 30
1.2 Comparison of Wireless Technologies	This section compares the characteristics of ZigBee, Bluetooth, UWB, Wi-Fi and Wi-MAX wireless technologies. Students can use an internet Search Engine to find out more information about any of these technologies.	10 - 30
1.3 ZigBee Applications	Two application areas, the Smart home and the Smart Office, are outlined here. There is a wealth of information available on the internet, especially from the ZigBee Alliance website which contains a 'White Papers' area on www.zigbee.org/en/resources/whitepapers1.asp	10 - 30
1.4 ZigBee protocol outline	This section starts with an overview of the standard OSI networking model, and compares it with the three layer model used by the ZigBee Alliance.	10 - 30
1.5 IEEE 802.15.4 Standard	The focus of this standard is outlined, and contrasted with that of the 802.11 and 802.16 standards. The areas covered by the PHY and MAC layer specs are then described. There is a brief outline of the modulation techniques, AM, FM and PSK, including QPSK and O-QPSK, and DSSS. Finally the two 802.15.4. device types, FFD and RFD, are contrasted.	10 - 30

2. The ZigBee Network			
2.1	ZigBee Logical Devices	<p>After a brief reminder about the two kinds of 802.15.4. physical devices, this section introduces the three kinds of ZigBee logical device, the co-ordinator, the router and the end device. Students should be familiar with these roles, and relate the performance of FFD and RFD devices to them.</p>	5
2.2	ZigBee Network Topologies	<p>This section compares and contrasts the three types of topology recognised by the ZigBee protocol stack – star, cluster tree (sometimes called simply ‘tree’) and mesh. They should be aware that the size of the proposed ZigBee network will probably dictate which of these is used. The characteristics, in terms of latency, routing and reliability are very different for these three topologies. Students need to aware of the significance of redundancy in leading to increased reliability.</p>	10 - 30
2.3	Multi-Access Networks	<p>ZigBee networks are multi-access because usually there are a number of radio transmitters, one to a node, all vying for use of the air-waves. Only one can transmit at a time, otherwise a collision occurs and the data will be corrupted. The issue is – what decides which radio transmits!</p> <p>In a fuller discussion, the distinction between deterministic and non-deterministic networks could be raised, and a comparison drawn with Ethernet vs Token Ring cabled networks.</p> <p>Here, the differences between beacon access (deterministic) and non-beacon access (non-deterministic) are explained. In theory, when beacon frames are used, collisions cannot occur. Again, the issue of which type to choose comes down usually to the size of the network. In a small network, non-beacon access probably has the edge in terms of latency.</p> <p>A possible advantage of beacon access is that it allows the routing devices to use sleep mode, whereas in the free-for-all atmosphere of non-beacon access, they must be awake at all times to manage any transmissions that take place.</p>	10 - 30
2.4	Creating a ZigBee network	<p>The role of the co-ordinator is paramount in setting up the network. It chooses a quiet transmission channel, and unique PAN ID to ensure as little interference with any neighbouring ZigBee networks. Also configured at this time are the parameters Cm, maximum number of children of a router, Rm the maximum number of child routers a router can have and Lm, the depth of the network. These will determine the addressing strategy used in the network.</p>	10 - 30

<p>2.5 ZigBee Addressing Scheme</p>	<p>This is a major topic and needs to be well understood by the students. The MAC address is permanent and unique to the device, but tells you nothing about the functionality or location of the device. The network address can help to locate the node, but may change if network devices fall over, or if the node is mobile. Name addresses can be used to categorise nodes by their functionality.</p> <p>Students should appreciate the differences between unicast and broadcast transmissions, and realise that broadcasts, while an essential part of the network, limit its performance, and should be kept to a minimum.</p>	<p>10 - 30</p>
<p>2.6 Routing and route discovery</p>	<p>This section describes two distinct forms of routing – tree routing and AODV routing. Once again, size is all-important in determining which will be used. A small tree will use tree routing. The disadvantage, the possibility that there might be shorter route than climbing up or down the tree structure, is not significant unless the tree is big. A mesh network has no tree, and so must use AODV.</p>	<p>10 - 30</p>
<p>2.7 Sleep Mode</p>	<p>There are two modes of sleep used in the XBee modules – pin sleep and cyclic sleep. The former is simply a matter of applying a voltage to a pin on the chip. The latter requires the use of AT commands and a timer.</p> <p>The associated issue is that of packet buffering, and the role of the parent device. The child must synchronise with the parent on wakeup to receive any buffered unicast messages that came its way while asleep.</p> <p>Students should be aware of the term ‘indirect messaging’ in this context.</p>	<p>10 - 30</p>
<p>2.8 Security</p>	<p>ZigBee networks offer three levels of security:</p> <ol style="list-style-type: none"> 1. Low level security 2. Use of Access Control Lists 3. Encryption <p>Access Control Lists use the MAC address of devices to decide which nodes should be allowed to join the network, and hence be party to its communications.</p> <p>Encryption relies on the 128 bit AES (Advanced Encryption Standard) which uses a private key, shared by participants, to encrypt messages.</p>	<p>10 - 30</p>

3. The Matrix ZigBee Training Solution		
3.1 The ZigBee E-Block	<p>This section starts with a description of the hardware provided in the Matrix kit.</p> <p>It goes on to introduce the use of AT (“Attention”) commands to control the XBee modules. Students do not need to memorise these commands, or worry about the syntax, as Flowcode delivers them through its macros.</p>	5
3.2 Installation	<p>Students need to read this while keeping an eye on the hardware itself. They should familiarise themselves with the layout of the Processor board board, and in particular, identify the position of its various ports.</p> <p>Detailed information on all the E-Blocks hardware is available from the Matrix website. Students should be encouraged to read and use this information.</p> <p>They can find valuable information about the ZENA Wireless Adapter and the Wireless Development Studio via a free download from the Microchip website. This gives detailed but straightforward instructions on how to install and run the device.</p>	5 - 20

Section	Notes for instructors	Timing (minutes)
4. The ZigBee assignments		
	This section lists the assignments and gives a brief outline of what each covers.	5

Section	Notes for instructors	Timing (minutes)
5. Exercise 1 – Moulding the network		
5.1 Introduction	<p>This is the first of a series of practical assignments using Flowcode to control the ZigBee co-ordinator device.</p> <p>The aim is to configure the device, using the Flowcode ZigBee Component properties. The significance of each section of this is explained.</p> <p>The students are also given information about the various hardware jumper settings for the E-Blocks boards, and target configuration details.</p> <p>They use the ZENA network analyser for the first time, and should take time to understand the output of this analysis.</p> <p>Detailed instructions on how to build the Flowcode program are given in the 'What to do' section. It is assumed that students already know how to:</p> <ul style="list-style-type: none"> load new components into a Flowcode program; configure component properties; add a new variable to a program; add and configure a time delay. <p>A suitable Flowcode program is described in the 'Solutions to Exercises' section.</p>	30
5.2 Objective		
5.3 Requirements		
5.4 The Flowcode program in detail		
5.4.1 Target microcontroller device		
5.4.2 Flowcode ZigBee component		
5.4.3 ZigBee component settings		
5.4.4 InitNetwork function		
5.4.5 Configuring the co-ordinator		
5.5 What to do		
5.5.1 Analysing the network with ZENA		
5.6 Further work		
6. Exercise 2 – Adding a Node		
6.1 Introduction	<p>The aim is to write a program to create a two-node ZigBee network and to monitor communications between the nodes.</p> <p>The co-ordinator node is set up as in the last exercise. The second node, an end node, uses the LEDs to indicate progress in joining the network.</p> <p>After that, when a keypad key is pressed, the data is sent to the co-ordinator, which displays the digit on the LCD. In addition, a Flowcode macro measures the signal strength, which is also displayed.</p> <p>The students are introduced to three more AT commands, and two macros, SendChar and ReceiveChar, used in communication between nodes.</p> <p>Detailed instructions on how to build the Flowcode program are given in the 'What to do' section. It is assumed that students already know how to:</p> <ul style="list-style-type: none"> incorporate a loop into the program; use Decision boxes; output a number to a Port <p>A suitable Flowcode program is described in the 'Solutions to Exercises' section.</p>	30
6.2 Objective		
6.3 Requirements		
6.4 The Flowcode program in detail		
6.4.1 End device node		
6.4.2 Talking on the network		
6.4.3 Specifying the destination for the message		
6.4.4 Monitoring the signal strength		
6.5 What to do		
6.5.1 Analysing the network with ZENA		
6.6 Further work		
Section	Notes for instructors	Timing (minutes)

7. Exercise 3 - Expanding the network		
7.1	Introduction	<p>The aim is to write a program to expand to a four-node network and again to monitor communications between the nodes.</p> <p>The co-ordinator node is set up before. The second node behaves as before, but now also uses the LEDs to indicate the state of the switches connected to the other end node, node 3, which transmits that data once it has joined the network. The fourth node is configured as a router.</p> <p>The students are also given information on parent/child association, and the use of names to address nodes. Two new AT commands are introduced.</p> <p>Detailed instructions on how to build the Flowcode program are given in the 'What to do' section. It is assumed that students already know how to use a Calculation icon: to set the value of a variable; to increment the value of a variable;</p> <p>A suitable Flowcode program is described in the 'Solutions to Exercises' section.</p>
7.2	Objective	
7.3	Requirements	
7.4	The Flowcode program in detail	
7.4.1	Configuring the router node	
7.4.2	Parent/child association	
7.4.3	Talking to specific nodes	
7.5	What to do	
7.5.1	Analysing the network with ZENA	
7.6	Further work	
30		
8. Exercise 4 - Reducing power consumption		
8.1	Introduction	<p>A big advantage of ZigBee over other wireless technologies is its low power demand. One way to keep it low is to use sleep mode especially for nodes that rely on battery power. To maintain communication with the network, the parent device of a sleeping node must store messages for it.</p> <p>This network functions the same as the previous one except that node 2 is programmed to enter sleep mode regularly.</p> <p>The students are given information on the two sleep modes used by the XBee modules, one hardware based, one software based, and on packet buffering, and are shown how to use the NodeConfigureSleep and NodeWake macros.</p> <p>Detailed instructions on how to build the Flowcode program are given in the 'What to do' section.</p> <p>A suitable Flowcode program is described in the 'Solutions to Exercises' section.</p>
8.2	Objective	
8.3	Requirements	
8.4	The Flowcode program in detail	
8.4.1	Sleep modes	
8.4.2	Packet Buffering	
8.4.3	Polling for buffered data	
8.4.4	Monitoing the signal strength	
8.5	What to do	
8.6	Further work	
30		

Section	Notes for instructors	Timing (minutes)
9. Exercise 5 – Dynamic networks		
9.1 Introduction	<p>A well-designed naming scheme can assist in monitoring the ZigBee network. In this case, names for the nodes are chosen to reflect their function, so that end devices are given names, which, though unique, all start with “End...” This allows a node to use the ScanNetwork and then ScanBuffer macros to search the names of the nodes identified as members of the network for those beginning with “End” Eventually, the LCD on node 1 displays the names of the nodes detected on the network and highlights those starting with the comparison string “End”.</p> <p>Students are shown how to interpret the output of the ScanBuffer macro.</p> <p>Detailed instructions on how to build the Flowcode program are given in the ‘What to do’ section.</p> <p>A suitable Flowcode program is described in the ‘Solutions to Exercises’ section.</p>	30
9.2 Objective		
9.3 Requirements		
9.4 The Flowcode program in detail		
9.4.1 MAC Addresses		
9.4.2 Identifier Addresses		
9.4.3 Polling for nodes		
9.5 What to do		
9.6 Further work		
10 Exercise 6 – Message Routing		
10.1 Introduction	<p>This exercise looks at using node names as destination addresses to ensure, in this case, that messages from end devices are all routed to the gateway node initially. The end nodes search for this gateway using the ConnectToName macro with parameter string “Gate” as the target</p> <p>After that, they sample the output voltage of the on-board sensors, use the ADC to convert these to digital signals, and then transmit them to the gateway. From there they are relayed to the co-ordinator, and displayed</p> <p>The notes discuss the importance of designing an appropriate message structure, incorporating, in this case, the number of the source node, and a parity byte, used to look for transmission errors. As this is a wireless technology, it is possible for chunks of messages from different sources to arrive as part of the same message. Using parity, such mistakes can be spotted (though not corrected.) A brief outline of the use odd and even parity is given.</p> <p>Detailed instructions on how to build the Flowcode program are given in the ‘What to do’ section.</p> <p>A suitable Flowcode program is described in the ‘Solutions to Exercises’ section.</p>	30
10.2 Objective		
10.3 Requirements		
10.4 The Flowcode program in detail		
10.4.1 Route calibration		
10.4.2 Mobile nodes and beacon signals		
10.4.3 Message structure		
10.5 What to do		
10.6 Further work		

11 Exercise 7 – Datalogging Gateway.		
11.1 Introduction	<p>The previous exercise resulted in passing all data from the end devices through the gateway node, en route to the co-ordinator. This exercise extends that by relaying the data to a computer as well, using HyperTerminal, a common communications application.</p> <p>The notes give detailed guidance on setting up communications using HyperTerminal.</p> <p>The problem in doing this is the data format, ASCII code, used by HyperTerminal, which differs from that used so far.</p> <p>The notes introduce the students to ASCII codes, and to bit banging, the process of using software, rather than hardware, to convert the raw data to serial ASCII format. The Flowcode program uses customised macros, called Byte_2_Number and Number_2_USB232 to do this.</p> <p>Another customised macro, called Format_LCD is used by node 1 to configure the LCD display.</p> <p>Because of the use of these custom macros, and the increased complexity of the programs for node 1 and node 4, the students are advised to copy the programs from the ZigBee course CD-ROM rather than try to construct them themselves. They should be actively encouraged to study the structure of these programs, and modify them to see what happens.</p>	30
11.2 Objective		
11.3 Requirements		
11.4 The Flowcode program in detail		
11.4.1 Collecting the data		
11.4.2 Logging the data		
11.4.3 Bytes and ASCII		
11.4.4 Connecting the USB232 board		
11.4.5 Bit Banging Serial Communications		
11.5 What to do		
11.6 Further work		

Section	Notes for instructors	Timing (minutes)
12 Exercise 8 – Modular fire and burglar alarm system		
12.1 Introduction	<p>The aim of this exercise is take the ZigBee four-node network developed in previous exercises and adapt it to model a real-world system – a burglar / fire alarm</p> <p>The sensor on the Sensor E-Block acts as temperature sensor.</p> <p>The Switch E-Block acts as a PIR sensor.</p> <p>The LCD acts as the central control panel alarm.</p> <p>The keypad is used to activate and de-activate the alarm, using specific 4 digit codes.</p> <p>Another custom macro “String_2_USB232” is used to convert a character string to ASCII format for transmission via HyperTerminal to the computer. In this way, alarm messages, like “Temp Too High”, resulting from examining the temperature raw data, can be relayed to the computer</p> <p>Once again, because of the use of these custom macros, and the increased complexity of the programs for node 1 and node 4, the students are advised to copy the programs from the ZigBee course CD-ROM rather than try to construct them themselves. They should be actively encouraged to study the structure of these programs, and modify them to see what happens.</p>	30
12.2 Objective		
12.3 Requirements		
12.4 The Flowcode program in detail		
12.4.1 Sensors and data inputs		
12.4.2 Data output		
12.4.3 Controlling the operation of the alarm		
12.5 What to do		
12.6 Further work		

Section	Notes for instructors	Timing (minutes)
13 Exercise 9 – Improving network security		
13.1 Introduction	<p>ZigBee networks offer three levels of security:</p> <ol style="list-style-type: none"> 1. Low security – you have to be near enough to pick up the weak radio signals if you want to hack in. 2. Medium security – the co-ordinator will have an access control list – a list of approved node MAC addresses. 3. High security – in addition, all transmissions will be encrypted using AES 128 bit encryption. Unless you have the private encryption key, you cannot unravel the transmissions. <p>This exercise replicates what was done in exercise 8, but now encrypts the transmissions between nodes 1, 2 and 4. Node 3 does not have the private key, and so cannot understand the transmissions.</p> <p>Once again, the students are advised to copy the programs from the course CD-ROM but should be encouraged to study them closely to understand what is going on, and to modify them.</p>	30
13.2 Objective		
13.3 Requirements		
13.4 The Flowcode program in detail		
13.4.1 Encryption commands		
13.5 What to do		

Solutions to Exercises

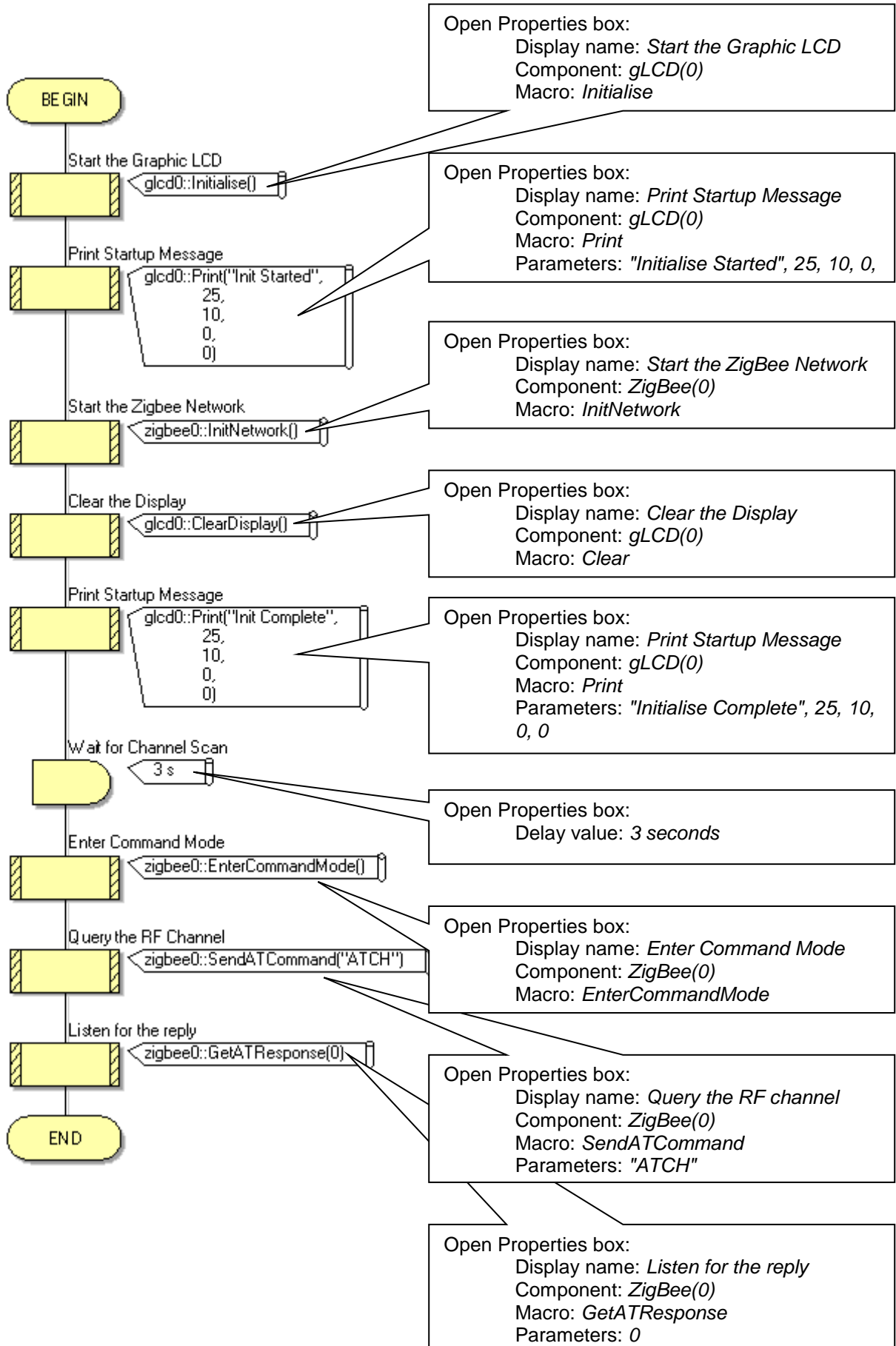
Exercise 1

Node 1 ZigBee(0) Properties

Component	
Handle	zigbee0
Type	Zigbee (EB051, XB...
Properties	
Device Name	Coord
Zigbee Mode	Coordinator
Auto ID	No
PAN ID	564
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
Connections	
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
Scan Channels	
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Simulation	
Label	Zigbee

Channel and port properties will vary depending upon target processor being used. Please see port allocation table on page 6

Exercise 1 – node 1



Exercise 2

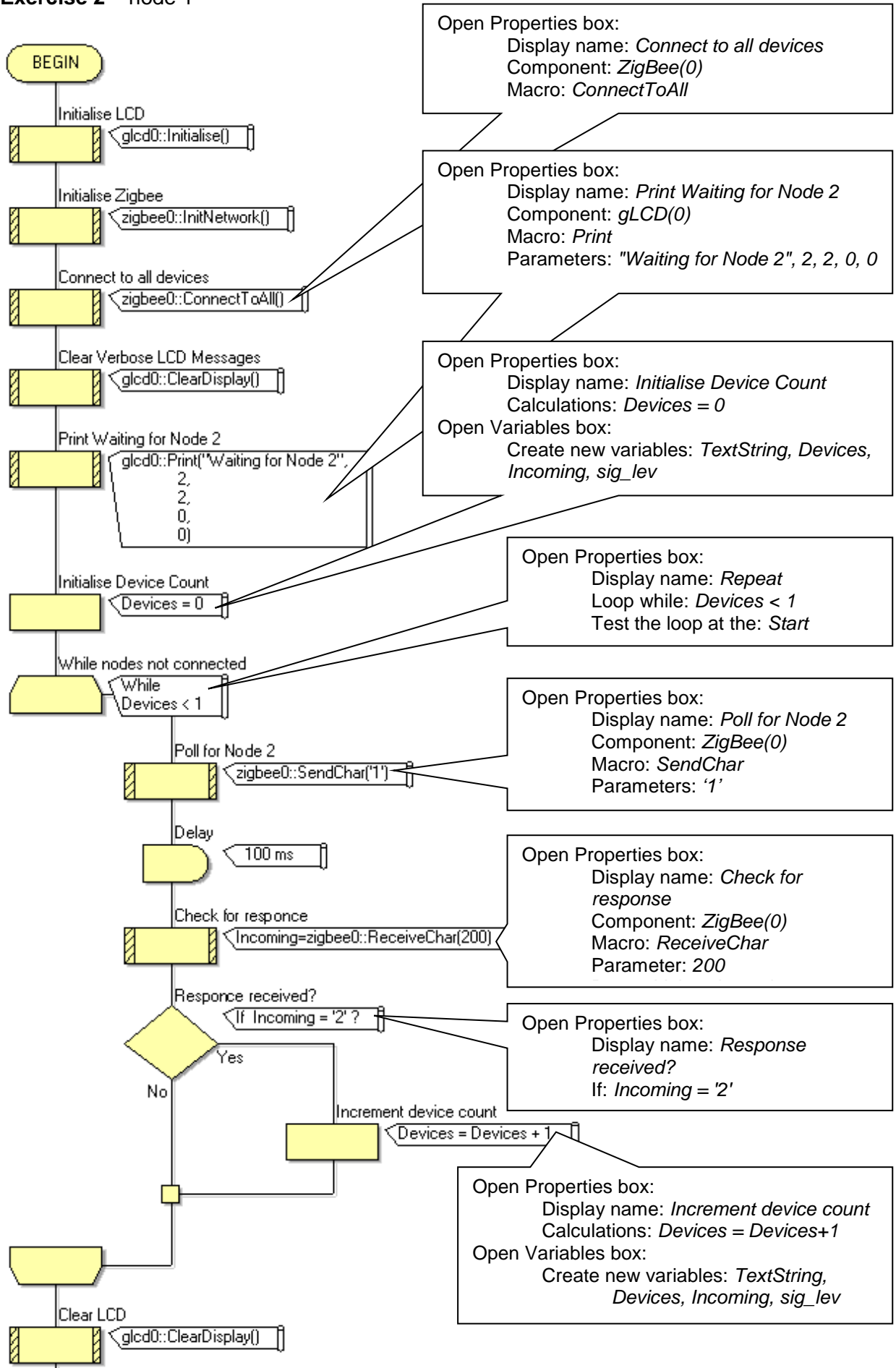
Node 1 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	Coord
Zigbee Mode	Coordinator
Auto ID	No
PAN ID	564
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	No
0x18	No
0x19	No
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	120
Label	Zigbee

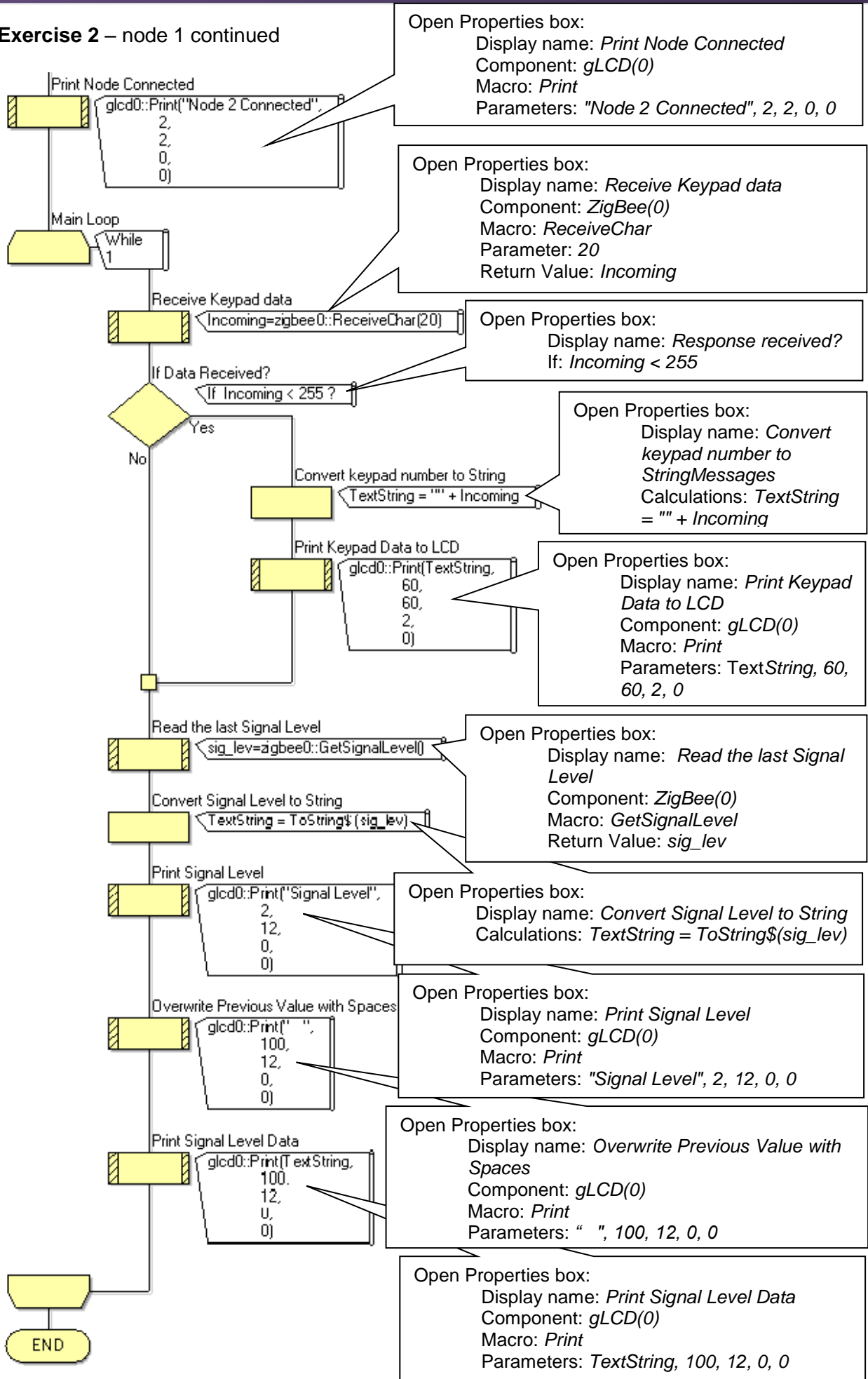
Node 2 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	Coord
Zigbee Mode	End Device
Auto ID	No
PAN ID	564
Join Notification	Coordinator
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	No
0x18	No
0x19	No
0x1A	Yes
Scan Duration	2
Total Scan Time (ms)	60
Label	Zigbee

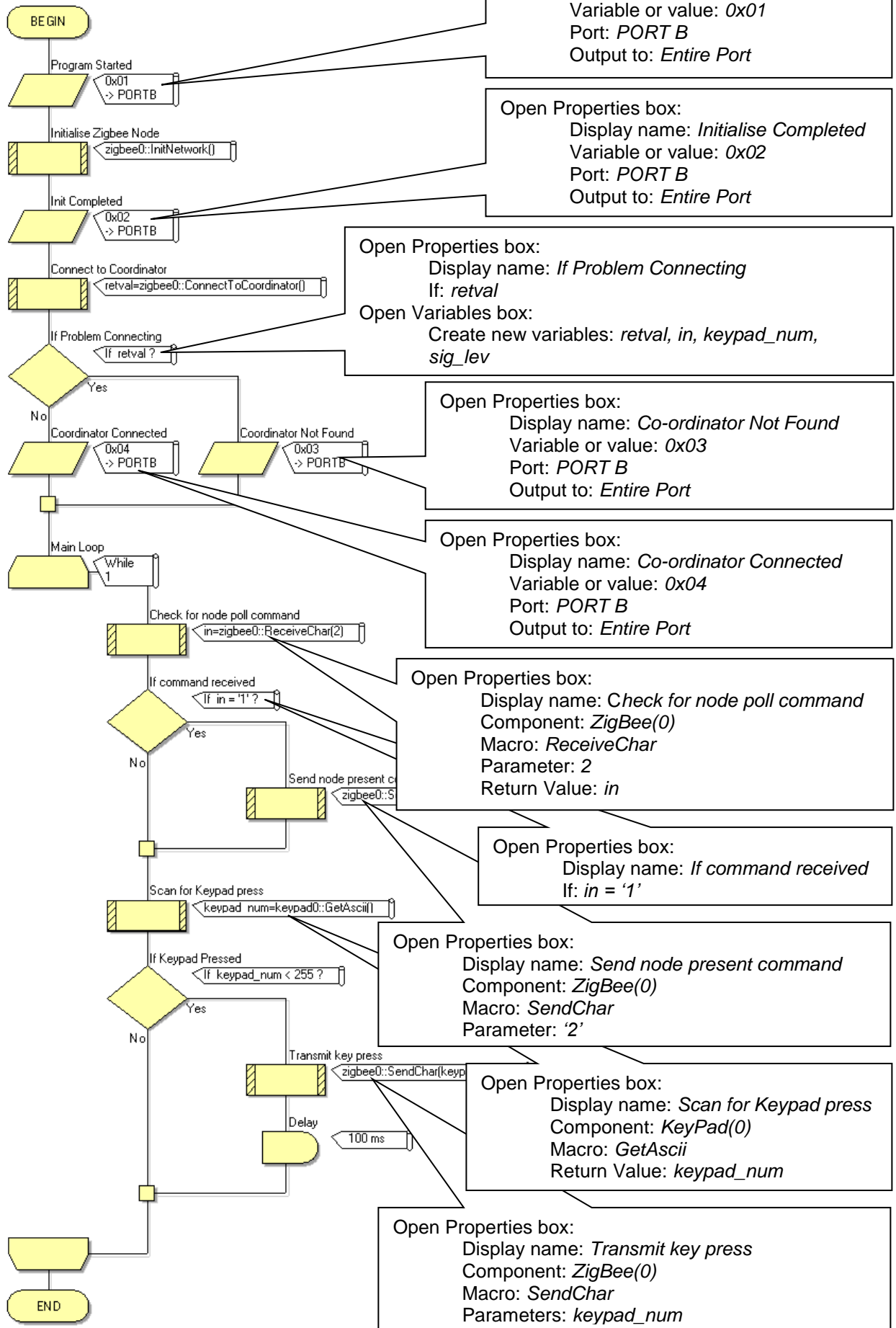
Exercise 2 – node 1



Exercise 2 – node 1 continued



Exercise 2 – node 2



Exercise 3

Node 1 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...
Device Name	Coord
Zigbee Mode	Coordinator
Auto ID	No
PAN ID	564
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	No
0x18	No
0x19	No
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	120
Label	Zigbee

Node 2 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...
Device Name	End1
Zigbee Mode	End Device
Auto ID	No
PAN ID	564
Join Notification	None
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

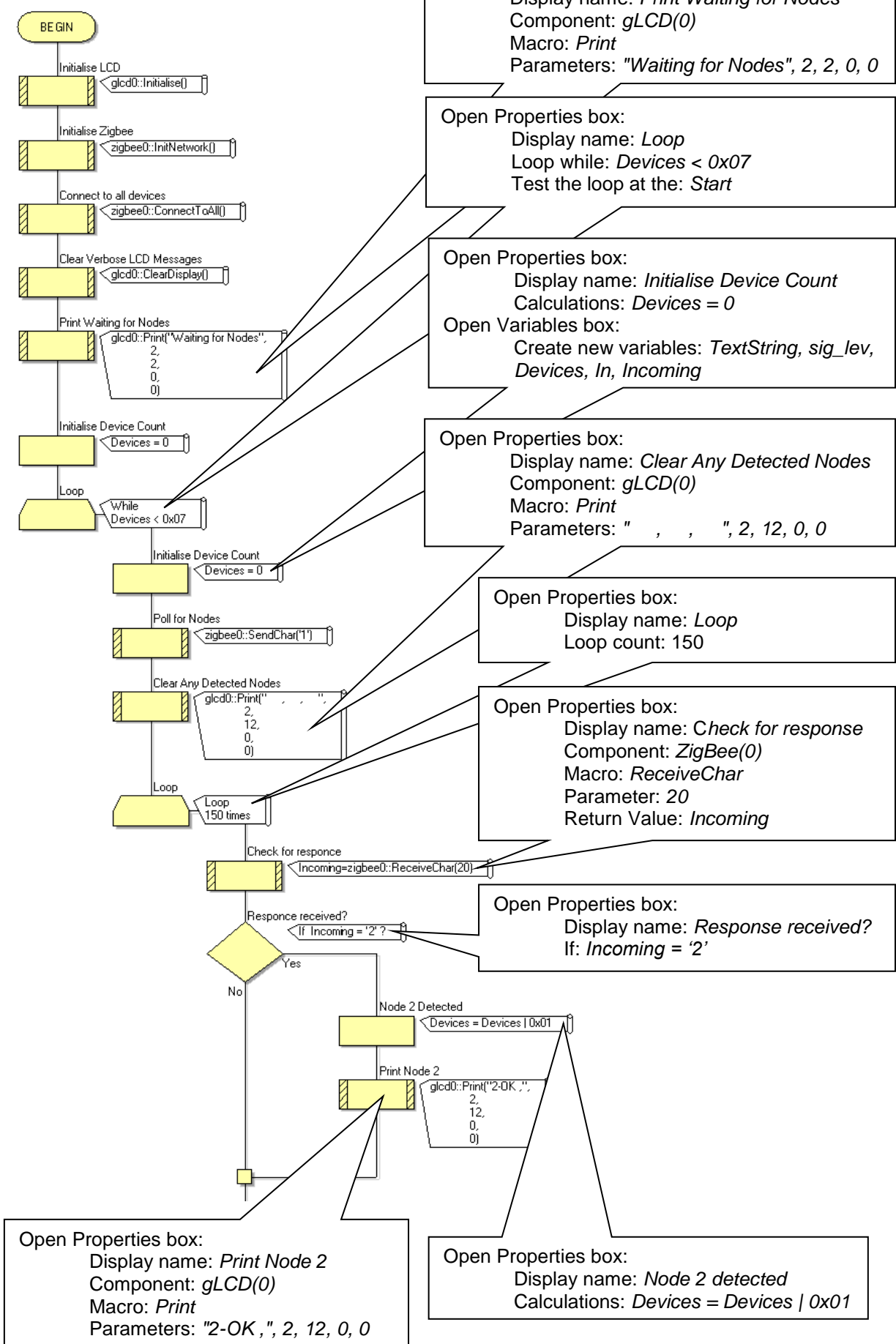
Node 3 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...
Device Name	End2
Zigbee Mode	End Device
Auto ID	No
PAN ID	564
Join Notification	None
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

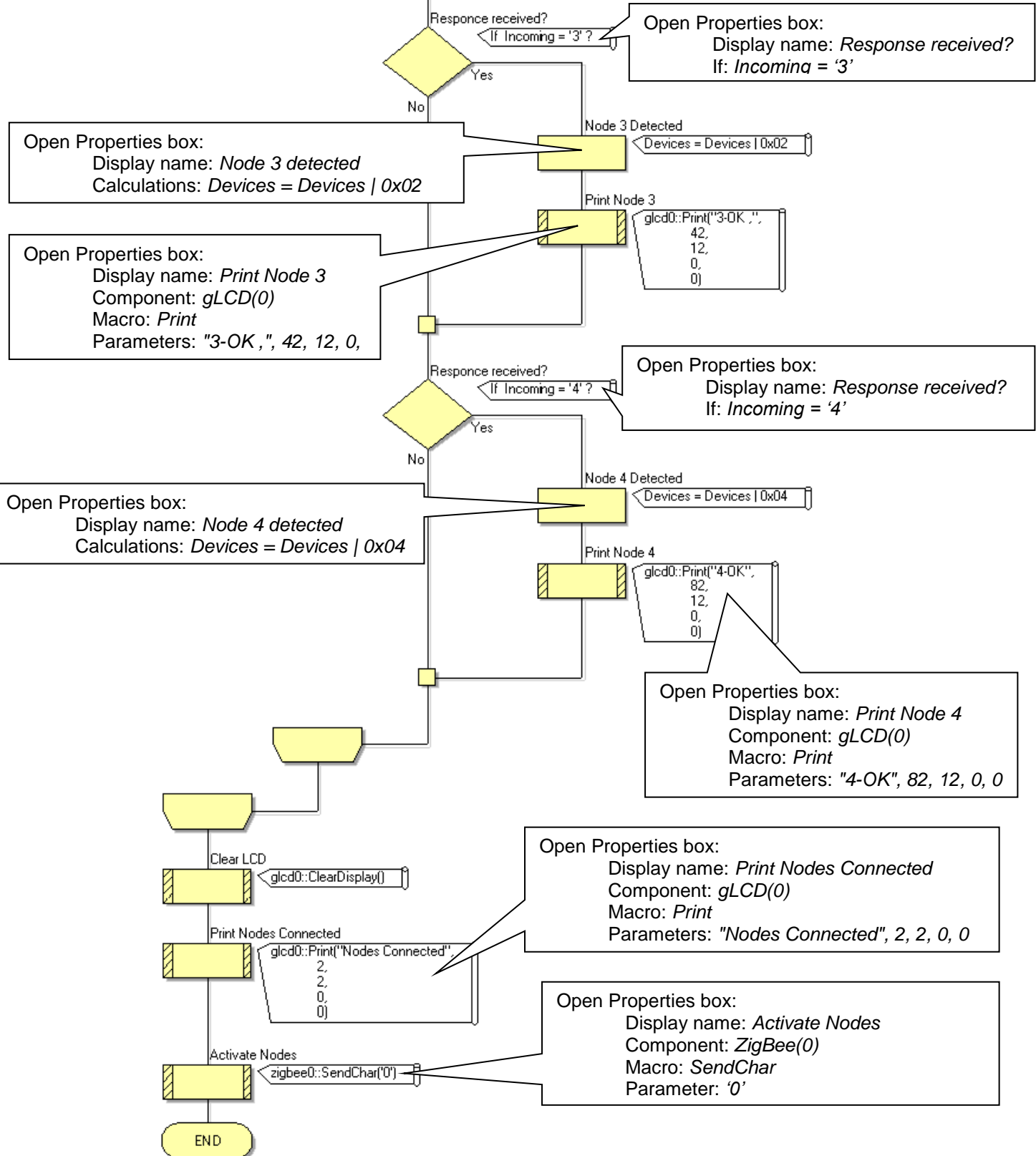
Node 4 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...
Device Name	Gate
Zigbee Mode	Router
Auto ID	No
PAN ID	564
Join Notification	None
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

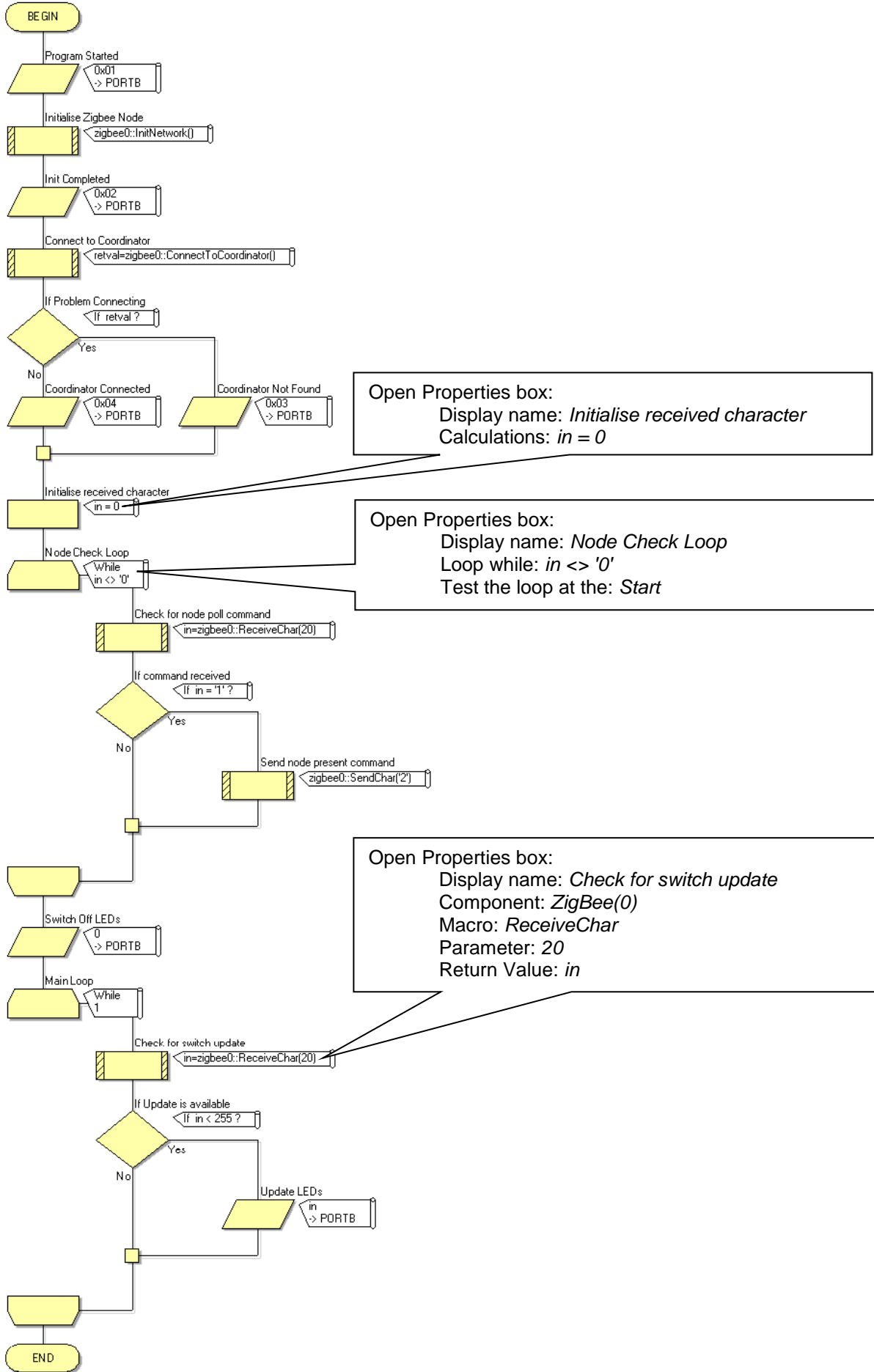
Exercise 3 – node 1



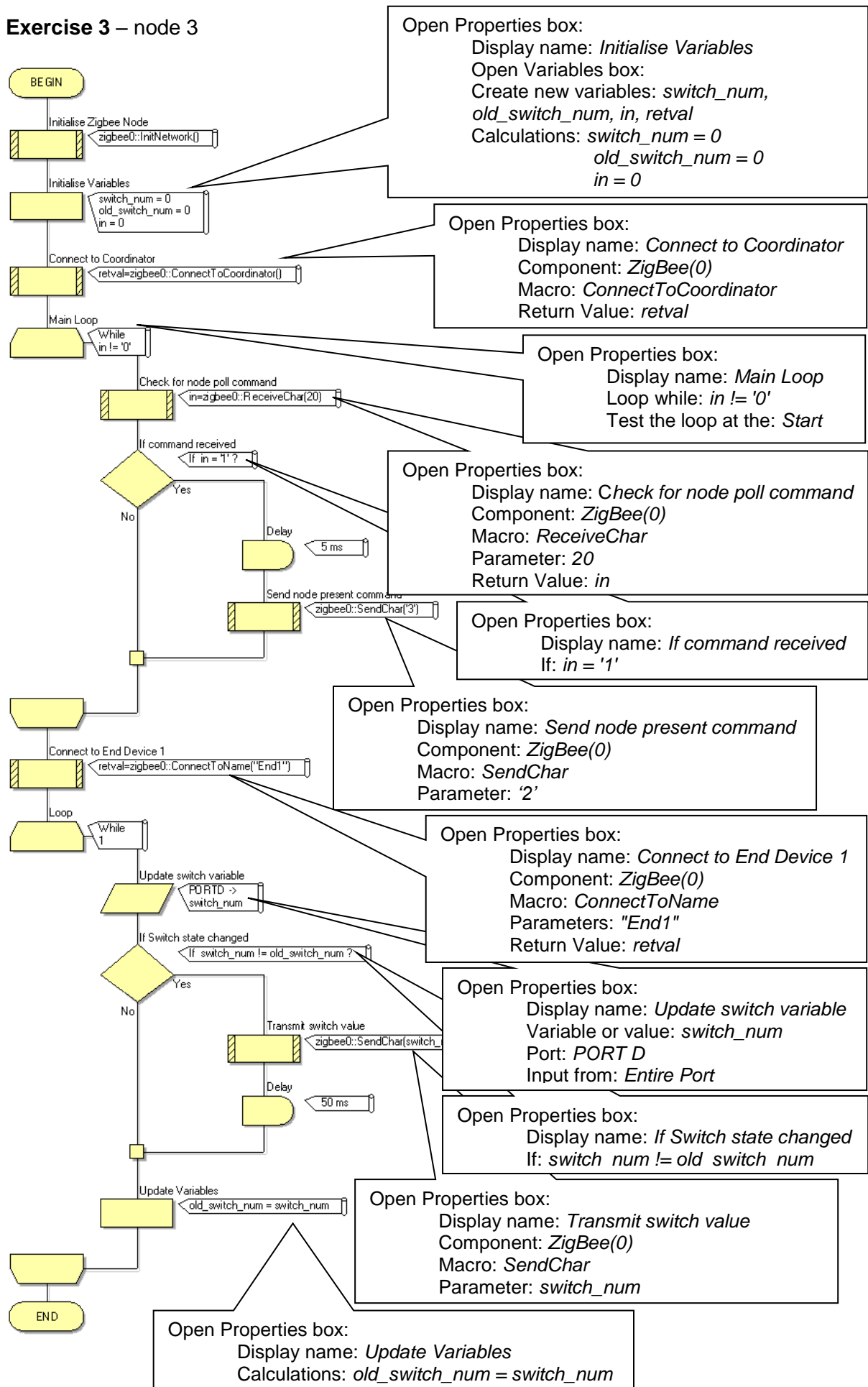
Exercise 3 – node 1 continued



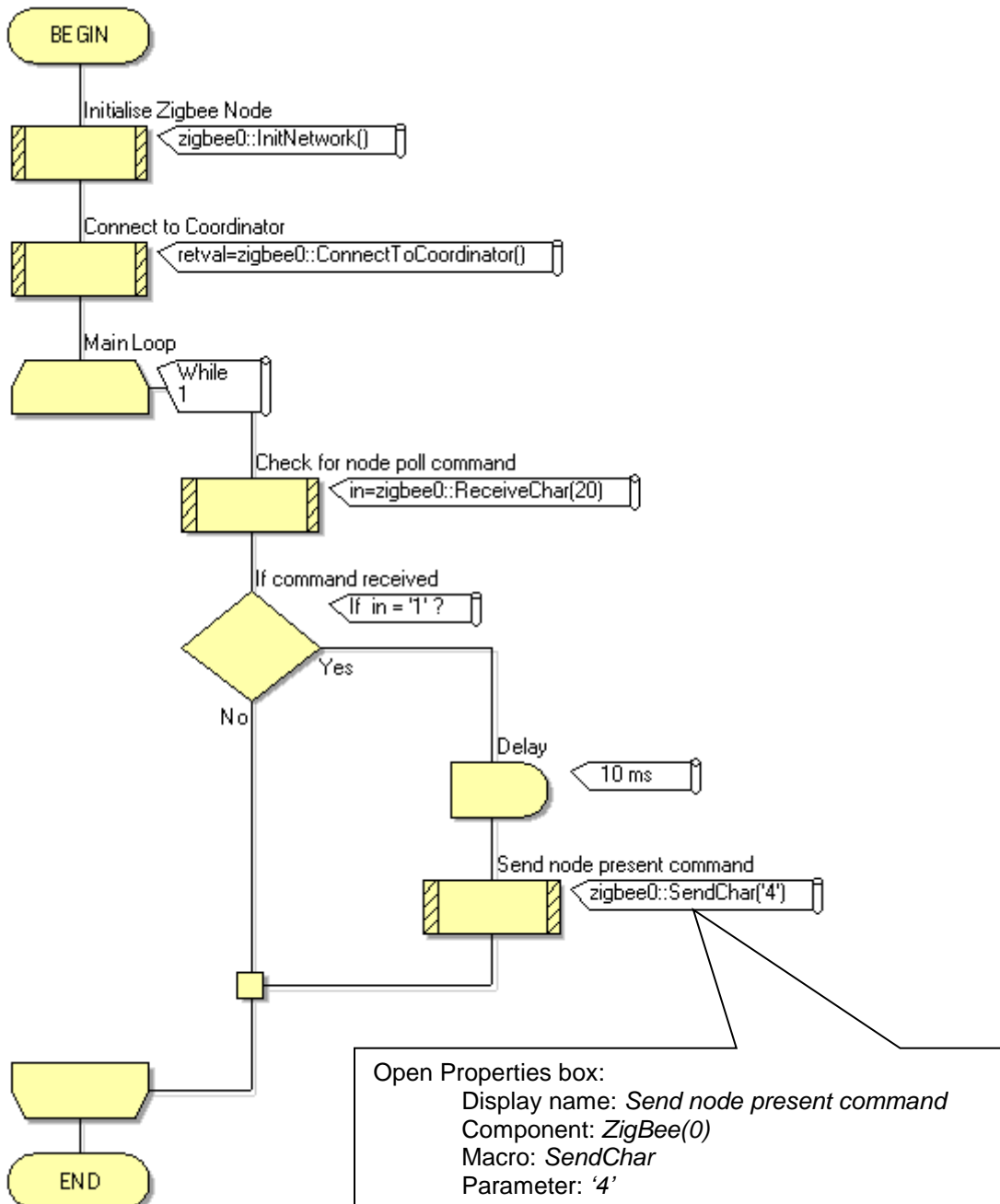
Exercise 3 – node 2 (See notes on exercise 2 node 2 above.)



Exercise 3 – node 3



Exercise 3 – node 4



Exercise 4

Node 1 ZigBee(0) Properties

Node 2 ZigBee(0) Properties

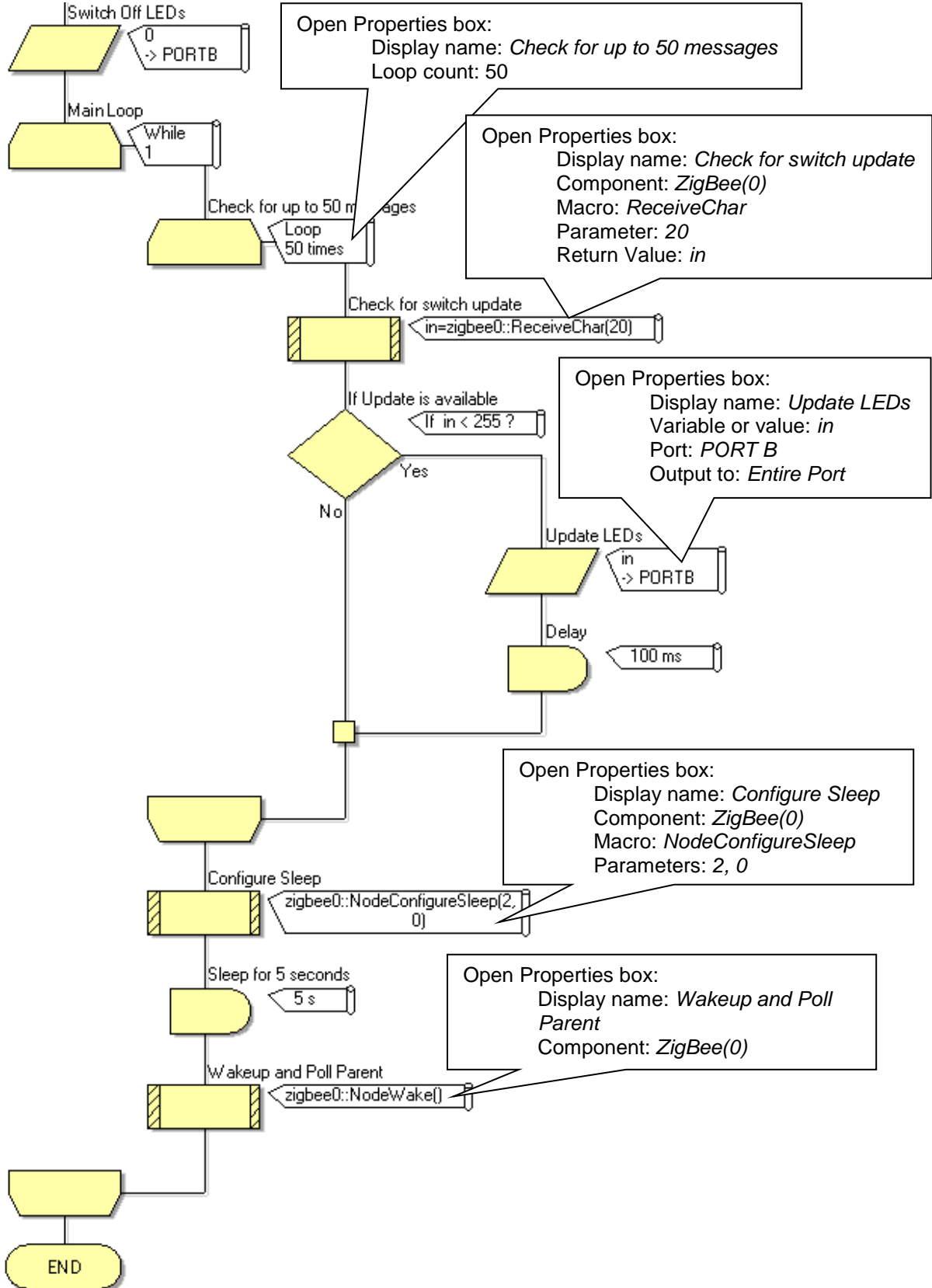
Node 3 ZigBee(0) Properties

Node 4 ZigBee(0) Properties

Exercise 4

The node 1 program is identical to the node 1 program used in exercise 3.
 The node 3 program is identical to the node 3 program used in exercise 3.
 The node 4 program is identical to the node 4 program used in exercise 3.

The node 2 program is identical to the node 2 program used in exercise 3 to the point where all the LEDs are switched off. Then:



Exercise 5

Node 1 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	Coord
Zigbee Mode	Coordinator
Auto ID	No
PAN ID	564
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

Node 2 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	End1
Zigbee Mode	End Device
Auto ID	No
PAN ID	564
Join Notification	None
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

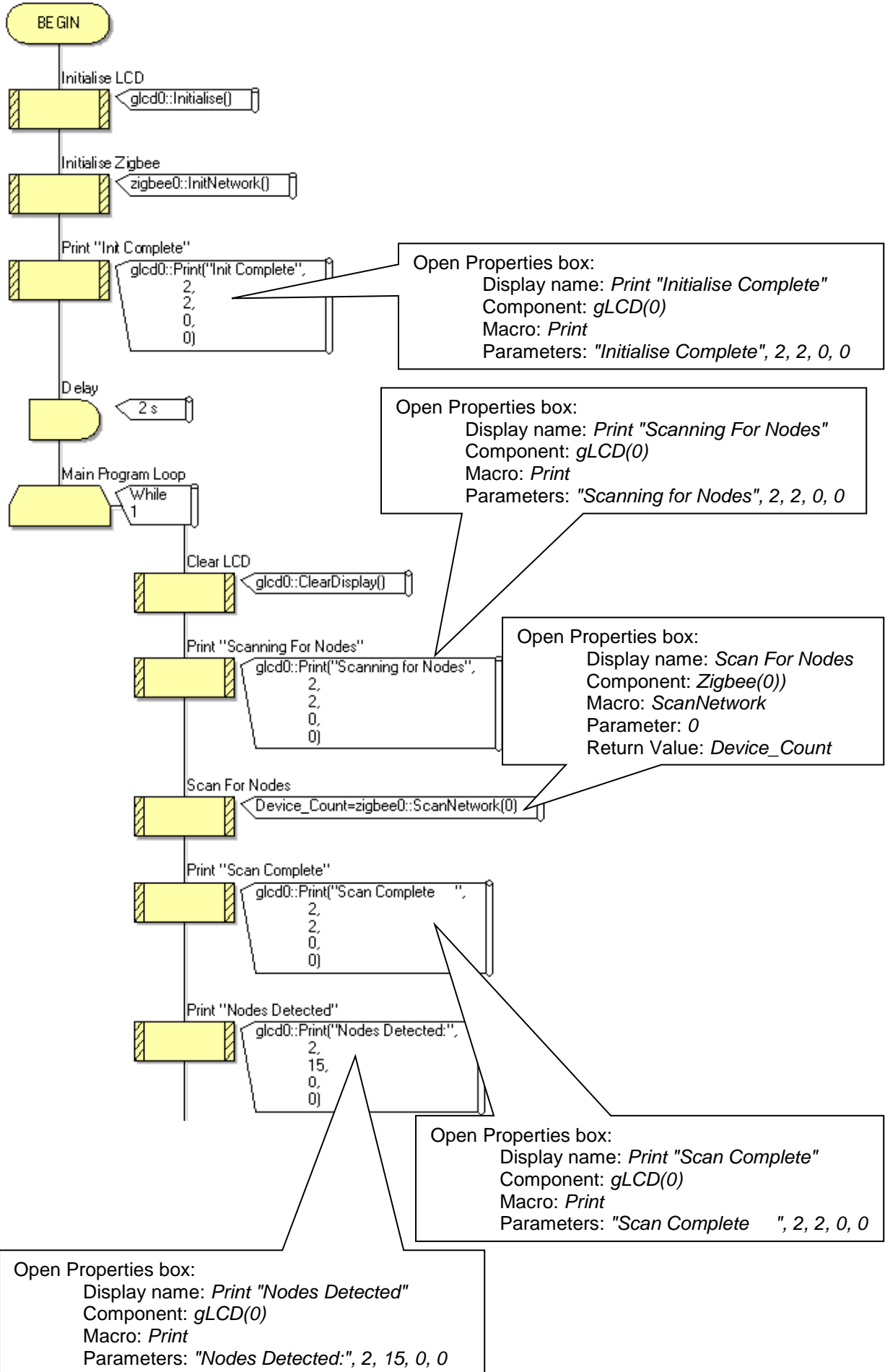
Node 3 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	End2
Zigbee Mode	End Device
Auto ID	No
PAN ID	564
Join Notification	None
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

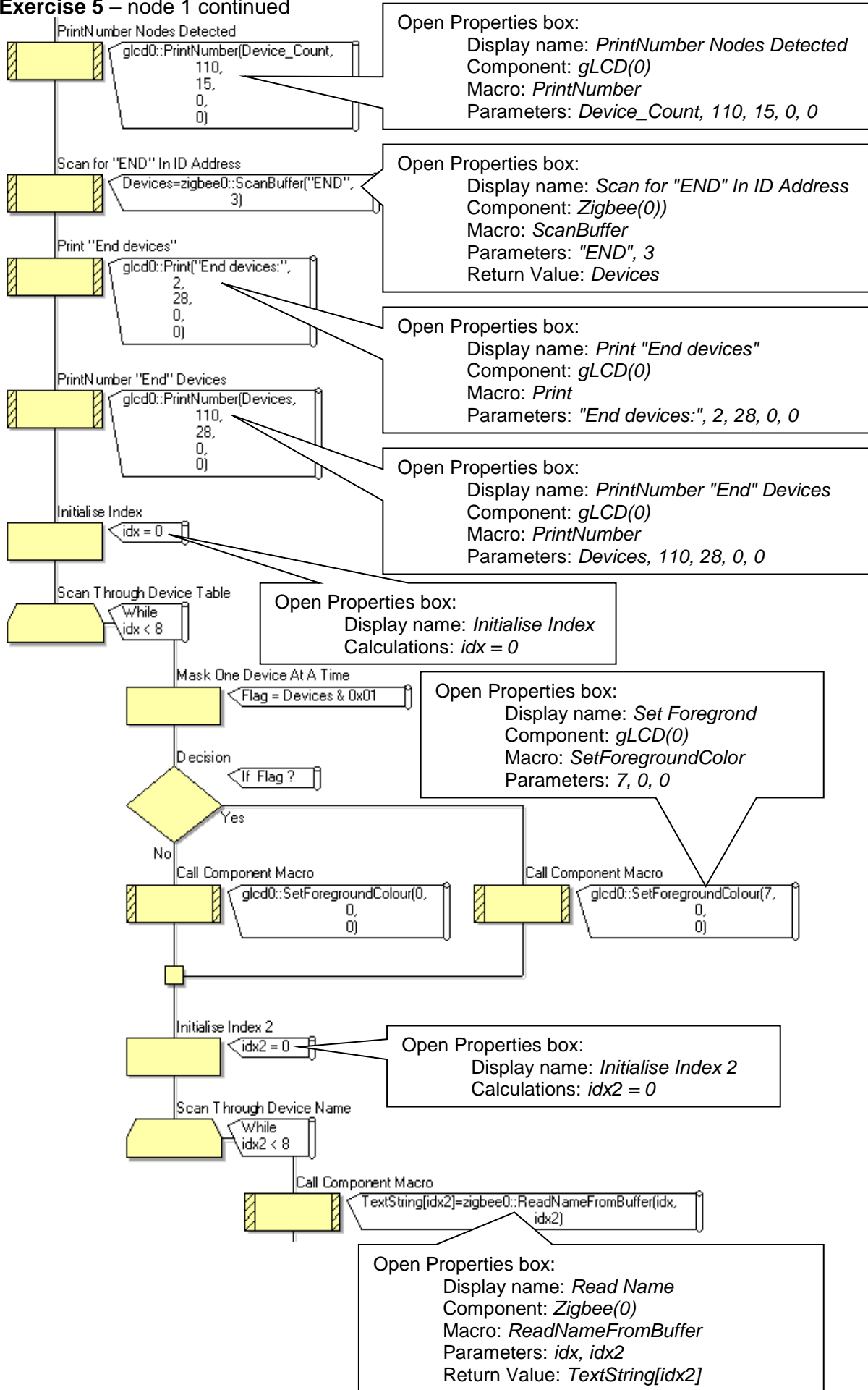
Node 4 ZigBee(0) Properties

Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	Gate
Zigbee Mode	Router
Auto ID	No
PAN ID	564
Join Notification	None
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

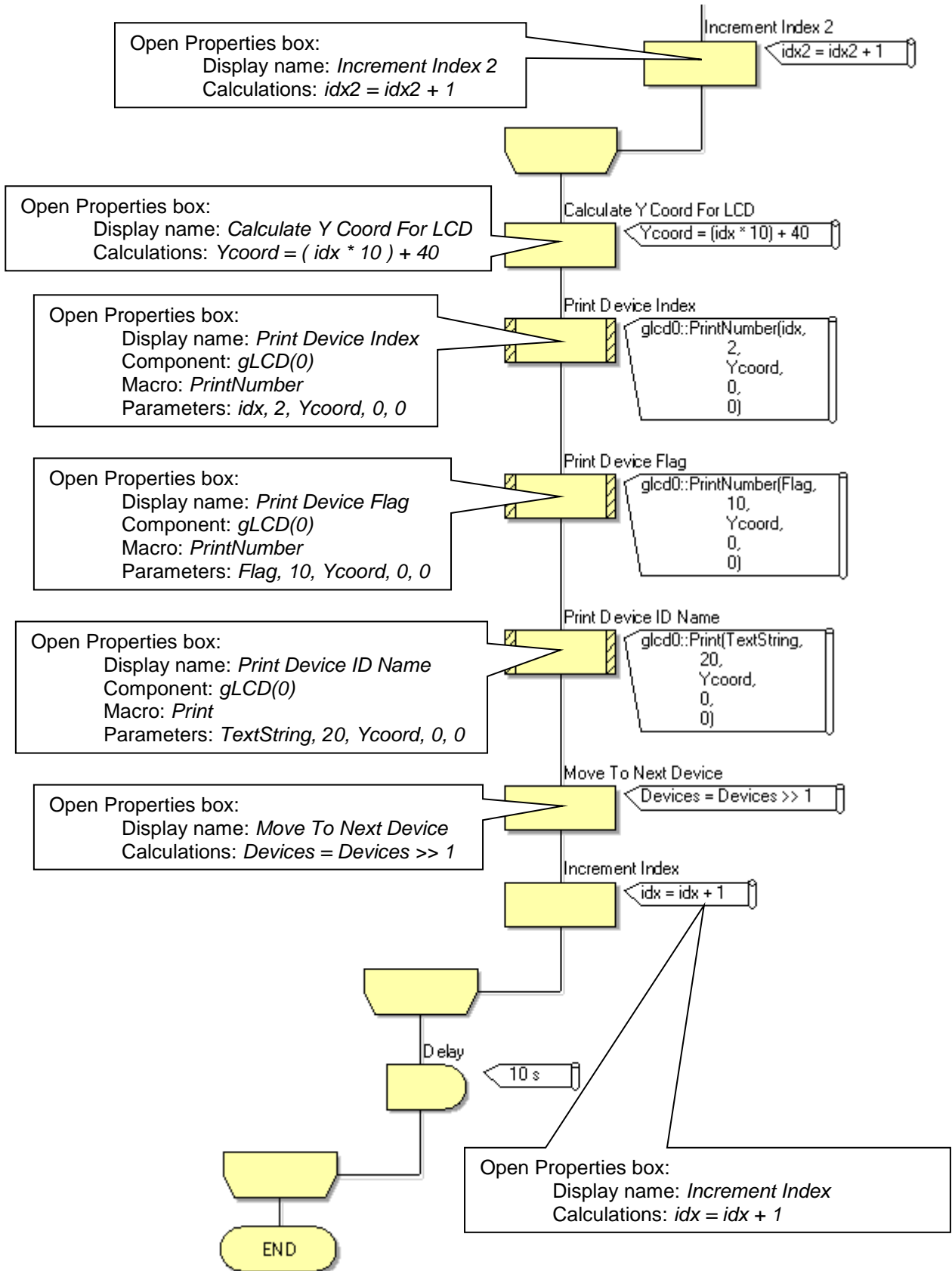
Exercise 5 – node 1



Exercise 5 – node 1 continued



Exercise 5 – node 1 continued



Exercise 5 continued

The node 2 program is identical to the node 2 program used in exercise 3, except that a delay of 100 milliseconds is added after the 'Send node present' command in the 'Yes' loop of the 'If command received' Decision box.

The node 3 program is identical to the node 3 program used in exercise 3, with the following modification:

- 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.

The node 4 program is identical to the node 4 program used in exercise 3 with the following modification:

- 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.

Exercises 6 to 9 - Overview

For these exercises, the Flowcode program structures are complex and so should be copied from the Matrix ZigBee course CD-ROM.

Students should be encouraged to analyse the programs carefully, and to experiment with the configuration.

The following pages contain diagrams showing these Flowcode programs, without the configuration details that have accompanied earlier exercises.

Exercise 6

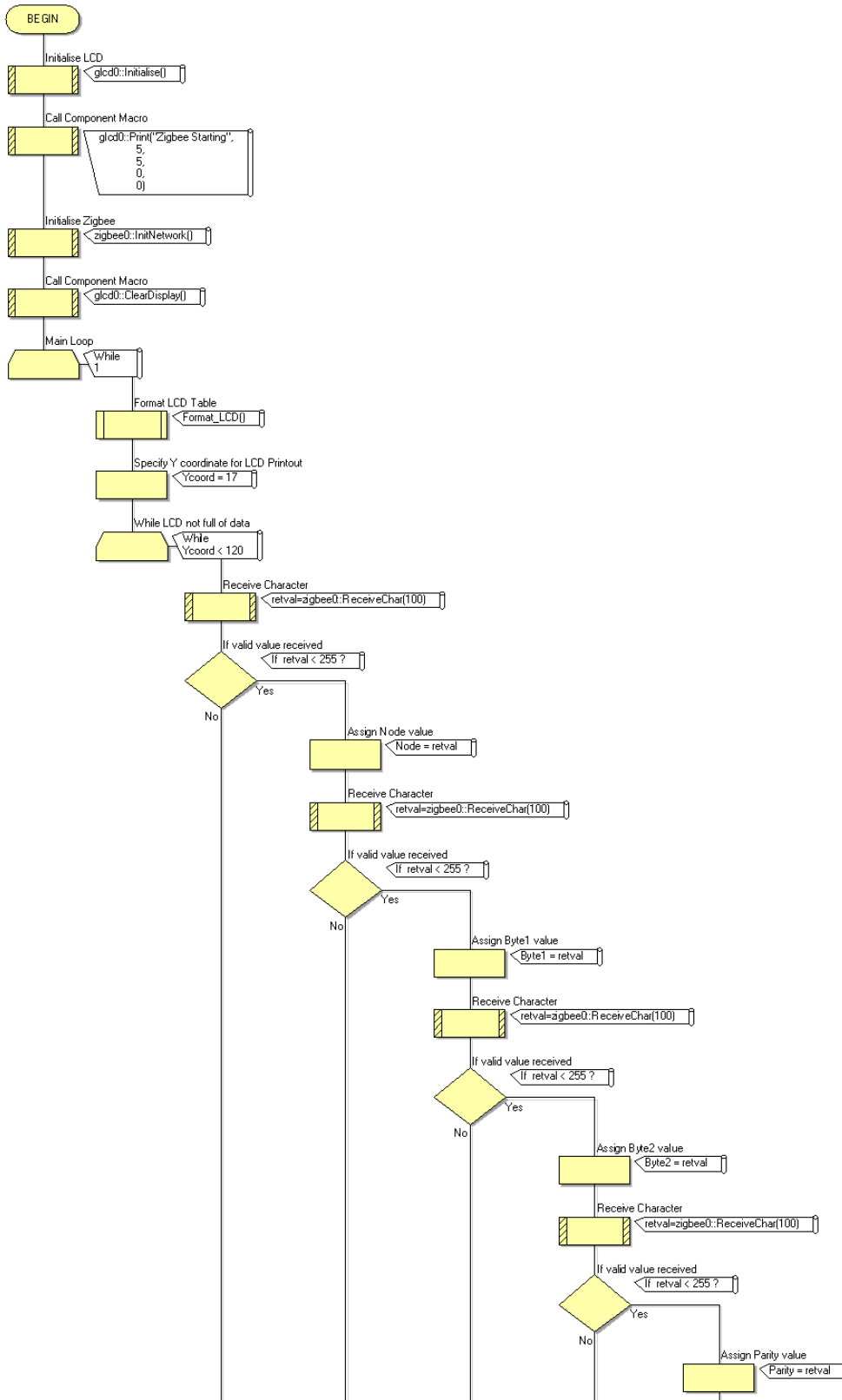
Node 1 ZigBee(0) Properties

Node 2 ZigBee(0) Properties

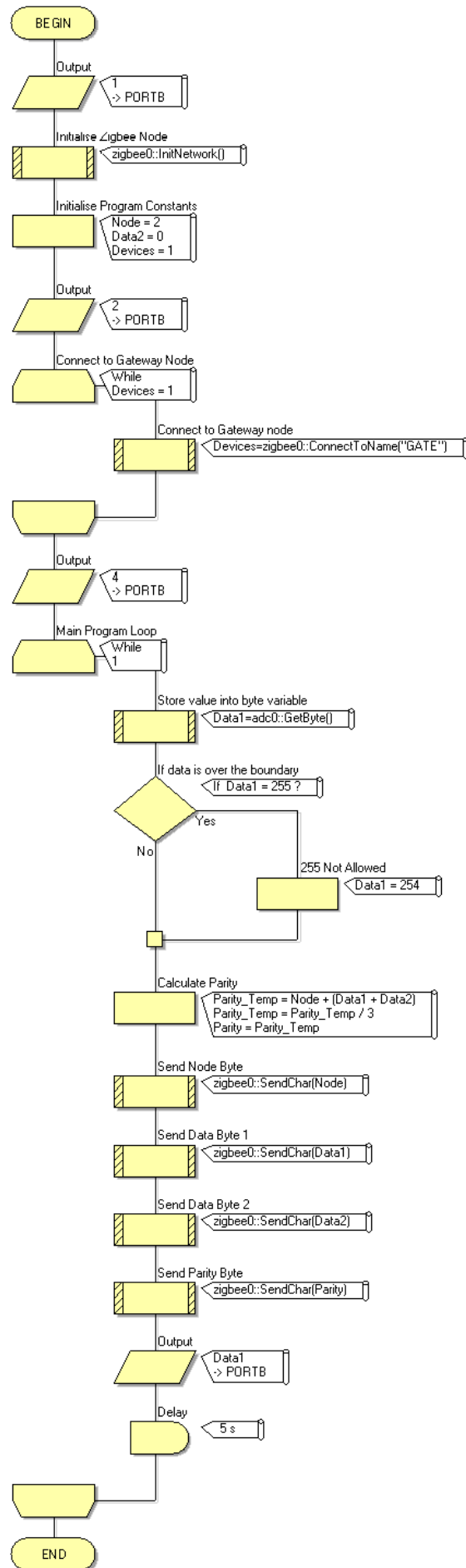
Node 3 ZigBee(0) Properties

Node 4 ZigBee(0) Properties

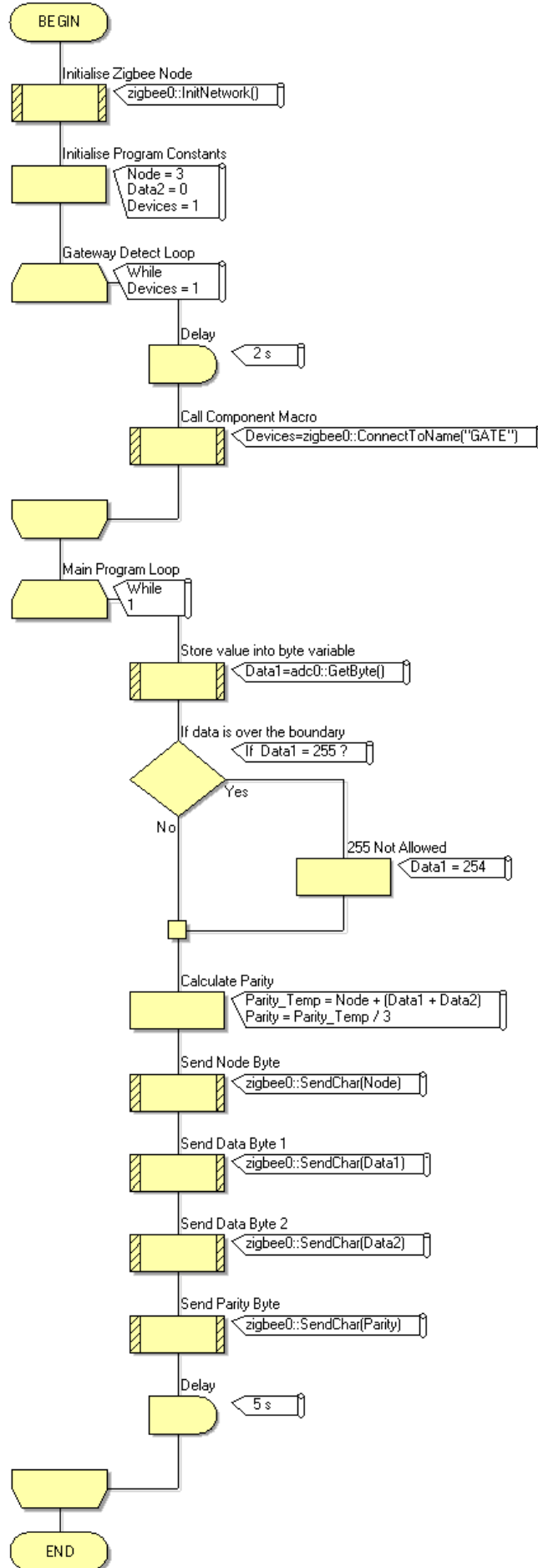
Exercise 6 – node 1



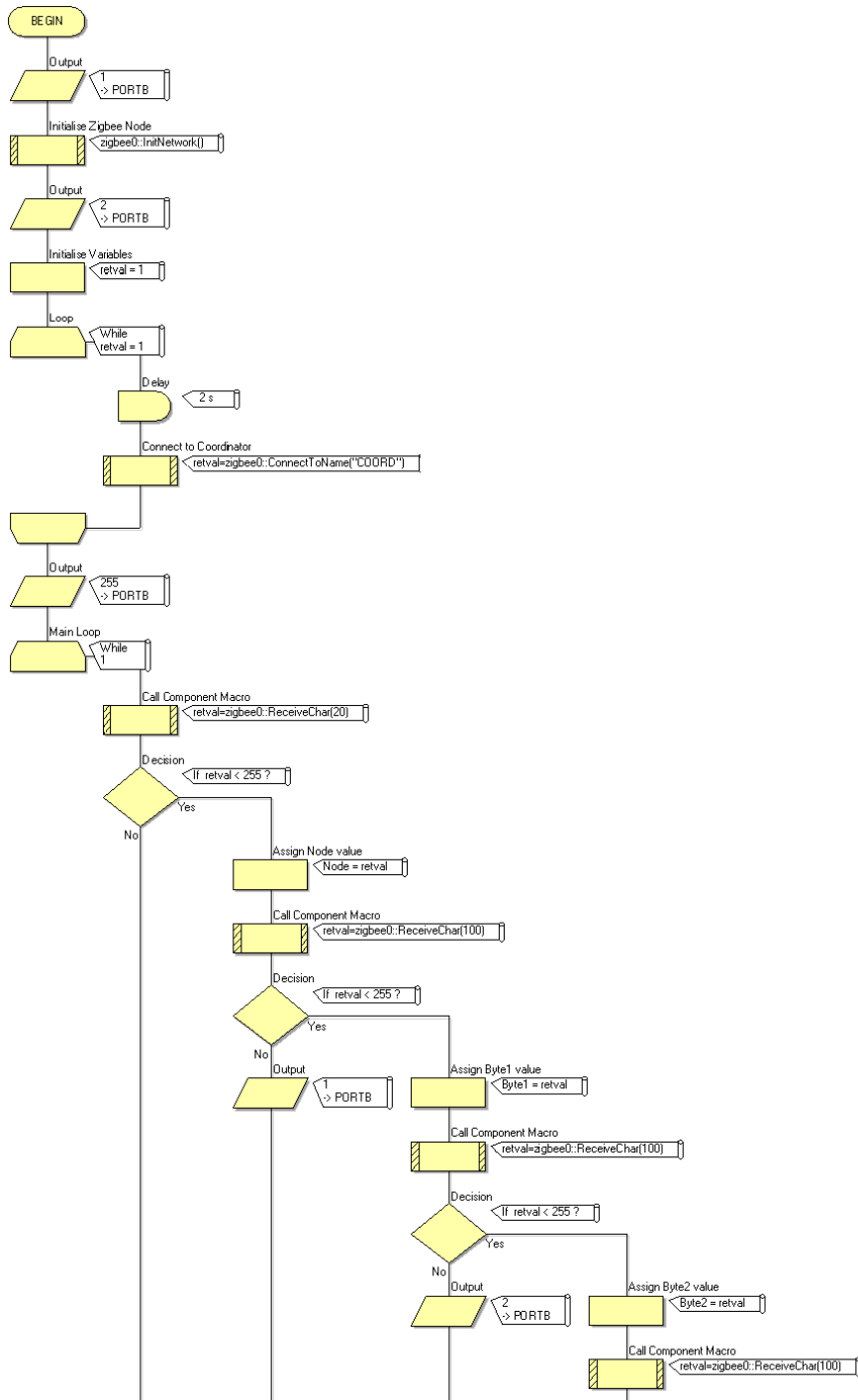
Exercise 6 – node 2



Exercise 6 – node 3

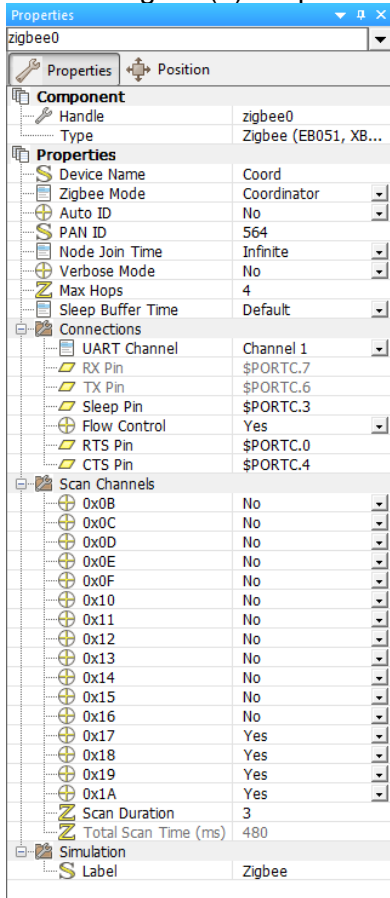


Exercise 6 – node 4



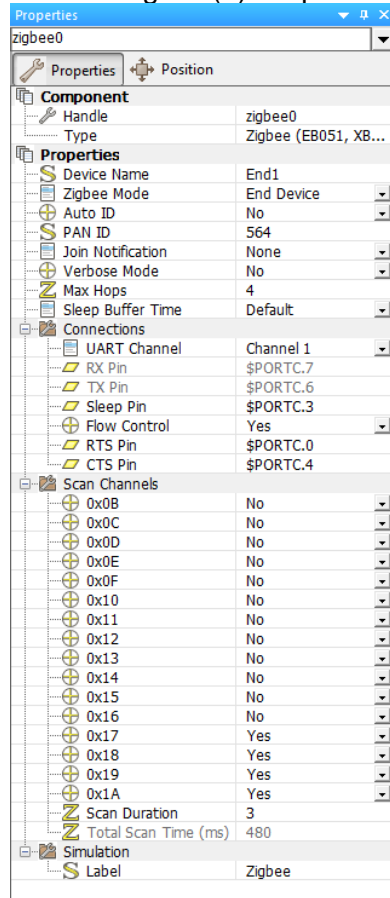
Exercise 7

Node 1 ZigBee(0) Properties



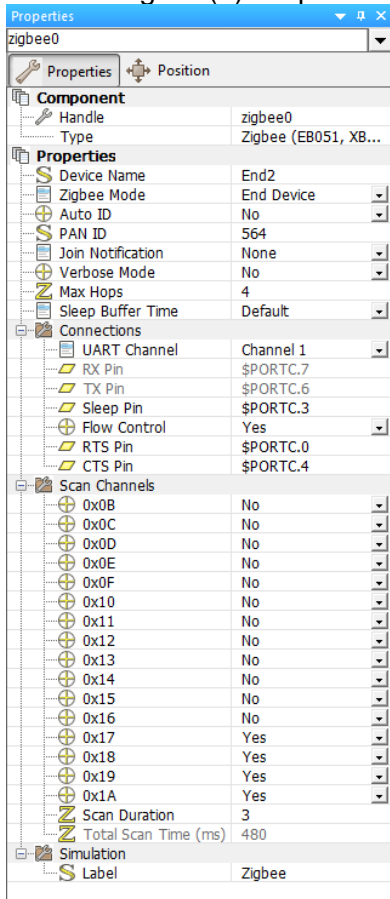
Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	Coord
Zigbee Mode	Coordinator
Auto ID	No
PAN ID	564
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

Node 2 ZigBee(0) Properties



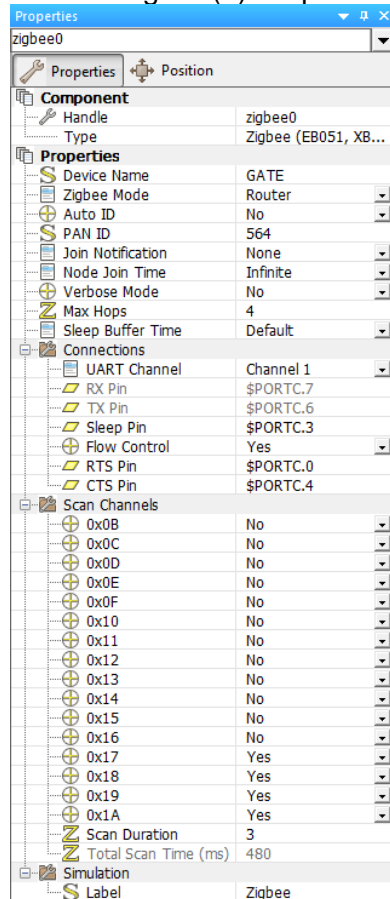
Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	End1
Zigbee Mode	End Device
Auto ID	No
PAN ID	564
Join Notification	None
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

Node 3 ZigBee(0) Properties



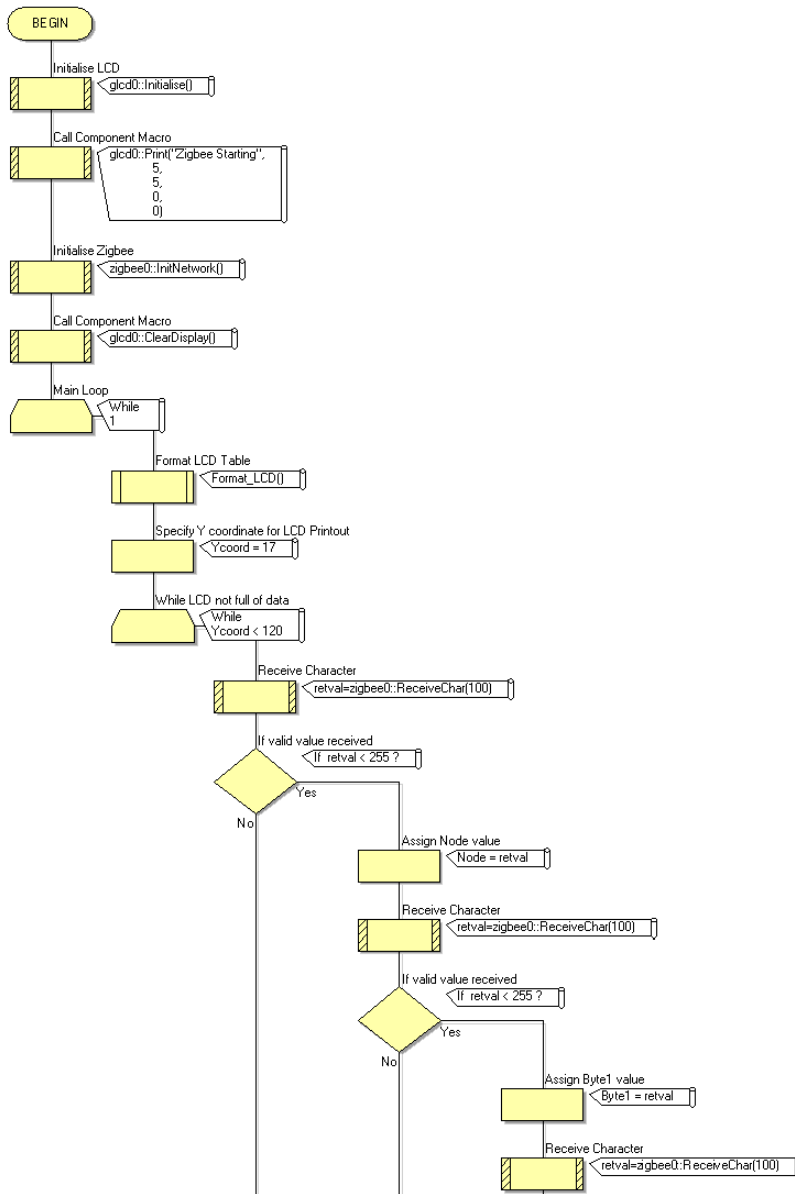
Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	End2
Zigbee Mode	End Device
Auto ID	No
PAN ID	564
Join Notification	None
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

Node 4 ZigBee(0) Properties

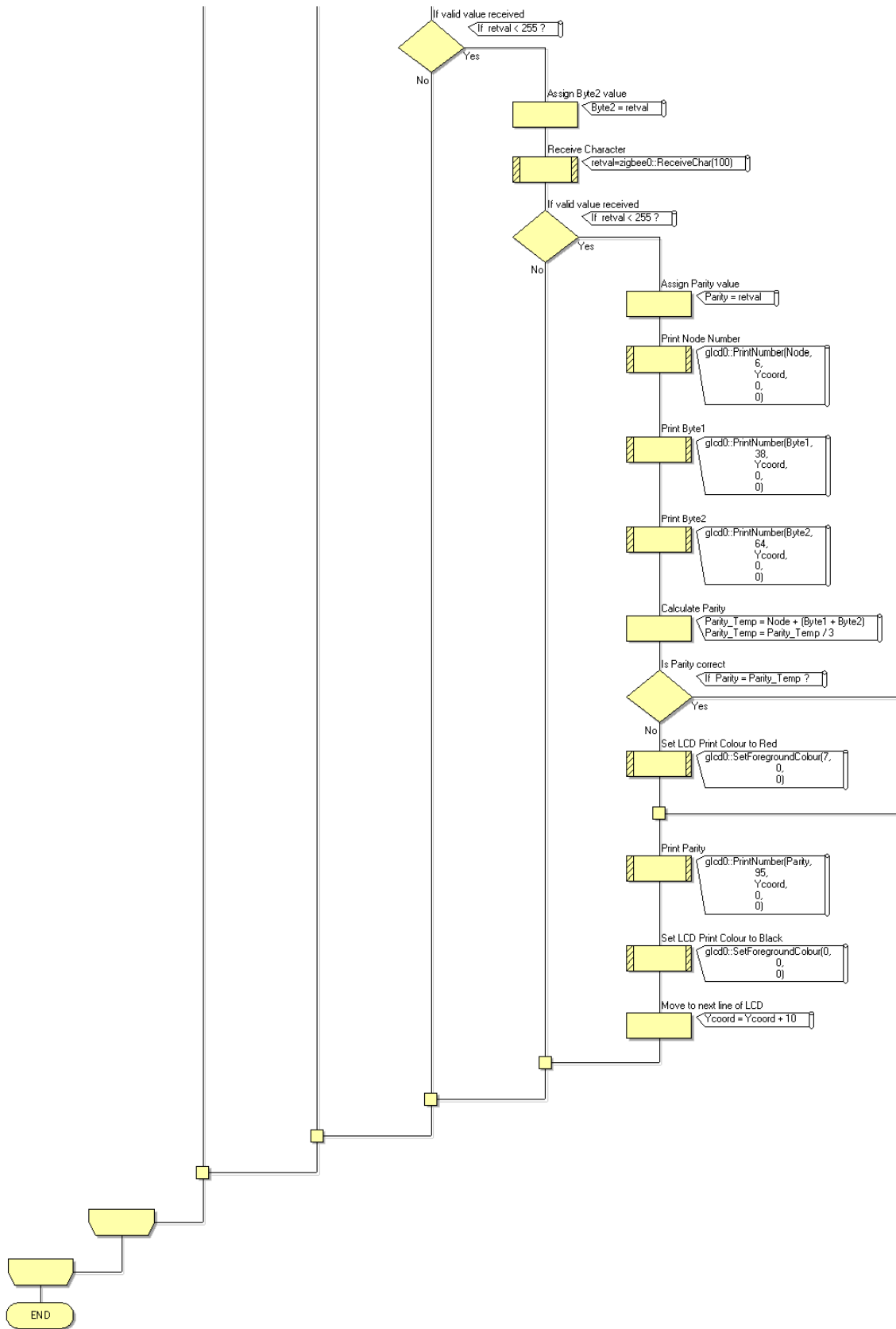


Property	Value
Handle	zigbee0
Type	Zigbee (EB051, XB...)
Device Name	GATE
Zigbee Mode	Router
Auto ID	No
PAN ID	564
Join Notification	None
Node Join Time	Infinite
Verbose Mode	No
Max Hops	4
Sleep Buffer Time	Default
UART Channel	Channel 1
RX Pin	\$PORTC.7
TX Pin	\$PORTC.6
Sleep Pin	\$PORTC.3
Flow Control	Yes
RTS Pin	\$PORTC.0
CTS Pin	\$PORTC.4
0x0B	No
0x0C	No
0x0D	No
0x0E	No
0x0F	No
0x10	No
0x11	No
0x12	No
0x13	No
0x14	No
0x15	No
0x16	No
0x17	Yes
0x18	Yes
0x19	Yes
0x1A	Yes
Scan Duration	3
Total Scan Time (ms)	480
Label	Zigbee

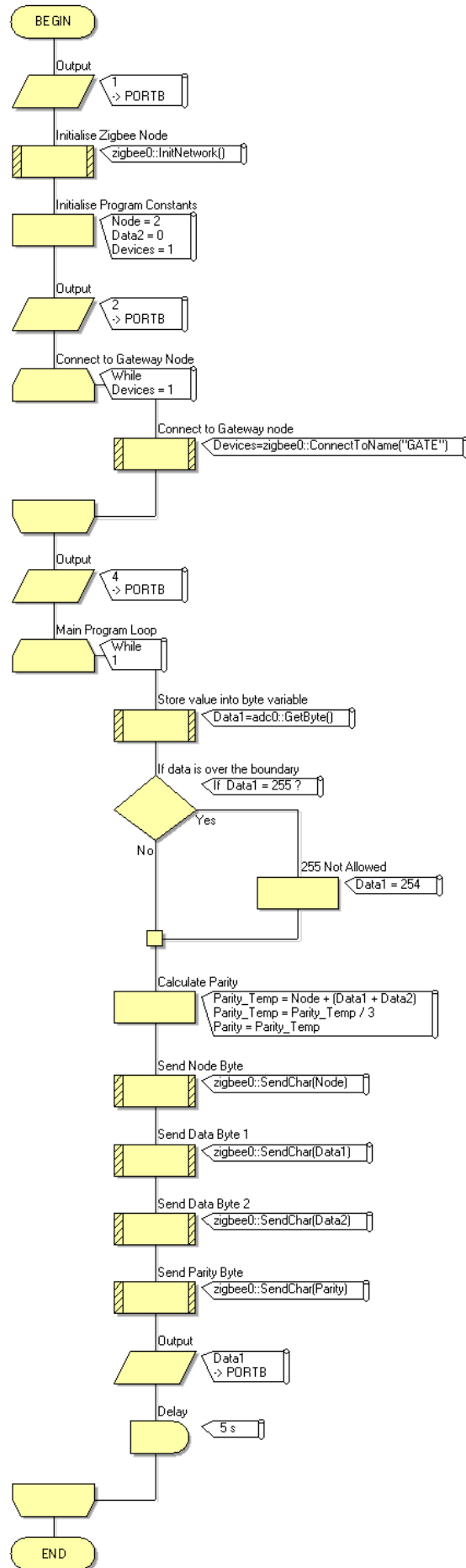
Exercise 7 – Node 1



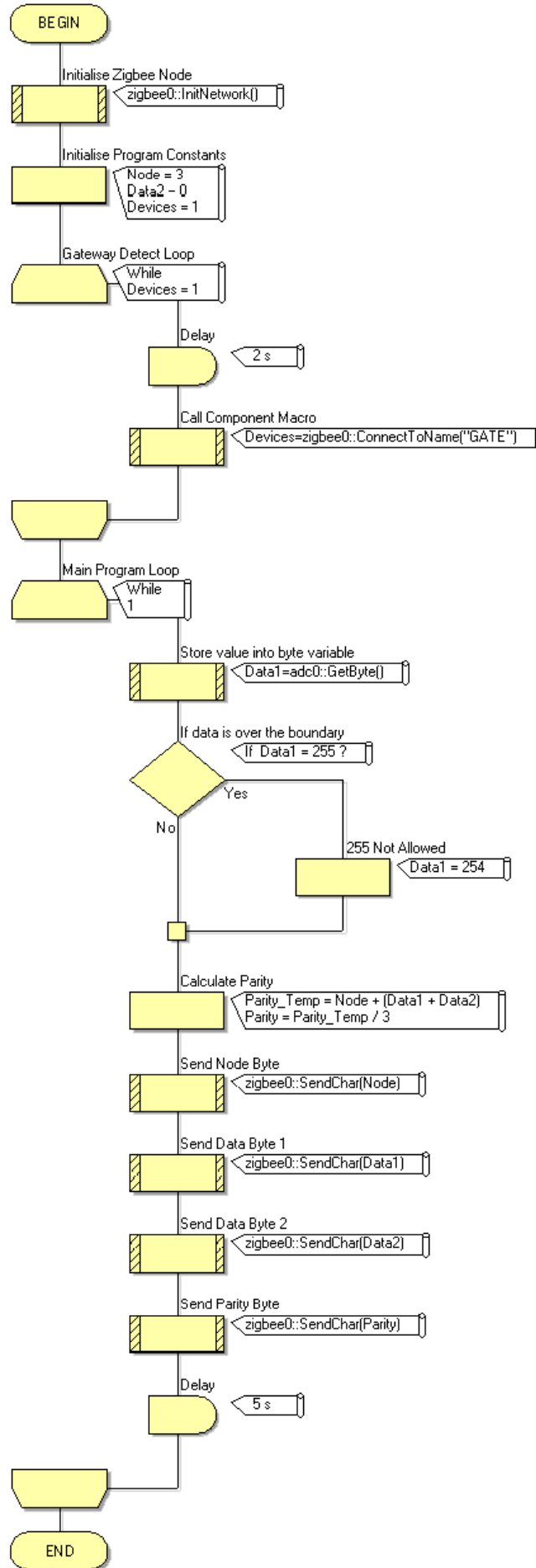
Exercise 7 – Node 1 continued



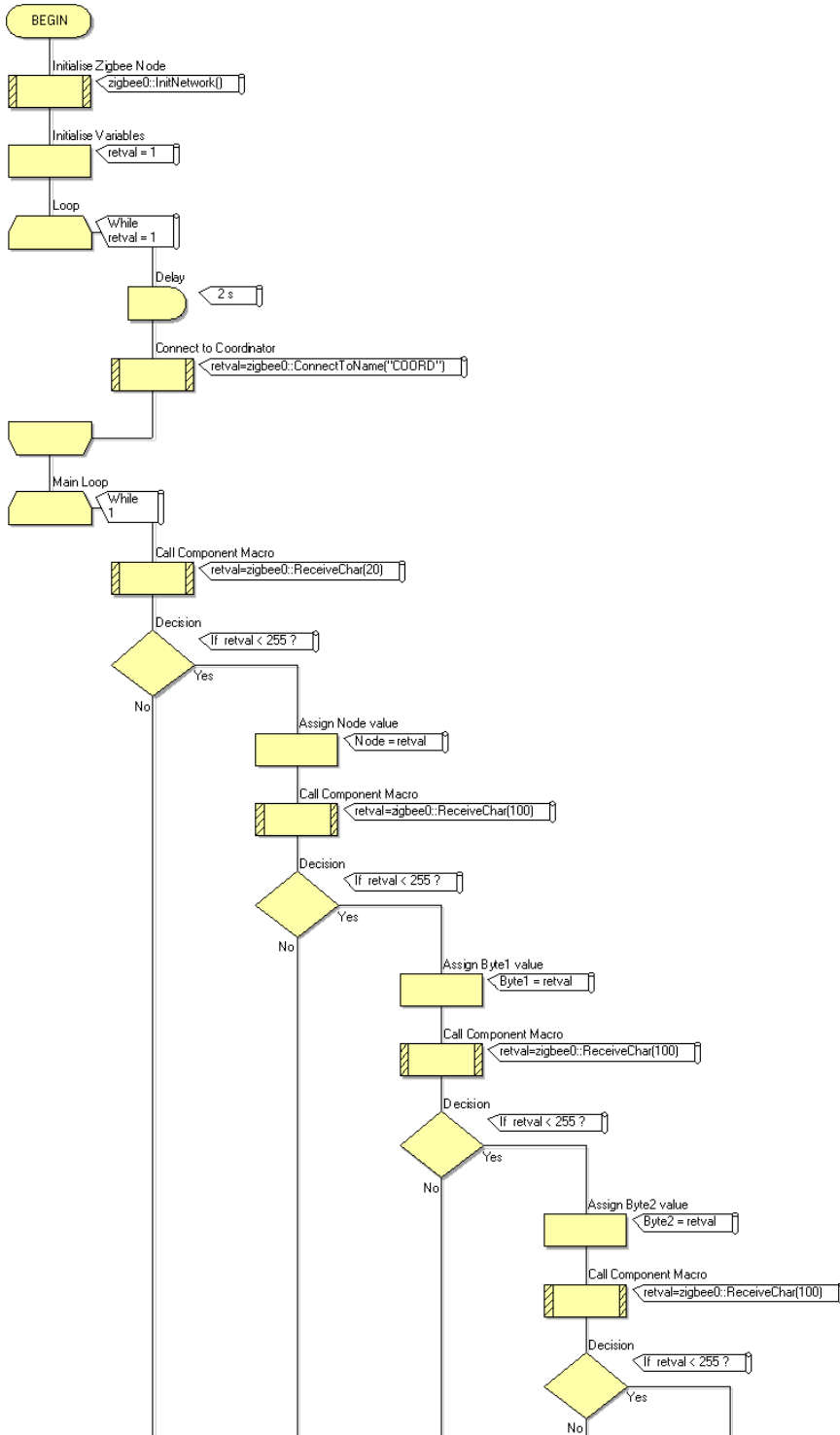
Exercise 7 – Node 2



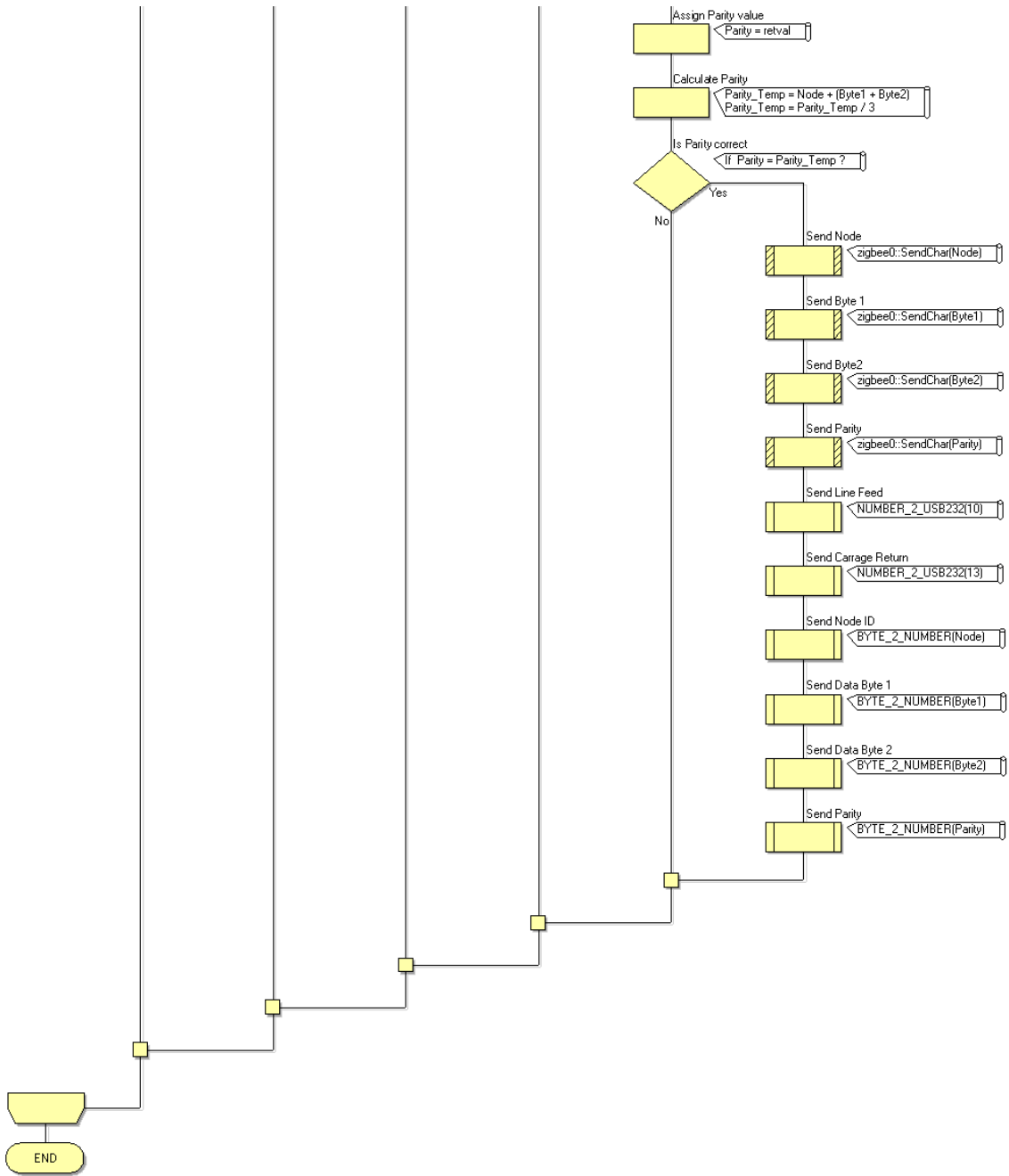
Exercise 7 – Node 3



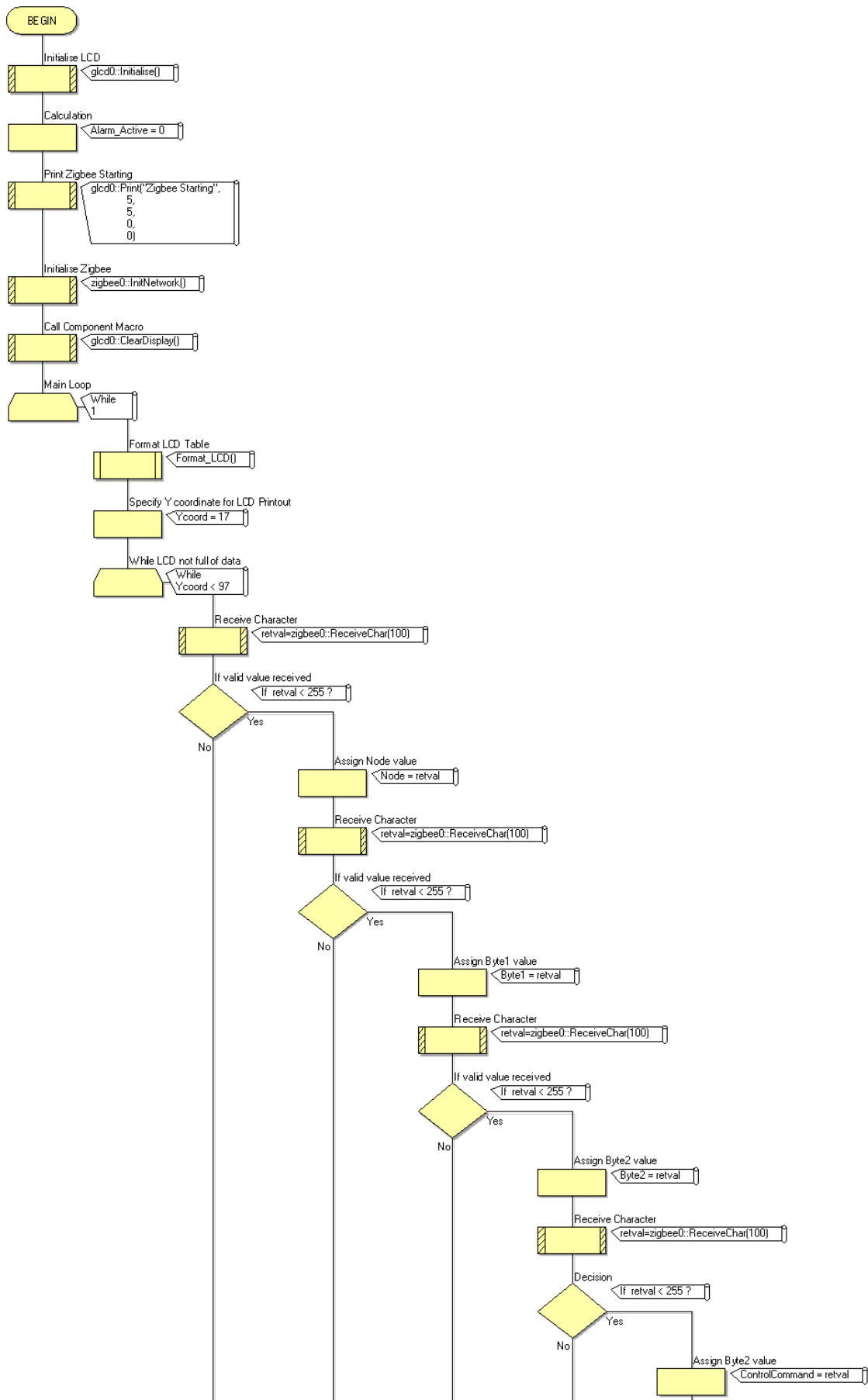
Exercise 7 – Node 4



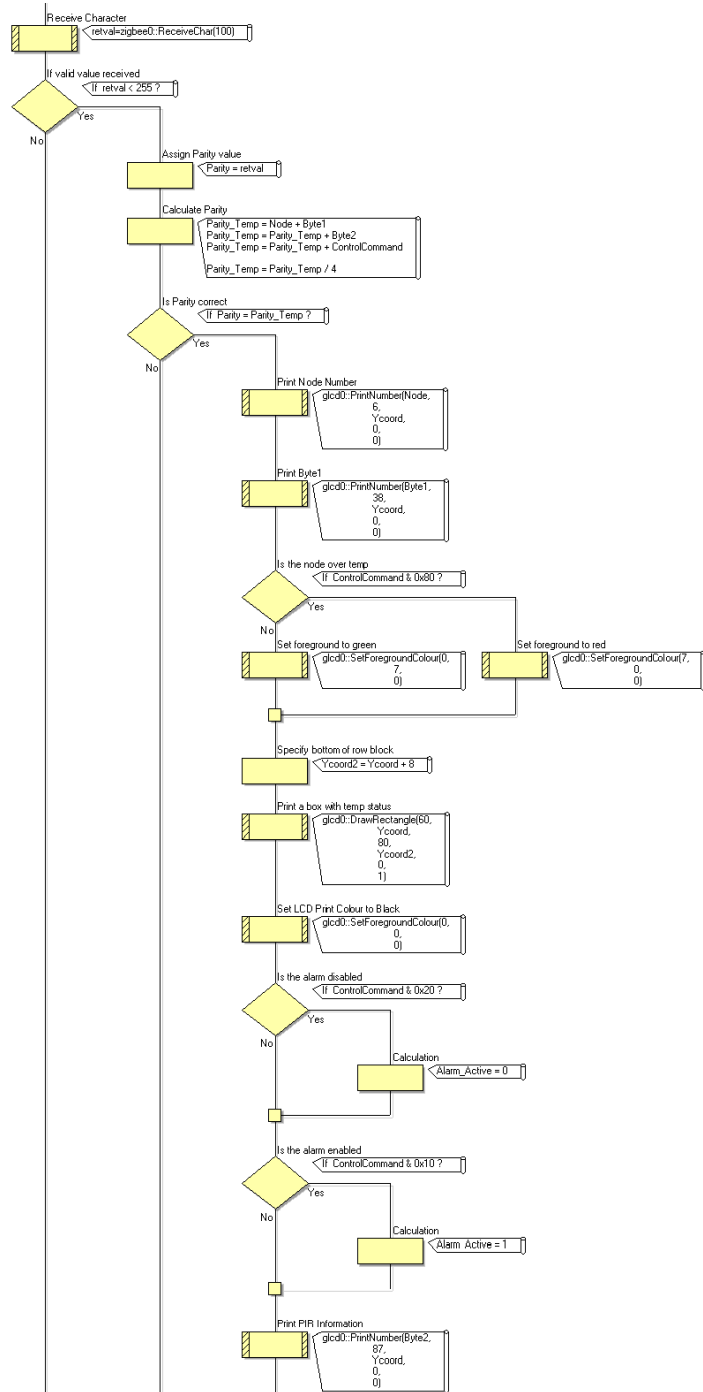
Exercise 7 – Node 4 continued



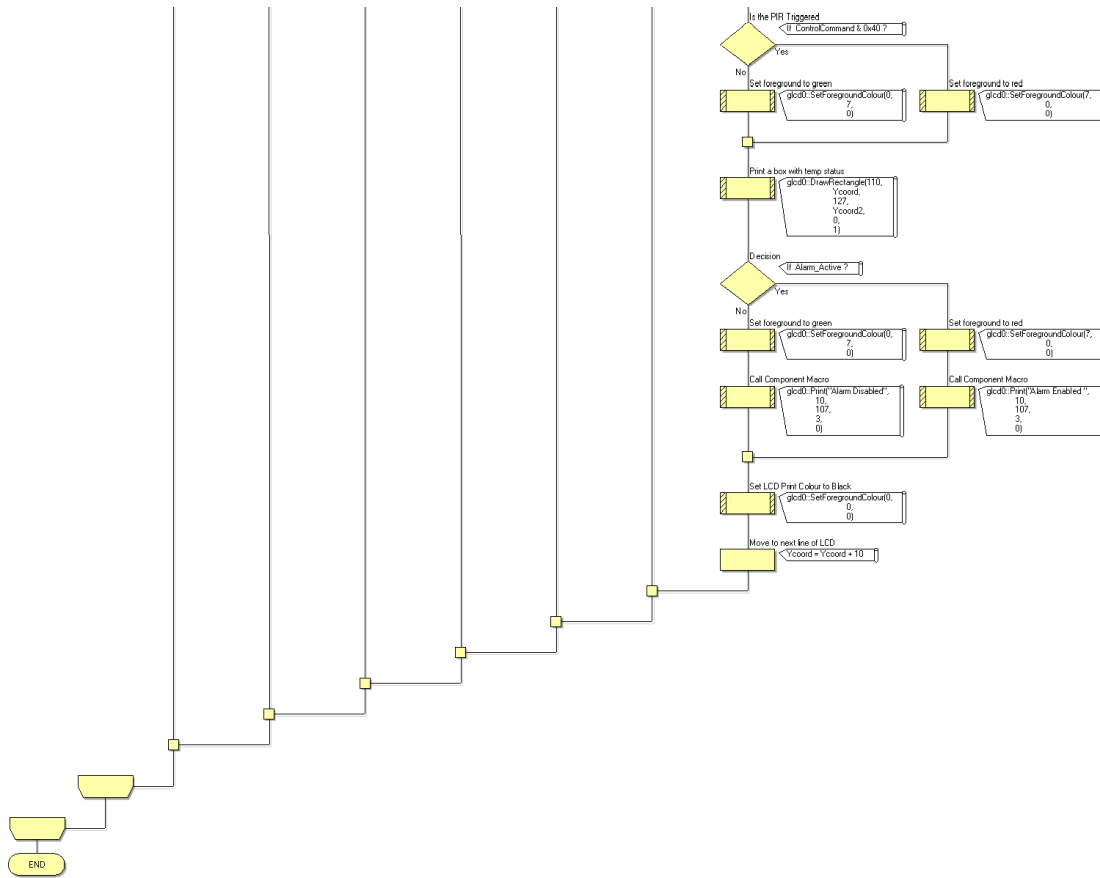
Exercise 8 – Node 1



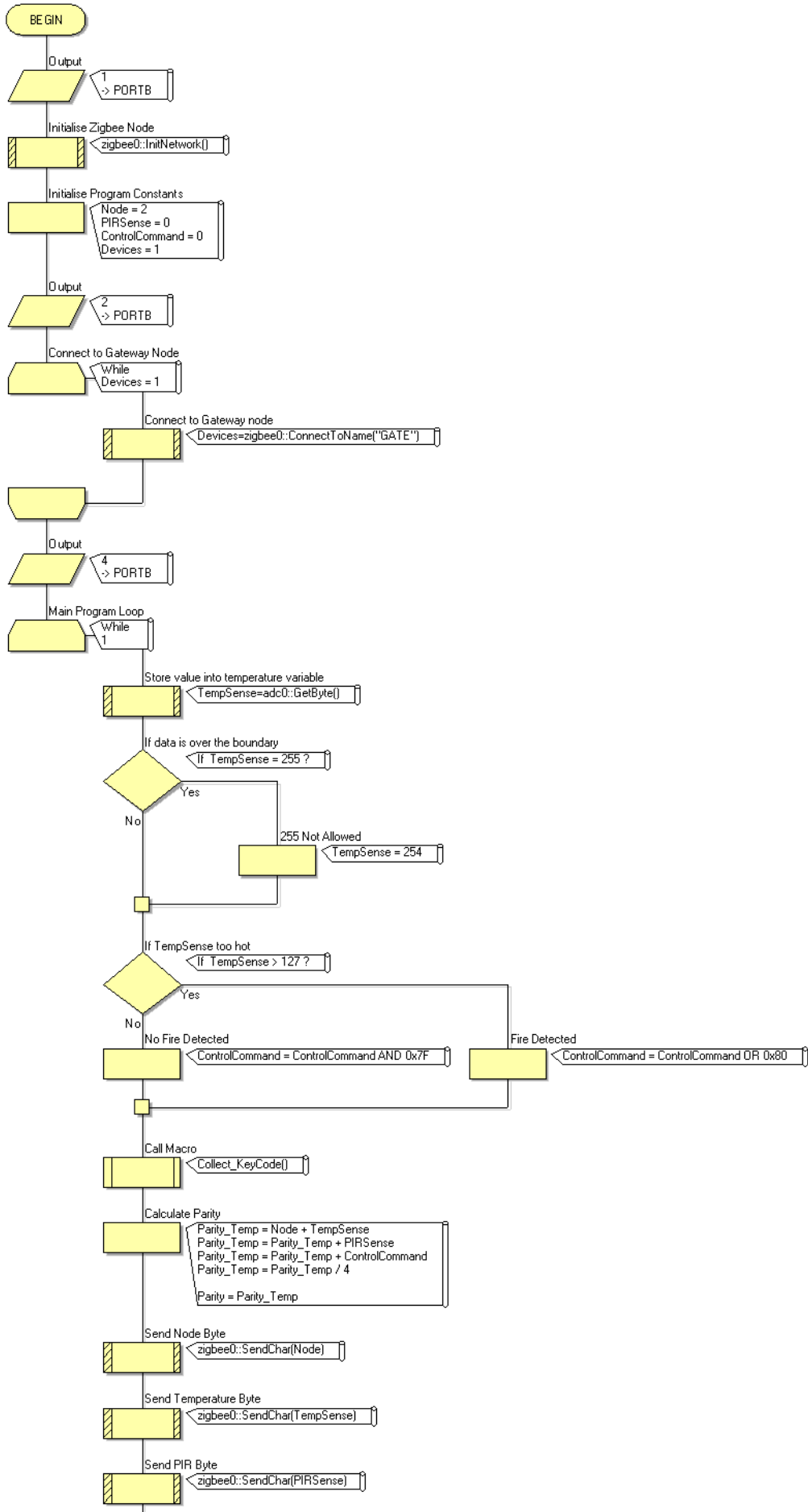
Exercise 8 – Node 1 continued



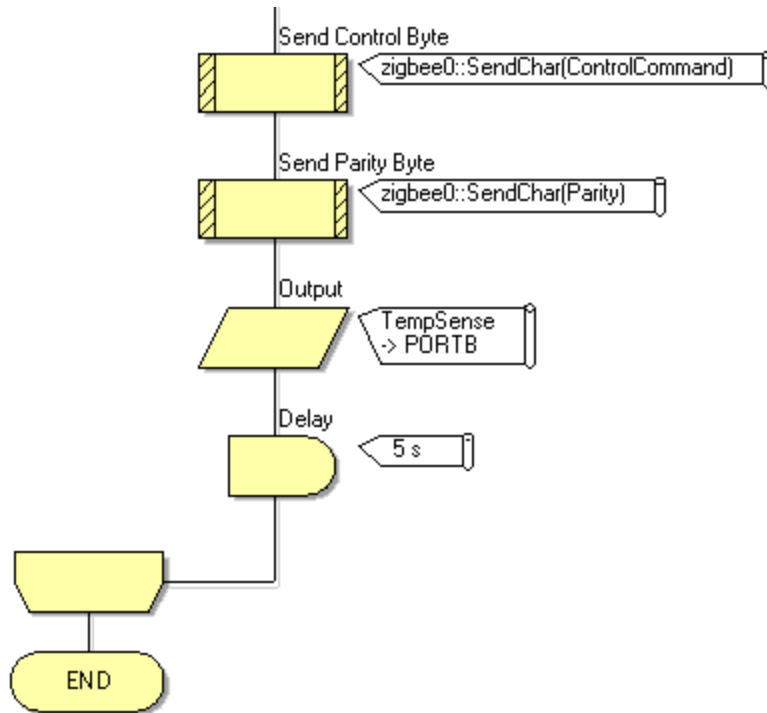
Exercise 8 – Node 1 continued



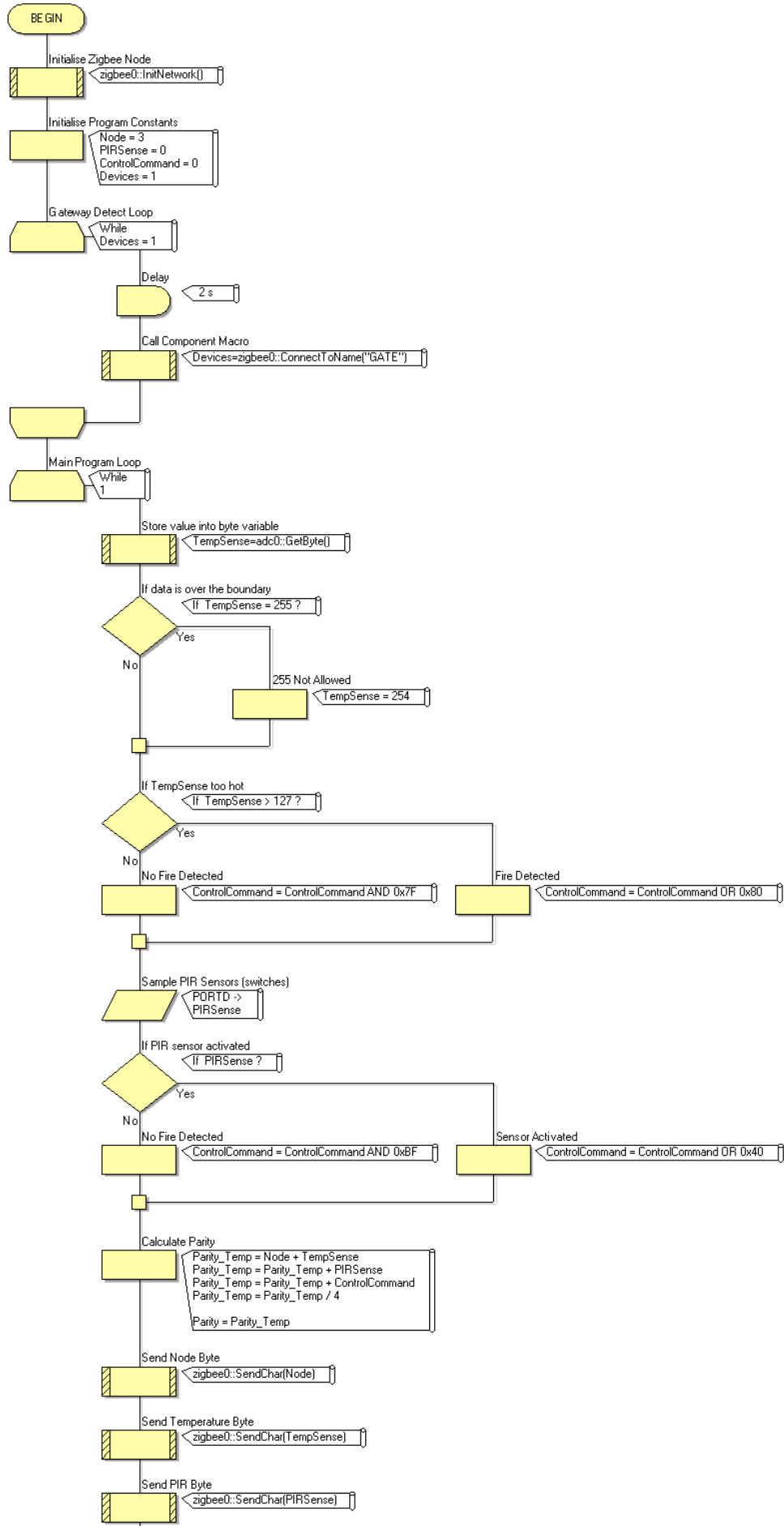
Exercise 8 – Node 2



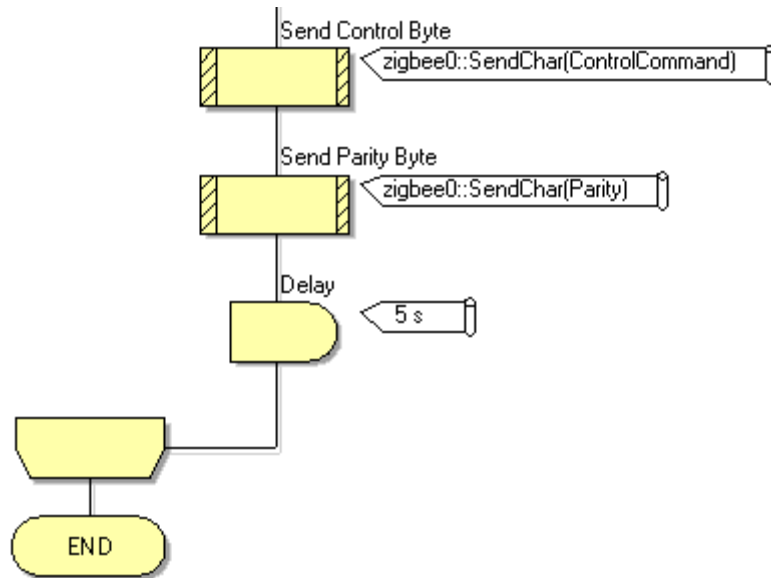
Exercise 8 – Node 2 continued



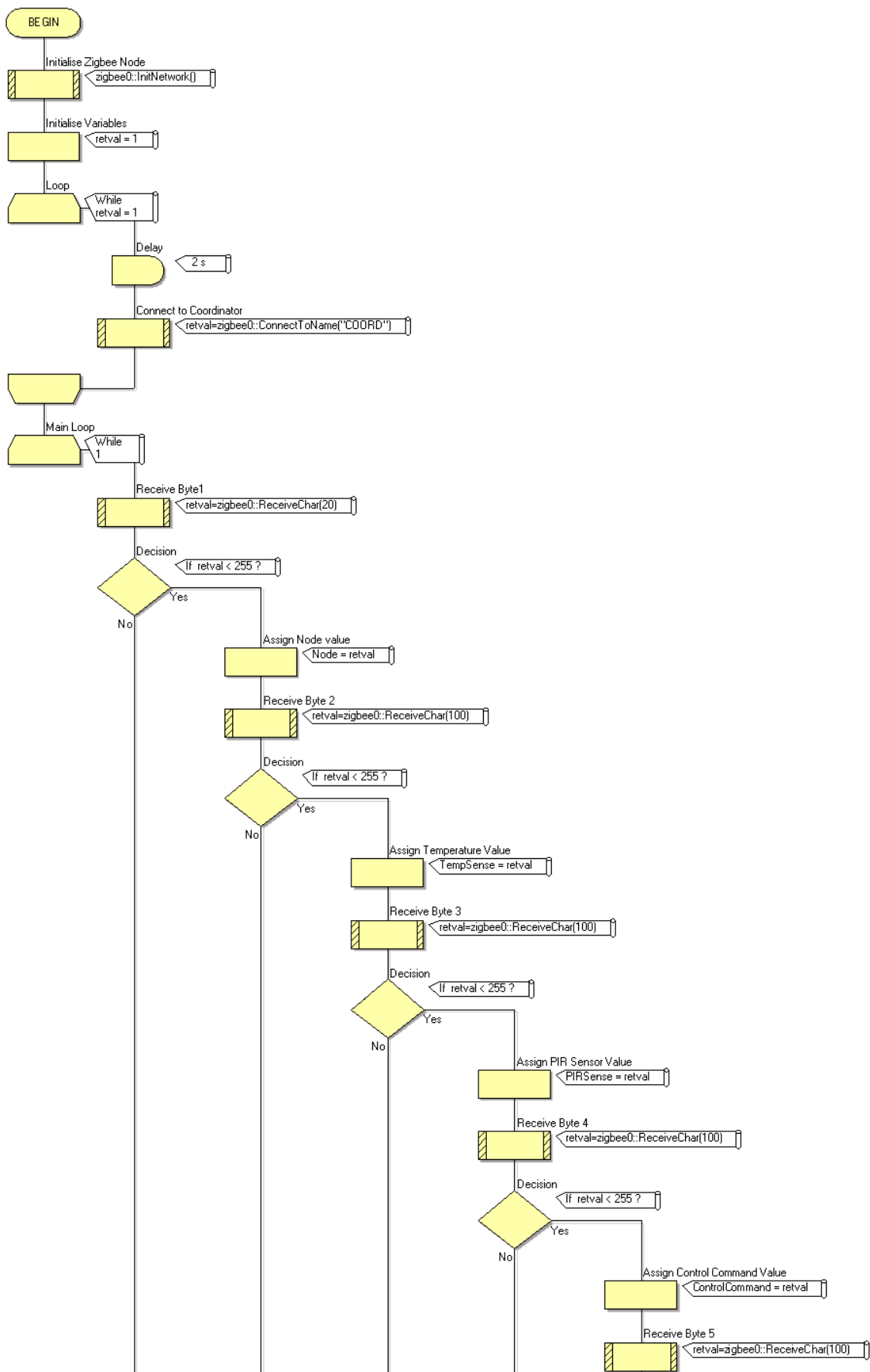
Exercise 8 – Node 3



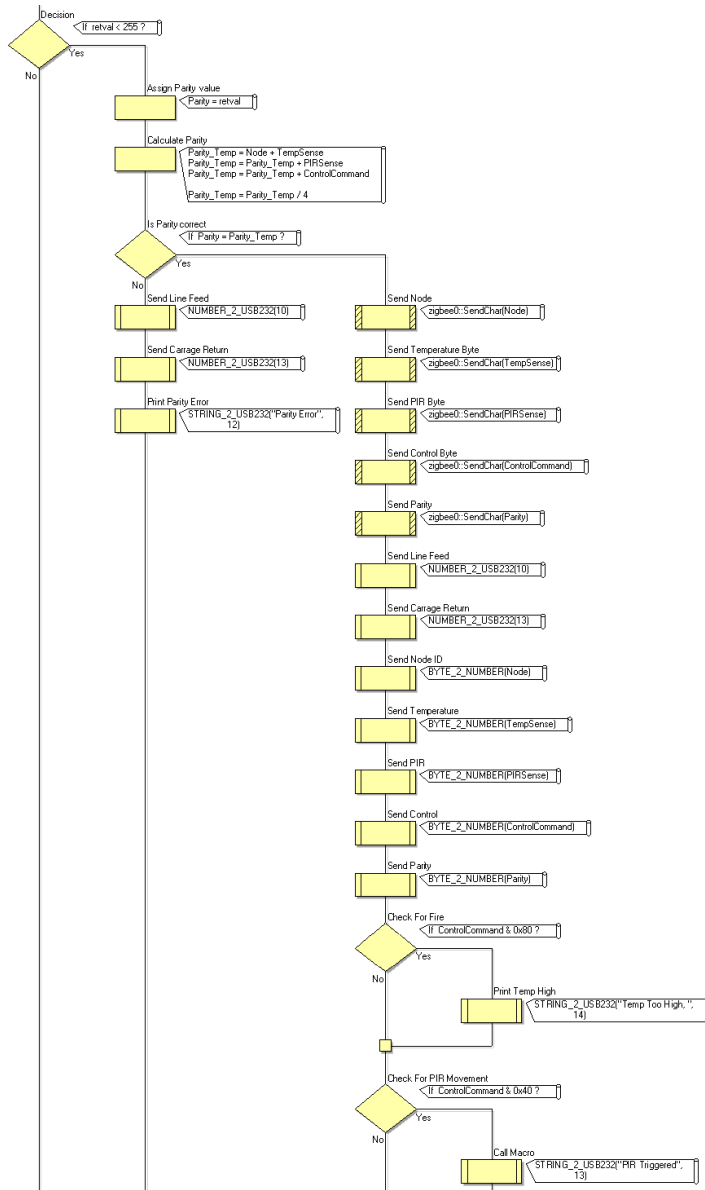
Exercise 8 – Node 3 continued



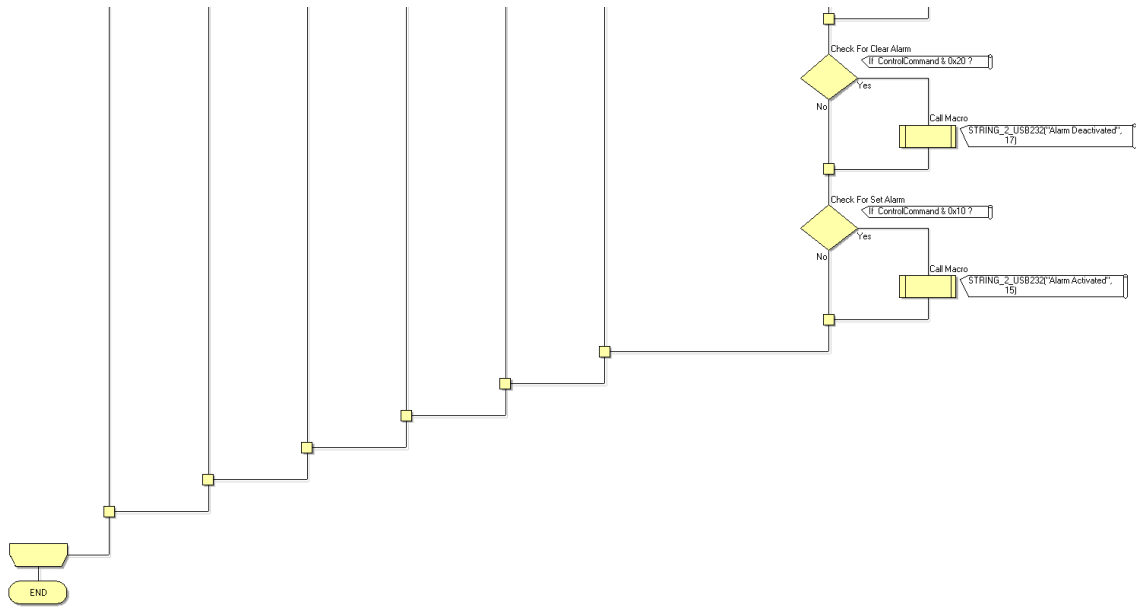
Exercise 8 – Node 4



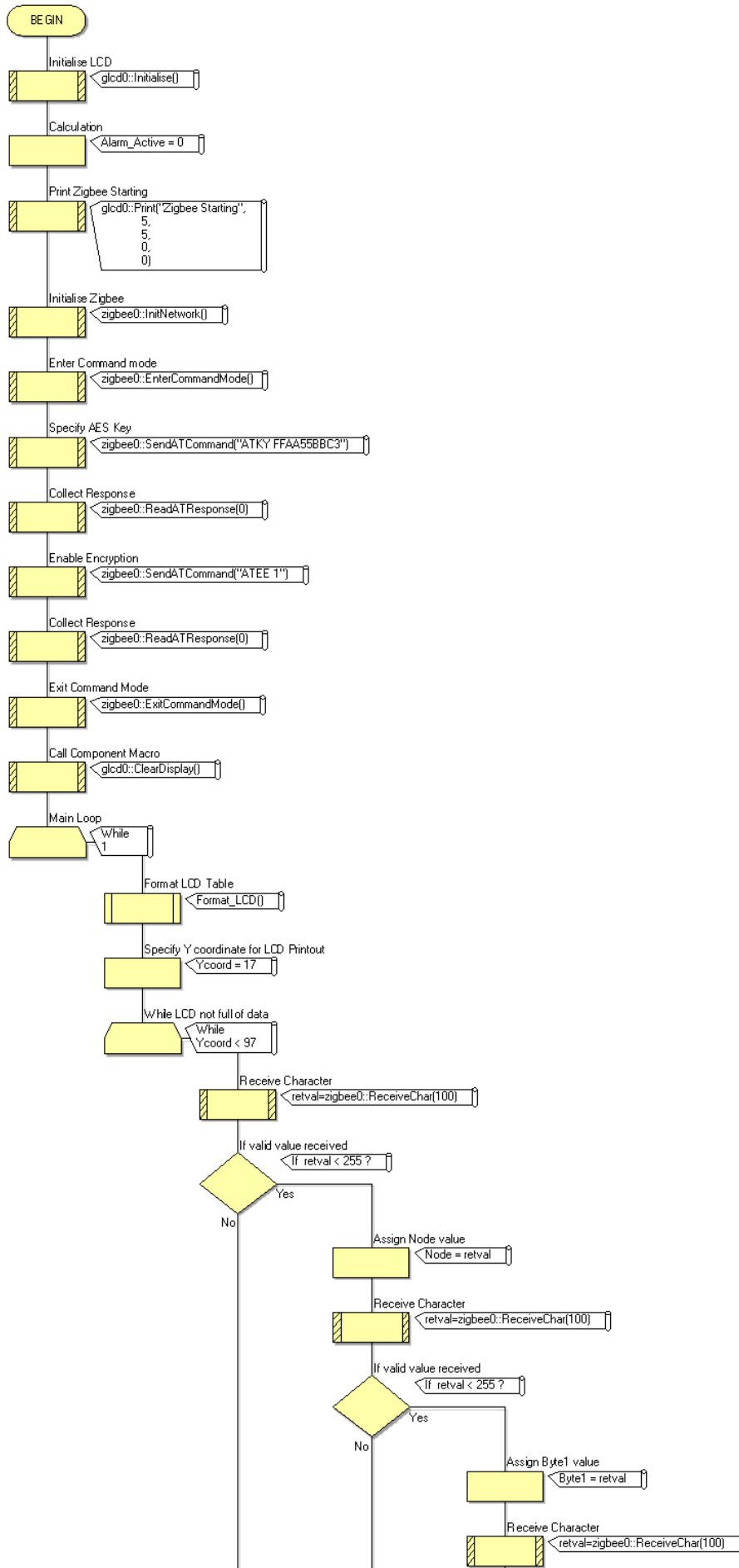
Exercise 8 – Node 4 continued



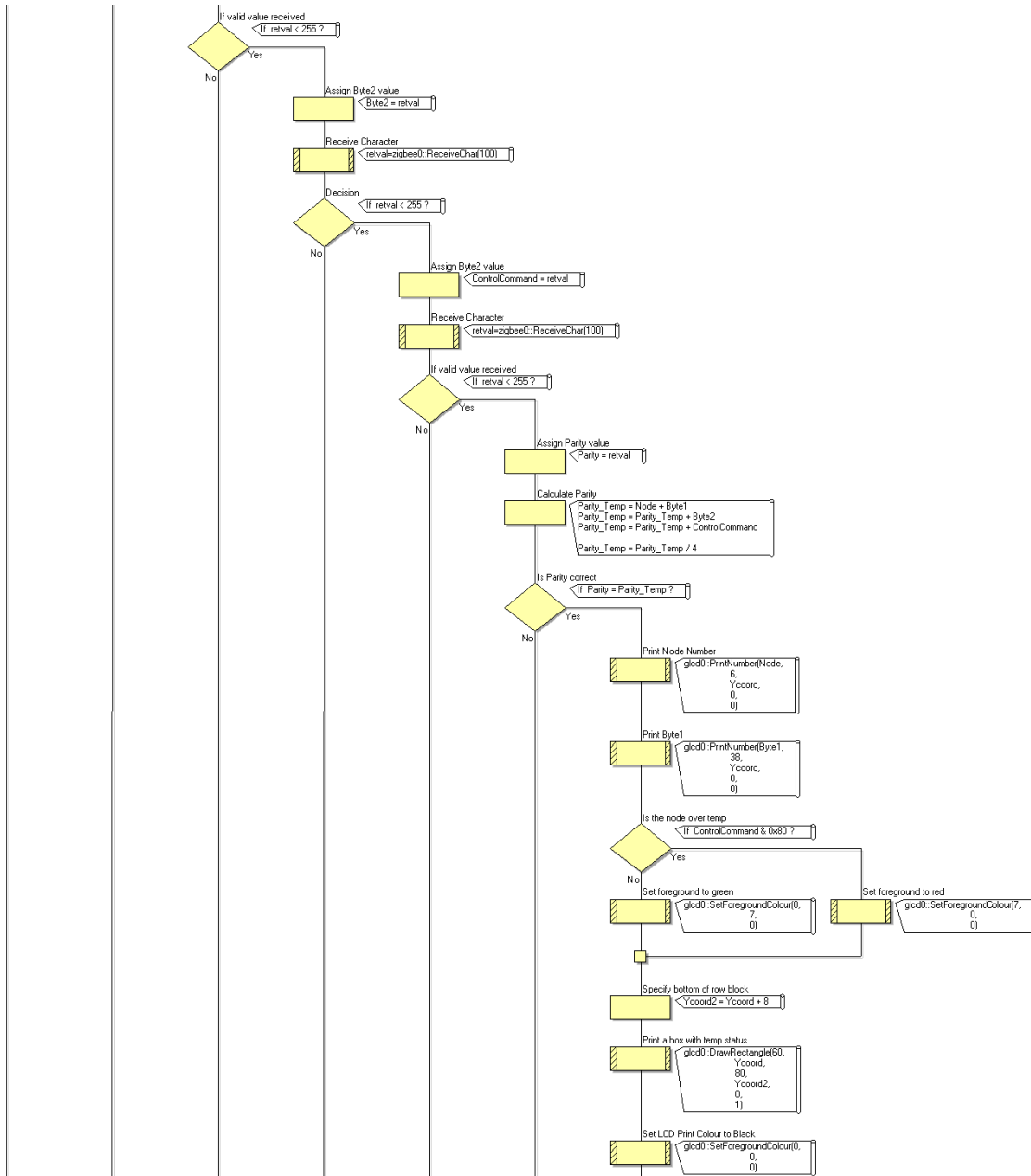
Exercise 8 – Node 4 continued



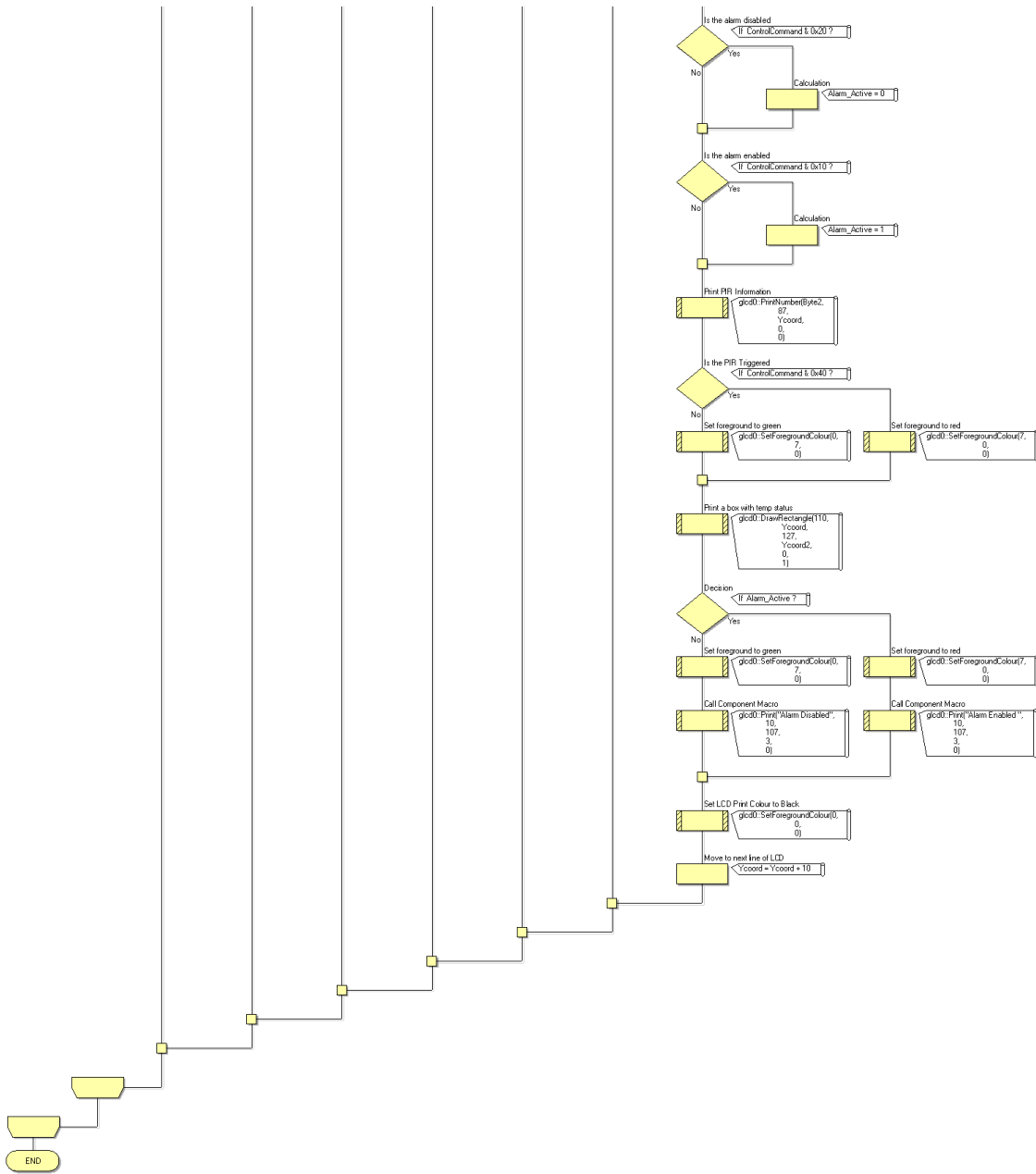
Exercise 9 – Node 1



Exercise 9 – Node 1 continued



Exercise 9 – Node 1 continued



The OSI Model

Layer	Characteristics	Data encapsulated in
7. Application	<ul style="list-style-type: none"> • Is closest to user applications • Provides network service to the operating system and applications such as web browser, file transfer, email etc. 	
6. Presentation	<ul style="list-style-type: none"> • Sets up data structures • Decides on data format, ASCII, EBCDIC, JPEG etc • Negotiates syntax, encoding and encryption • Ensures compatibility for data arriving from network 	
5. Session	<ul style="list-style-type: none"> • Establishes / manages / terminates sessions with destination host • Controls full-duplex / half-duplex etc nature of communication 	
4. Transport	<ul style="list-style-type: none"> • Establishes / maintains / terminates virtual circuits between source and destination during communication • Controls flow of information • Does error detection and correction 	Segments
3. Network	<ul style="list-style-type: none"> • Determines best path between source and destination • Uses network (logical) addressing scheme e.g. IP addresses 	Packets
2. Data Link	<ul style="list-style-type: none"> • Controls access to media in multi-access environments • Uses physical addressing e.g. MAC addressing scheme • Is concerned with physical topology of the network 	Frames
1. Physical	<ul style="list-style-type: none"> • Deals with the physical and electrical properties of the signal and of the media, such as voltage levels, connector types, cable specifications, etc. • Sends and receives bits via the connecting medium 	Bits

The RS232 Protocol

RS-232 is a telecommunications standard dating from the 1960's, defined originally for use in teletypewriters and still in widespread use. For example, it is the basis for data transfer from a computer's 9-pin serial and 25-pin parallel ports.

It appears in a number of different forms, such as EIA/TIA232, RS-232D, V.24, V.28, X20, and X21. It is used in both asynchronous data transfer and synchronous links such as HDLC, Frame Relay and X.25.

Scope

It includes not only electrical specifications, and definitions of the signals used, but also pin outs for a range of connectors such as 9 and 25 pin D-type connectors and RJ45 connectors.

In its native form, logic voltage levels are -15 to -3V for a logic 1 (mark), and +3 to +15V for a logic 0 (space). TTL based RS232 makes use of an inverting level-converter IC to change from TTL voltage levels to those valid for RS232.

Jargon!

Devices which use serial cables for their communication are split into two categories, DCE (Data Communications Equipment) and DTE (Data Terminal Equipment.) Data Communications Equipment includes devices such as an analogue modem, TA adapter (on an ISDN line), CSU/DSU (Channel Service Unit / Data Service Unit – a digital modem, in effect) etc., while Data Terminal Equipment is often a computer or router. Usually, the DCE device controls the flow of data between the DCE and the DTE by providing synchronisation signals or timing signals. The DTE device is also known as the data terminal, whereas the DCE device is the data set.

Confusion can arise over the pin descriptions TD (Transmit Data) and RD (Receive Data). In reality, both pins may 'transmit' data and 'receive' data at times, depending on whether they are located on the DTE or the DCE device. The solution is to look at these pins from the viewpoint of the DTE device. The DTE device transmits data on the TD line. When the DCE device receives this data, it receives it on the TD line as well! When the modem or CSU/DSU receives data from the outside world and sends it to the DTE, it sends it on the RD line because from the viewpoint of the DTE, the data is being received!

Signalling overview

Data is transmitted from the microcontroller to the XBee module using the TX connection. Data is returned from the XBee module using the RX connection.

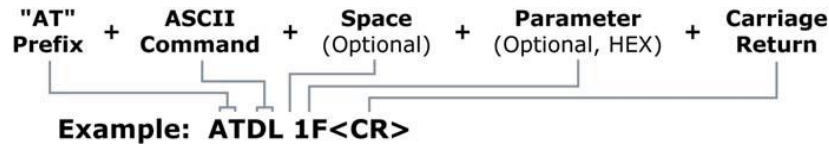
Request To Send (RTS) and Clear To Send (CTS) signals are used to provide hardware flow control. The XBee module can prevent the microcontroller from transmitting data by holding the CTS line high. The microcontroller can prevent the XBee module from transmitting data by holding the RTS line high.

AT commands

The Version2 XBee modules are controlled by a set of ASCII commands known as AT commands. The modules are set by default into data transfer mode and must be placed 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. When the command sequence is entered correctly then the module returns an "OK" string.

The diagram shows an example of an AT command.



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

Any data sent to the Version2 XBEE module while it is in data transfer mode will result in transmission to the destination address specified in the module. By default a destination node is not specified and so any data is transferred as a broadcast message to all nodes. If a destination node is specified then data will be transferred as a unicast message to the specified node.

Useful AT Commands

(A full list of AT commands is given in the XBee User Guide)

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

Further information

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

A range of resources is found in the White Papers section -
<http://zigbee.org/LearnMore/WhitePapers.aspx>

An overview of ZigBee appears at –
<http://pages.cs.wisc.edu/~suman/courses/838/papers/zigbee.pdf>

An e-learning course covering the basics of ZigBee is found at –
<http://www.jennic.com/elearning/zigbee/index.htm>

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

For information about modulation techniques see –

http://www.maxim-ic.com/appnotes.cfm/appnote_number/686
http://www.techbooksforfree.com/intro_to_data_com/page88.html

For information about XBee modules see –
<http://www.digi.com/support/>

Wireless Development Studio –
<http://ww1.microchip.com/downloads/en/DeviceDoc/WdsApp-3.2-windows-installer.zip>

ZENA™ Wireless Adapter User's Guide -
<http://ww1.microchip.com/downloads/en/DeviceDoc/70664B.pdf>

EB538

ZigBee Solution

Student Guide

Contents

About this course:	4
1. Introduction to ZigBee	5
1.1 Overview	5
1.2 Comparison of Wireless Technologies	6
1.3 ZigBee Applications	7
1.4 ZigBee protocol outline	8
1.5 The IEEE 802.15.4 Standard	9
2. The ZigBee network	12
2.1 ZigBee Logical Devices	12
2.2 ZigBee Network Topologies	13
2.3 Multi-Access Networks	14
2.4 Creating a ZigBee network	15
2.5 ZigBee Addressing Scheme	16
2.6 Routing and route discovery	18
2.7 Sleep mode	19
2.8 Security	20
3. The Matrix ZigBee training solution	21
3.1 The ZigBee E-Block	21
3.1.1 AT Commands	22
3.2 Installation	22
4. The ZigBee assignments	26
5. Exercise 1 – Moulding the network	27
5.1 Introduction	27
5.2 Objective	27
5.3 Requirements	27
5.4 The Flowcode program in detail	27
5.5 What to do	30
5.6 Further work	32
6. Exercise 2 – Adding a Node	33
6.1 Introduction	33
6.2 Objective	33
6.3 Requirements	33
6.4 The Flowcode program in detail	33
6.5 What to do	36
6.6 Further work	40
7. Exercise 3 – Expanding the network	41
7.1 Introduction	41
7.2 Objective	41
7.3 Requirements	41
7.4 The Flowcode program in detail	41
7.5 What to do	43
7.6 Further work	45
8. Exercise 4 – Reducing power consumption	46
8.1 Introduction	46
8.2 Objective	46
8.3 Requirements	46
8.4 The Flowcode program in detail	47

- 8.5 What to do..... 48
- 8.6 Further work..... 49
- 9 Exercise 5 – Dynamic networks 50
 - 9.1 Introduction 50
 - 9.2 Objective 50
 - 9.3 Requirements..... 50
 - 9.4 The Flowcode program in detail 51
 - 9.5 What to do..... 53
 - 9.6 Further work..... 55
- 10 Exercise 6 – Message Routing..... 56
 - 10.1 Introduction 56
 - 10.2 Objective 56
 - 10.3 Requirements..... 56
 - 10.4 The Flowcode program in detail 57
 - 10.5 What to do..... 58
 - 10.6 Further work..... 59
- 11 Exercise 7 – Datalogging Gateway 60
 - 11.1 Introduction 60
 - 11.2 Objective 60
 - 11.3 Requirements..... 60
 - 11.4 The Flowcode program in detail 60
 - 11.5 What to do..... 64
 - 11.6 Further work..... 64
- 12 Exercise 8 – Modular fire and burglar alarm system..... 65
 - 12.1 Introduction 65
 - 12.2 Objective 65
 - 12.3 Requirements..... 65
 - 12.4 The Flowcode program in detail 66
 - 12.5 What to do..... 67
 - 12.6 Further work..... 67
- 13 Exercise 9 – Improving network security 68
 - 13.1 Introduction 68
 - 13.2 Objective 68
 - 13.3 Requirements..... 68
 - 13.4 The Flowcode program in detail 69
 - 13.5 What to do..... 70

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 version 8 or higher software
- Power supply
- USB programming lead
- E-blocks2 boards including:
 - Four BL0011 or BL0055 (with Arduino Uno fitted)
 - One BL0152C ZigBee co-ordinator
 - Three BL0152R ZigBee router / end-node
 - One BL0157 Graphical Colour LCD
 - One BL0145 Switches
 - One BL0138 Keypad
 - Two BL0167 LEDs
 - One BL0158 Connector
 - One BL0129 Grove interface with light and rotary sensors

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: To undertake all of the exercises on this course will take you around twelve hours.

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

1. Introduction to ZigBee

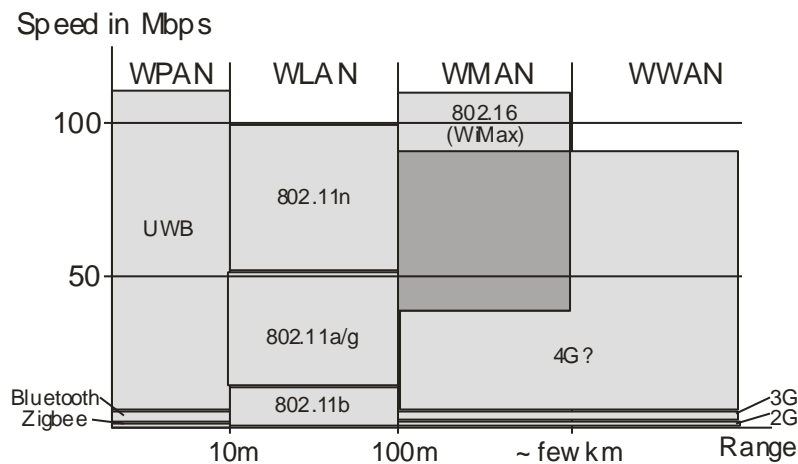
1.1 Overview

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)
- WWANs (Wireless Wide Area Networks.)

The speeds shown in the diagram are guides only.



WWANs are dominated by mobile phone (cell phone) technologies, known as 2G, 3G and, forthcoming, 4G.

WLANs try to provide all the features of wired LANs, but without the wires. They can cover areas ranging in size from a small room to an industrial complex. Public accessible wireless LANs, known as Wi-Fi Hotspots, are often found in airports, hotels, libraries and other public places.

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. This is a rapidly developing technology, making possible applications such as video and audio streaming wirelessly around the home between a base device and subsidiary devices. Medium rate WPANs (IEEE 802.15.1 / Bluetooth) 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 a transmission chain, receiving messages, deciding if they are for local use, and re-transmitting them to other nodes in the network if they are 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
- focuses on 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 Kbps at 915 Mhz, and 20 Kbps 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 lower 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;
- has a range of ~10 metres;
- allows for three modes of security;
- 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 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 network 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. Devices 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, whereas staff and visitors can walk freely in and out of a ZigBee-secured smart building, intruders will 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 incur the same costs as cable installation.

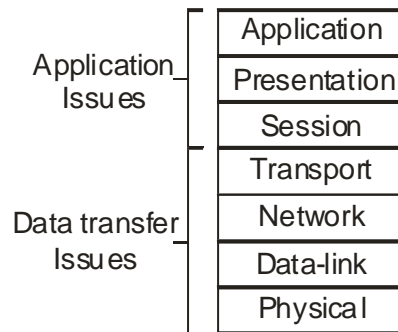
At home, the user has a digital key to unlock the door. The control system then identifies the user and adjusts the lighting, heating and curtains accordingly. While the house was empty, the control system recorded the output of security cameras around the property. It monitored sensors looking for water or gas leaks, for freezing pipes, and for fire. It monitored 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, and the outside lights flash to make it easier for the fire brigade to find the house.

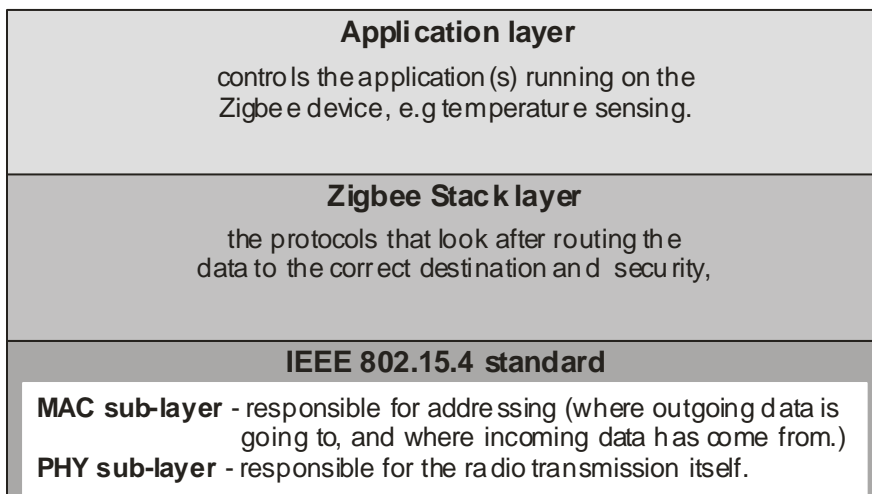
1.4 ZigBee protocol outline

Computer and computer-related networking is often simplified by using the OSI (Open System Interconnection) model. This divides the tasks involved in 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 both 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.



In a similar way, 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 ZigBee 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 IEEE 802.15.4 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 deals with broadband wireless Metropolitan Area Network communications.

Whereas both of these 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 specific 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° when the signal changes from logic 0 to logic 1, or back again. The next diagram illustrates this:

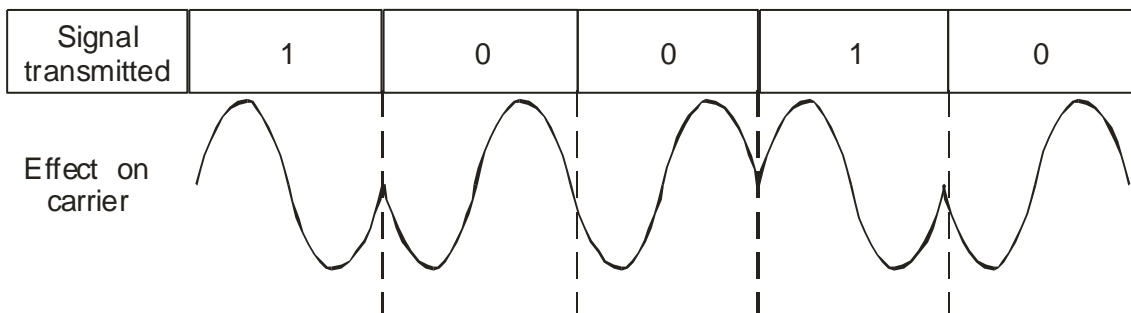
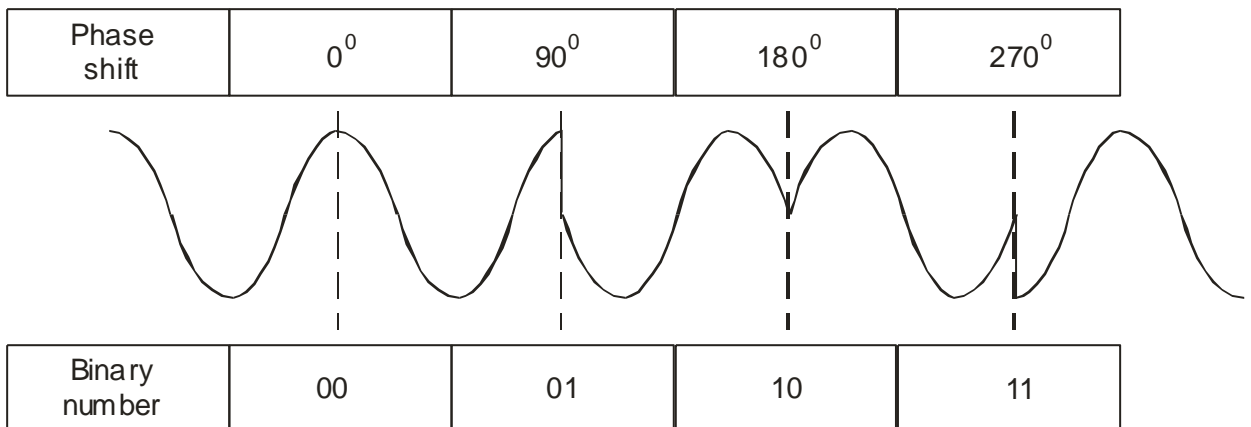


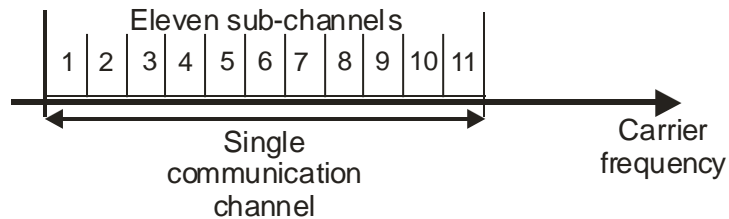
Illustration of PSK

QPSK (quadrature phase-shift keying) uses phase-shifts of 90° instead of 180° . 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 operate at low power and have limited memory capacity.

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:

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 battery powered

The RFD

- requires only limited RAM and ROM resources and is thus less expensive.
- 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.
- can talk only to a FFD.

The FFD

- can serve as a network co-ordinator, a router or as an end device.
- 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 on mains electricity, and so is usually located in a fixed position. As a result, it is often used 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 fixed in 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:

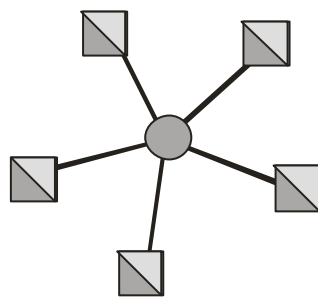
- This is used to communicate between the network and the real world, and is either a FFD or a RFD device. For example end devices can be placed into sensors, switches, displays, actuators etc.
- End devices are low in power consumption, and are highly suited to battery operation.
- They can be placed 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 through 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

Symbols

● Network co-ordinator

▲ Router

■ End device

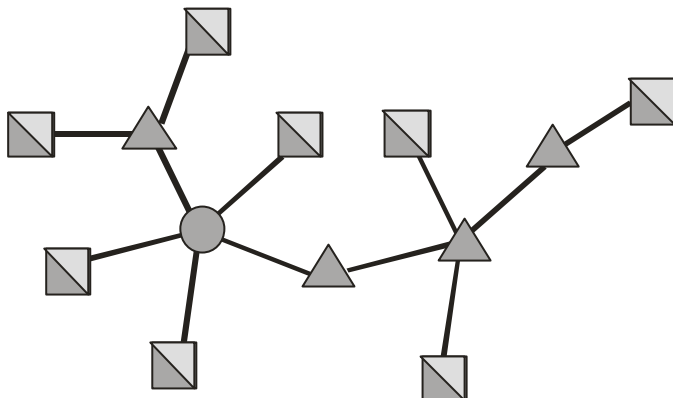
Shading

● FFD

● RFD

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

Symbols

● Network co-ordinator

▲ Router

■ End device

Shading

● FFD

● RFD

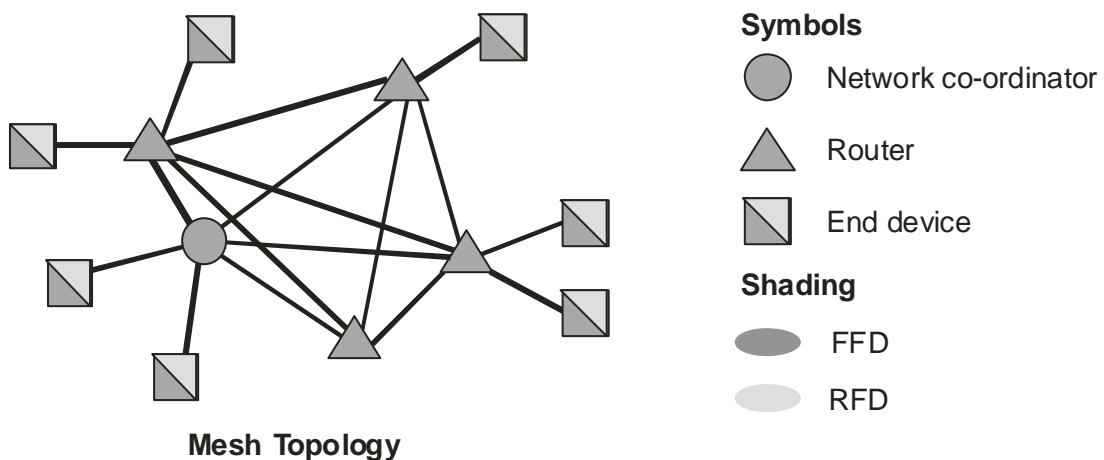
In the cluster tree topology, the co-ordinator, the parent, is linked to a number of routers and end devices, its children.

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



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.

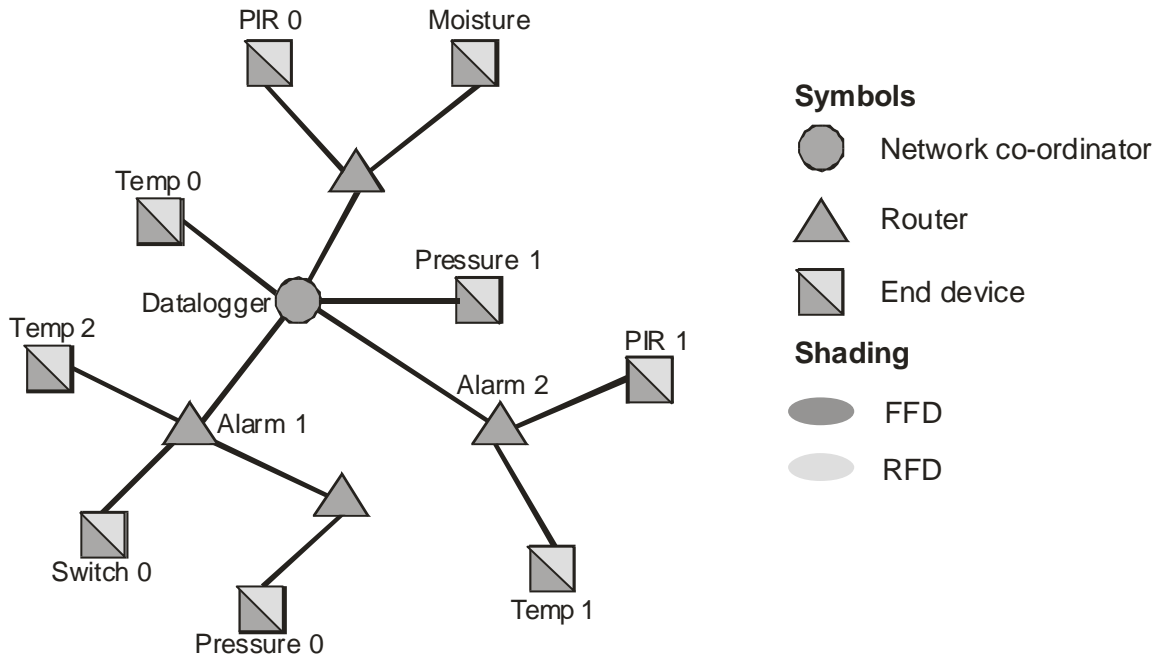
It also sets the maximum depth of the network (maximum number of hops from the co-ordinator to the most distant device.)

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, Alarm 1 and Alarm 2. 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 in sleep mode, then that router is responsible for buffering incoming messages for them.

2.5 ZigBee Addressing Scheme

The Matrix 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.

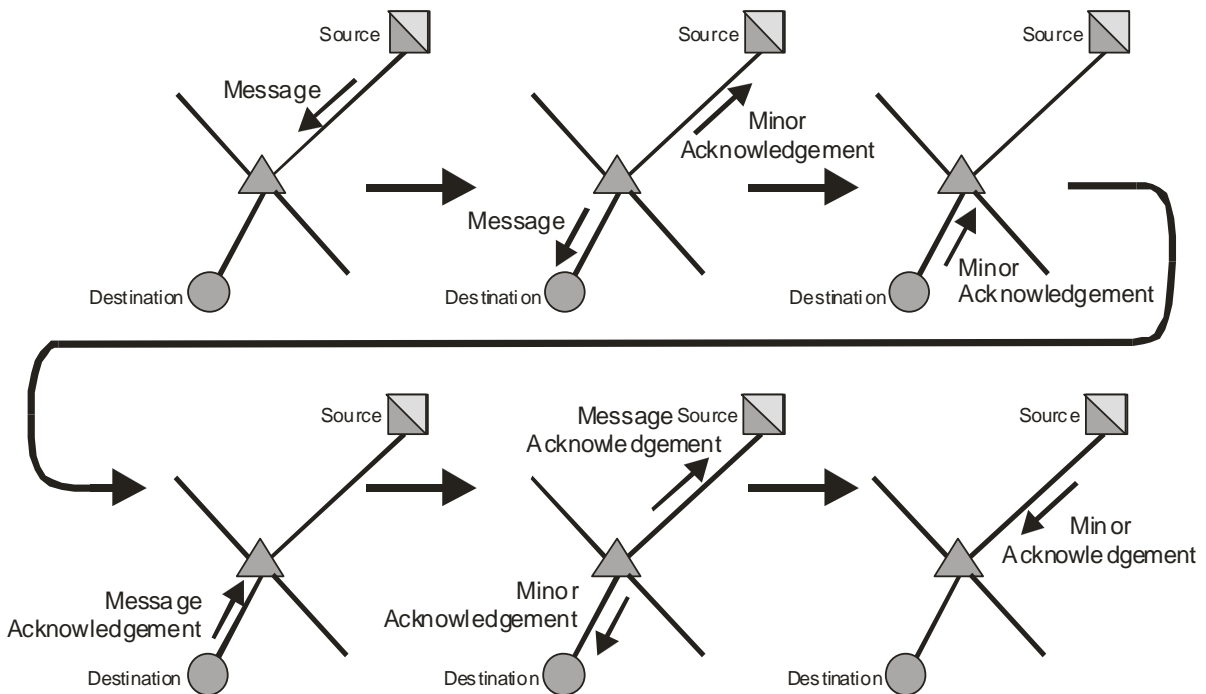
Under some circumstances, the network address may change. For example, the node may be mobile, or the parent router may become disconnected. As a result, it is not good practice to address nodes directly using their network address.

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 it has joined, it is assigned a sixteen bit network address, which it then uses to communicate with the other network devices.

The 802.15.4 specification uses two forms of communication between devices in the network, called **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 re-joining 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 a network to be formed and then, at a later date, extended by the addition of extra sensor nodes without the need 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 difficult, 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 aim 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 then maintain a routing table, which maps 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. When they need to communicate, 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, including the destination node or its parent, eventually see the broadcast. The destination node 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 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 re-join 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 node wakes up, it must check whether its parent has any stored messages.

The Version 2 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 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 one 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 wake up.

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:

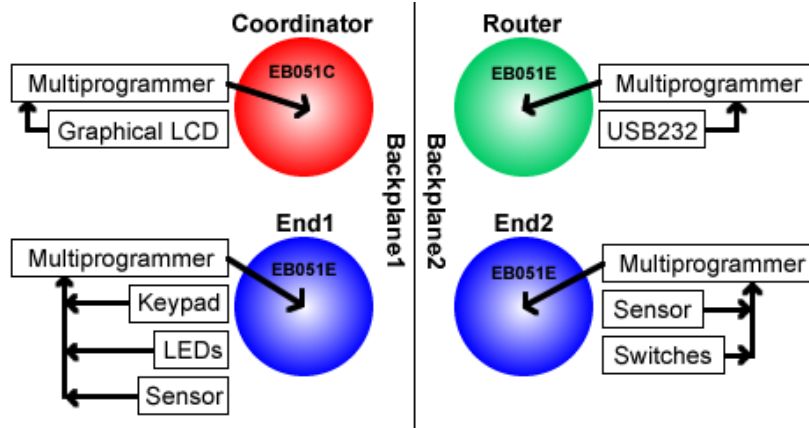
- Access Control Lists: Only pre-defined "friendly" nodes can join a network. The co-ordinator contains a table of permitted MAC addresses.
- Message Freshness Timers: Timed-out messages are rejected, preventing replay attacks on the network. A replay attack occurs when an attacker records a transmission 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 ZigBee training solution

The practical work in 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, nodes will need to be re-configured. A fifth connection point is used in conjunction with the ZENA Wireless Adapter to monitor network traffic.



The ZigBee training solution consists of two metal backplane 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 USB virtual comm port from the gateway. If two USB ports are not available then the USB devices can be interchanged as needed.

3.1 The ZigBee E-Blocks2 board



The ZigBee E-Blocks2 board connects to the input / output pins on the microcontroller board and has an XBee module that is used to set up a wireless connection between the microcontroller and the rest of 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 co-ordinator, the LED will flash at 1Hz.
- When the node is not connected to a network then the LED will remain off.

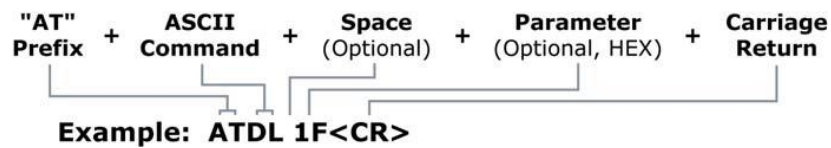
The XBee modules supplied with the ZigBee boards are configured in one of two modes, Co-ordinator or Router/End device. The Co-ordinator (BL0152C) must be used to start the network. There can be only one co-ordinator node on the network. The Router/End Device node (BL0152R) can be configured as a routing device or as an end device. There can be almost any number of these on the network.

3.1.1 AT Commands

The Version 2 XBee modules are controlled by a set of ASCII commands known 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 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. A complete list of supported commands is provided in the Instructor's Guide to the course.



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, any check for updates via the menu Help->Check for updates
- The ZENA™ Wireless Adapter requires the installation of a USB driver. See the ZENA™ Wireless Adapter User's Guide for instructions.

3.2.1 Setting up the ZigBee system

The standard ZigBee node consists of a development panel based on either the PIC BL0011 or Arduino based BL0055 programmer board, together with various combinations of downstream E-blocks2 interface boards.

The following section tables the required boards for each exercise.

3.2.2 Setting up the ZigBee nodes

The four ZigBee nodes are set up as follows:

Exercise 1

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C

Exercise 2

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167	BL0138	BL0152R	BL0167	BL0138	BL0152R

Exercise 3, 4, 5

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167		BL0152R	BL0167		BL0152R
Node 3	BL0145		BL0152R	BL0145		BL0152R
Node 4			BL0152R			BL0152R

Exercise 6

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167		BL0152R	BL0167		BL0152R
Node 3	BL0129		BL0152R	BL0129		BL0152R
Node 4	BL0167		BL0152R	BL0167		BL0152R

Exercise 7

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167		BL0152R	BL0167		BL0152R
Node 3	BL0129		BL0152R	BL0129		BL0152R
Node 4		BL0158	BL0152R		BL0158	BL0152R

Exercise 8, 9

	PIC BL0011			Arduino BL0055		
	Port A	Port B	Port C	A0-5 (C)	D0-7 (D)	D8-13 (B)
Node 1	BL0157		BL0152C	BL0157		BL0152C
Node 2	BL0167	BL0138	BL0152R	BL0167	BL0138	BL0152R
Node 3	BL0129	BL0145	BL0152R	BL0129	BL0145	BL0152R
Node 4		BL0158	BL0152R		BL0158	BL0152R

Corresponding ZigBee component connection information

Connections	
Channel	Channel 1
TX	\$PORTC.1
TX Remap Pin	\$PORTC.1
RX	\$PORTC.0
RX Remap Pin	\$PORTC.0
Baud Options	9600
Baud Rate	9600
Use Flow Control	Off
RTS	\$PORTC.3
CTS	\$PORTC.4
Sleep Pin	\$PORTC.2

PIC (BL0011) Zigbee connections

Connections	
Channel	Software
TX	\$PORTB.1
RX	\$PORTB.0
Baud Options	9600
Baud Rate	9600
Use Flow Control	Off
RTS	\$PORTB.3
CTS	\$PORTB.4
Sleep Pin	\$PORTB.2

Arduino (BL0055) Zigbee connections

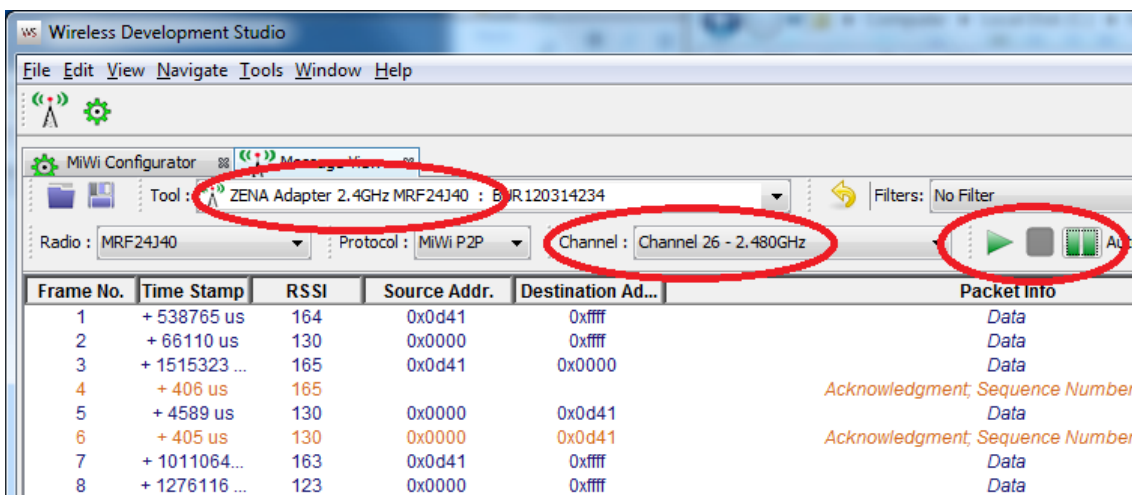
The sleep pin is used to control the pin sleep mode available to the module.

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, the Microchip ZENA™ Wireless Adapter is included in the ZigBee solution pack.

The **ZENA™ Wireless Adapter**, connected to a computer, allows the user to monitor and debug a ZigBee network when used in conjunction with the Microchip **Wireless Development Studio**. Please refer to the Microchip ZENA™ documentation for details on installing and setting up the software. Install the Wireless Development Studio, the USB drivers and attach the ZENA™ Wireless Adapter.

Run the Wireless Development Studio, select the ZENA™ Wireless Adapter and 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 information about each transmission. This makes troubleshooting much easier and is an aid in the understanding of ZigBee and other wireless networks.

3.2.4 Testing the ZigBee system

- Set up the ZigBee system and switch on power to the four nodes.
- Connect the ZENA ZigBee analyzer to the computer.
- Program the target devices by loading each program from the test folder into Flowcode and download them to the appropriate nodes
 - Node 1 – Coordinator and display node
 - Node 2 – Switch node
 - Node 3 – LED node
 - Node 4 – Sensor node

Once the four nodes have been programmed you can test Node 1 and Node 4 by moving the sensor control knob and watching for a corresponding change on the Node 1 display. Nodes 2 and 3 can be checked by pressing switch 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 a four node ZigBee network, developed 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 USB comm port 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 while allowing the addition of further 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 no hardware flow control.

Information sent and received is stored in a series of buffers. (These are part of the Matrix 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 target system configured as Node 1 (see section 3.2.2)
- a Microchip ZENA ZigBee analyser.

5.4 The Flowcode program in detail

The program will:

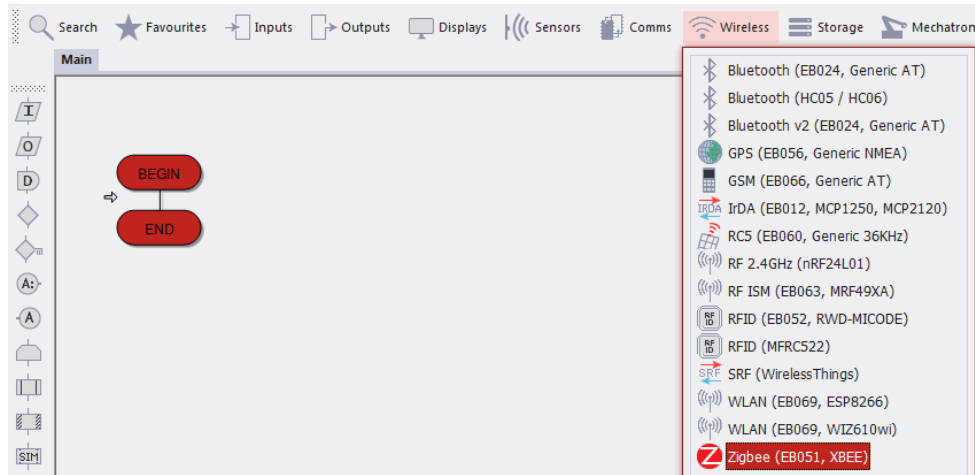
- assign properties to the ZigBee Flowcode component;
- configure the co-ordinator mode with:
 - a preset PAN ID;
 - a device name of *Coord*;
 - an infinite join time;
 - the channel set to 1A;
 - a scan duration of 3;
 - a maximum hop count of 4;
- display the current active channel on the graphical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

5.4.1 Target microcontroller

The course assumes the use of development panels using BL0011 for PIC or BL0055 for Arduino Uno. Other microcontroller devices can be used instead, but the settings, programs and instructions may need to be adapted to match the new device.

5.4.2 Flowcode ZigBee component

The Flowcode ZigBee component is added to a Flowcode program by clicking the ZigBee icon, which can be found in the Wireless section of the Component Toolbar

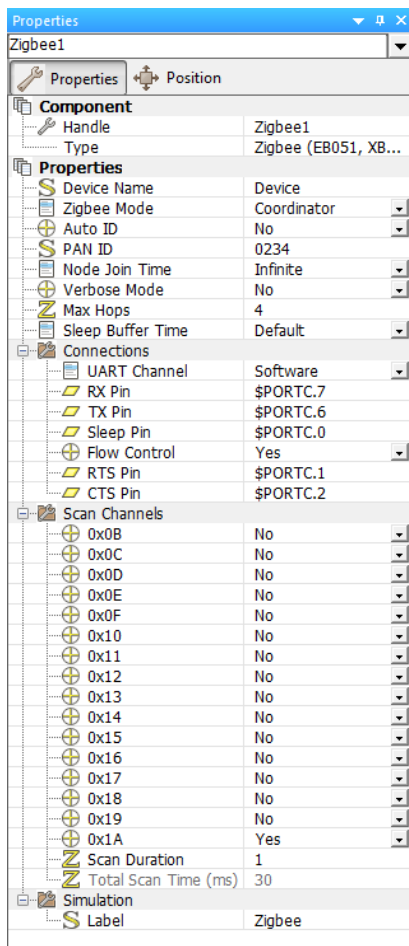


Once the ZigBee component has been added to the program, the component icon appears in the System Panel window.



5.4.3 ZigBee component settings

Select the Zigbee component by clicking on its icon in the System Panel window, or by selecting the Zigbee component from the drop-down box at the top of the Properties Panel.



Component Properties:

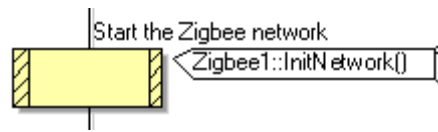
The ZigBee component properties are grouped into several categories as shown in the following diagram:

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

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

5.4.4 InitNetwork function

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



The InitNetwork macro configures the ZigBee module to the profile specified in the Component Properties, by sending appropriate AT commands.

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. Note:

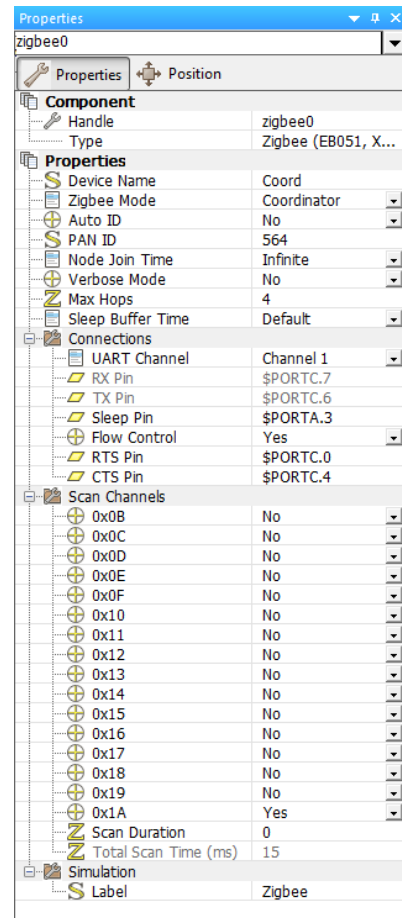
- All parameters sent with the AT commands are in hexadecimal format except for the character based address which is in ASCII character format.
- All AT commands are terminated with a carriage return, ASCII code 13.
- The Send AT command function is used to send out an AT command in the form of a string.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

ATID xxxx	Assigns the PAN ID xxxx.
ATNI xxxxxxxx	Assigns the character based node address xxxxxxxx.
ATSC xxxx	Assigns the RF channel xxxxt to be scanned.
ATSD x	Sets the RF channel scan duration to x.

The ZigBee InitNetwork and other macros handle all of the actual AT commands that control the ZigBee module.

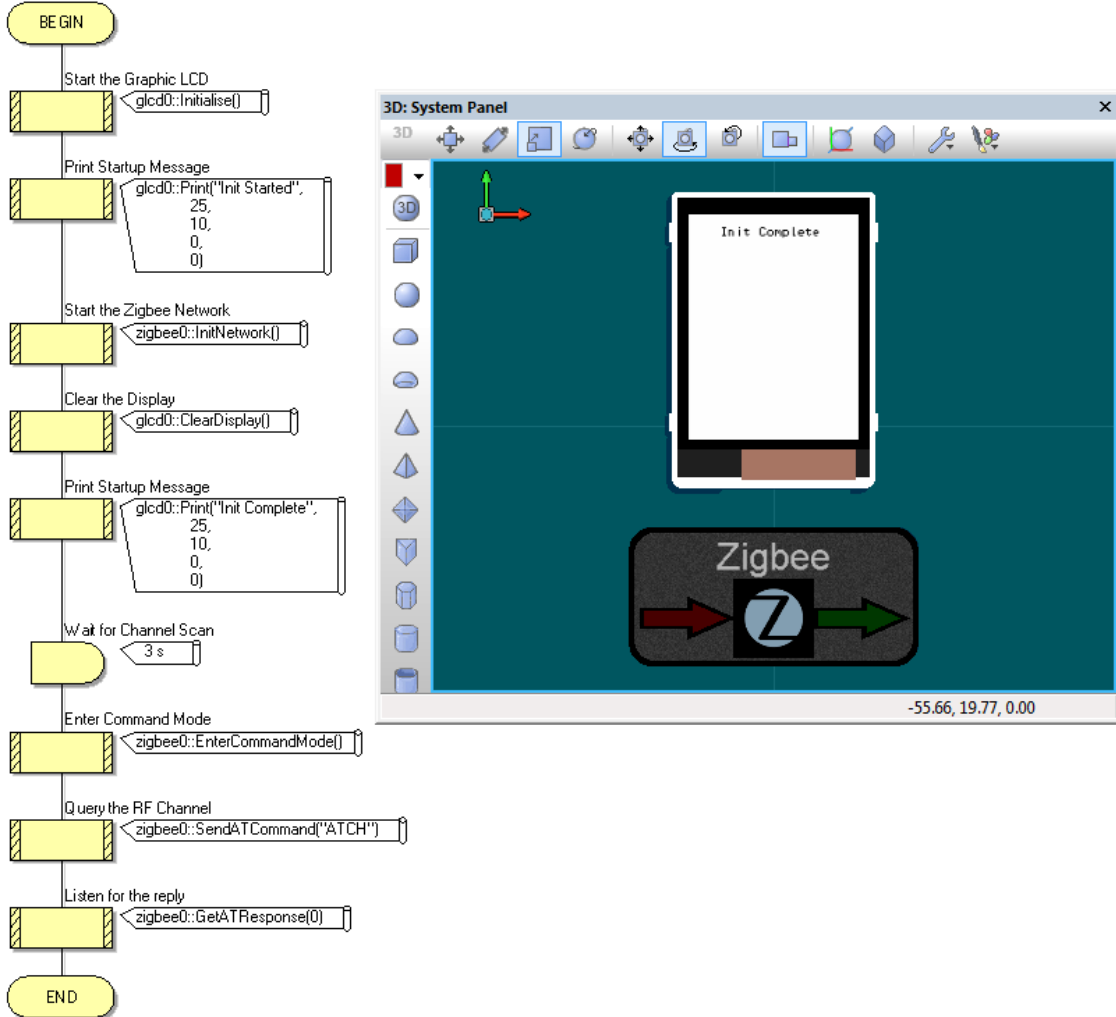
5.5 What to do

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;
 - i. Insert a Component Macro, and select the gLCD(0) component and the Init macro to initialise the Graphical LCD component;
 - ii. Give the macro the Display Name "Start the Graphic LCD";
 - (4) Insert another Component Macro, and select the gLCD(0) component and the Print macro;
 - iii. Insert the following in the Parameter box: "Init Started", 25, 10, 0, 0;
 - iv. Give the macro the Display Name "Print Startup Message";
 - (5) Insert a Component Macro, and select the Zigbee0) component and the InitNetwork macro to initialise the ZigBee co-ordinator;
 - v. Give the macro the Display Name "Start the ZigBee Network";
 - (6) Insert another Component Macro, and select the gLCD(0) component and the Clear macro to clear the Graphical LCD;
 - vi. Give the macro the Display Name "Clear the Display";
 - (7) Insert another Component Macro, and select the gLCD(0) component and the Print macro;
 - vii. Insert the following in the Parameter box: "Init Complete", 25, 10, 0, 0;
 - viii. Give the macro the Display Name "Print Startup Message";
 - (8) Add a delay icon, set to 3 seconds with the Display Name "Wait for Channel Scan";
 - (9) Insert a Component Macro, and select the Zigbee0) component and the EnterCommandMode macro;
 - ix. Give the macro the Display Name "Enter Command Mode";
 - (10) Insert a Component Macro, and select the Zigbee0) component and the SendATCommand macro;
 - x. Insert the following in the Parameter box: "ATCH"
 - xi. Give the macro the Display Name "Query the RF Channel";
 - (11) Insert a Component Macro, and select the ZigBee(0) component and the GetATResponse macro;
 - xii. Insert the number 0 as the logging parameter ;
 - xiii. Give the macro the Display Name "Listen for the Reply";
 - (12) End the program.



2. Compile the program and transfer it to the target microcontroller 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.



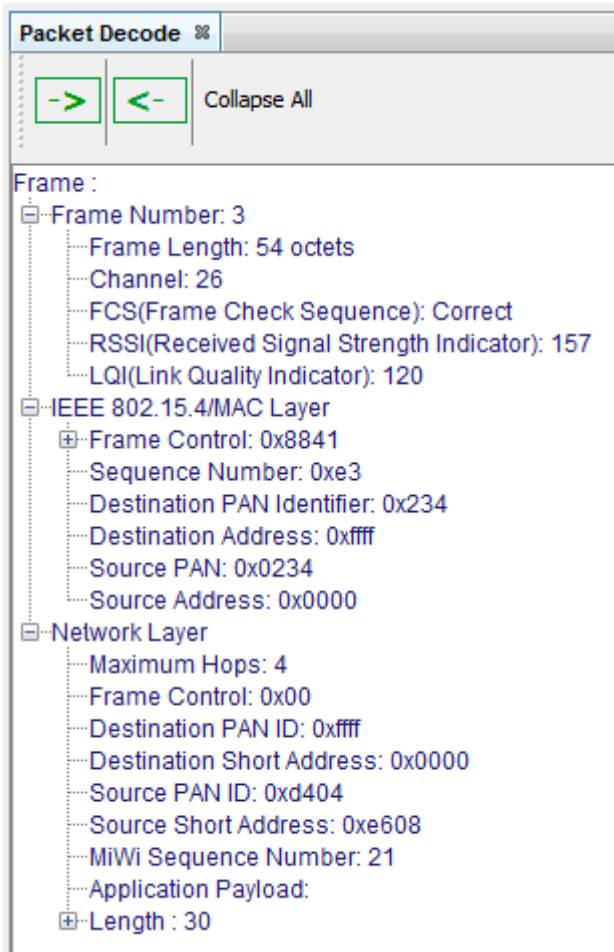
5.5.1 Analysing the network with ZENA

To monitor the network with the ZENA analyser, plug it into a spare USB port on the computer and then run the Wireless Development Studio.

Select the ZENA™ Wireless Adapter and 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.

Click on the green “Start Capture” button and the Wireless Development Studio should now capture a signal sent out regularly from the ZigBee co-ordinator node.

To analyze a packet, click on it and check the “Packet Decode” window, it will look something like this:



Notice details such as:

- The Personal Area Network Identifier: 0x0234
- Destination address: 0xFFFF means broadcast to all nodes on the PAN
- 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 to 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 target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- 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 with 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.

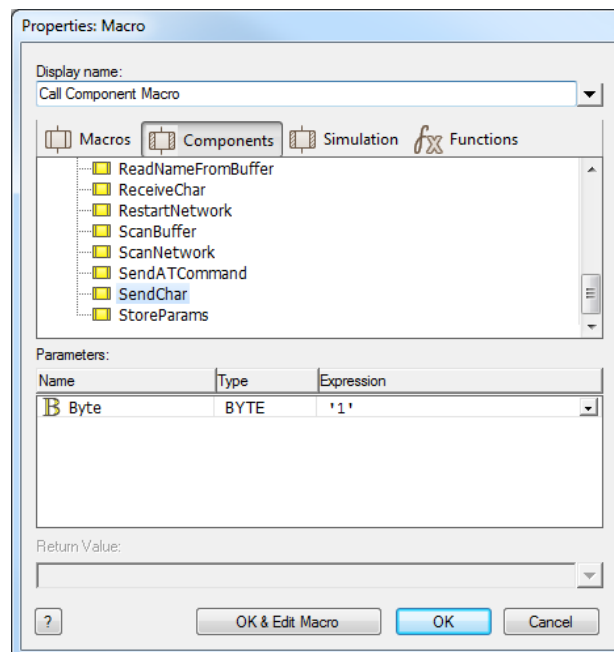
Here are some of the AT commands for the end device node that will be used in this exercise:

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 SendChar and ReceiveChar macros can be used to communicate across the network.



Both macros can handle single characters and data in variables.

If sending a character use apostrophes/single quotes around the character e.g. '1'.

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 how to specify which device or devices the node wishes to talk to.

This is done by using the Flowcode ZigBee component ConnectTo macros, which include:

- ConnectToCoordinator;
- ConnectToAddress (to connect to a specific network address);
- ConnectToName (to connect to a specific name address);
- ConnectToAll

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

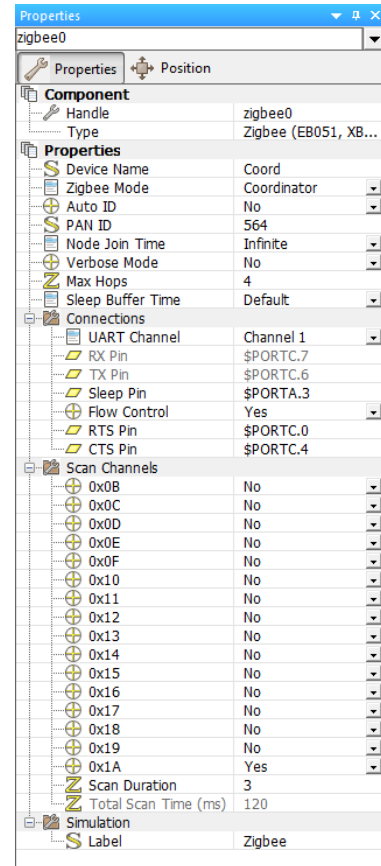
6.4.4 Monitoring the signal strength

The signal strength is displayed on the Node 1 graphical display.

6.5 What to do

For Node 1:

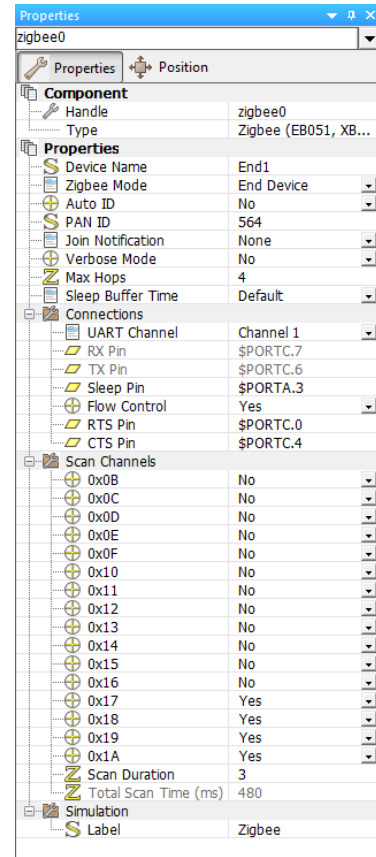
1. Write the Flowcode program using the following steps as a guide:
 - (1) Load the ZigBee component into a new Flowcode flowchart;
 - (2) Configure the ZigBee component properties as shown in the next diagram;
 - (3) Load the Graphical LCD component into the flowchart;
 - (4) Insert a Component Macro, 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 InitNetwork 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 SendChar 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 ReceiveChar 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) Add another loop called 'Main', tested at the start of the loop, with a Loop While value of 1 (i.e. forever,);
 - (30) Insert a Component Macro, and select the Zigbee0) component and the ReceiveChar macro with a nTimeout(BYTE) parameter of 20;
 - (31) Use the variable "Incoming" as the Return Value;
 - (32) Give the macro the Display Name "Receive Keypad data";
 - (33) Insert a Decision box, called "If Data Received?", with the condition 'Incoming <255' (meaning that data has arrived,);
 - (34) On its 'Yes' loop add a Calculation icon, called 'Convert Keypad number to String', to carry out the function 'TextString = "" + Incoming' ;
 - (35) Then insert a Component Macro, and select the gLCD(0) component and the Print macro;
 - (36) Insert the following in the Parameter box: ' TextString, 60, 60, 2, 0';
 - (37) Give the macro the Display Name "Print Keypad Data to LCD";
 - (38) Its 'No' arm continues to the next icon;
 - (39) Insert a Component Macro, and select the Zigbee0) component and the GetSignalLevel macro with a nTimeout(BYTE) parameter of 20;
 - (40) Create a new variable called sig_lev and use it as the Return Value;
 - (41) Give the macro the Display Name "Read the last Signal Level";
 - (42) Add a Calculation icon, called 'Convert Signal Level to String', to carry out the function 'TextString = ToString\$(sig_lev)';
 - (43) Insert a Component Macro, and select the gLCD(0) component and the Print macro;
 - (44) Insert the following in the Parameter box: "Signal Level", 2, 12, 0, 0;
 - (45) Give the macro the Display Name "Print Signal Level";
 - (46) Insert a Component Macro, and select the gLCD(0) component and the Print macro;
 - (47) Insert the following in the Parameter box: " ", 100, 12, 0, 0;
 - (48) Give the macro the Display Name "Overwrite the Previous Value with Spaces";
 - (49) Insert a Component Macro, and select the gLCD(0) component and the Print macro;
 - (50) Insert the following in the Parameter box: TextString, 100, 12, 0, 0;
 - (51) Give the macro the Display Name "Print Signal Level Data";
 - (52) The next icon is the end of the Main loop;
 - (53) That is followed by the End icon.
2. Compile the program and transfer it to the target microcontroller 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 InitNetwork 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 ConnectToCoordinator 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 ReceiveChar 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 SendChar 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 GetAscii 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 SendChar 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 target microcontroller 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 from the end device to the co-ordinator.
- That is followed by an automatic minor acknowledgement, and then by a 'message received' acknowledgement from the co-ordinator to the end node.
- This again generates an automatic minor acknowledgement to confirm reception of the 'message received' acknowledgement.

Frame No.	Time Stamp	RSSI	Source Addr.	Destination Addr.	Packet Info
1	+ 476048 us	187	0x0d41	0x0000	Data
2	+ 403 us	187			Acknowledgment; Sequence Number - 0x4d
3	+ 3283 us	139	0x0000	0x0d41	Data
4	+ 498 us	139	0x0000	0x0d41	Acknowledgment; Sequence Number - 0xb4

6.6 Further work

- How would you scan for more than 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 idea is to add two more nodes onto the ZigBee network 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, the co-ordinator node (node 1) is used 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 target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- a target system configured as Node 3 (see section 3.2.2)
- a target system configured as Node 4 (see section 3.2.2)
- 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 the co-ordinator in an attempt to join the network;
- display the current status of the process on the LEDs as it joins the network ;
- transmit confirmation to the co-ordinator when it has joined the network;
- wait to receive a transmission from node 3, and use that data 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;
- then 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 joined 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 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 InitNetwork 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 for 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

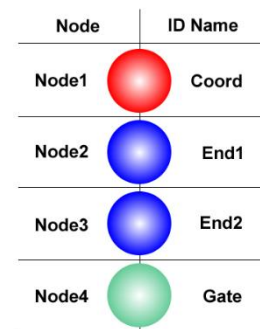
Each node that joins the network by connecting directly to another node is referred to as a child of that node. Only co-ordinator and router nodes are capable of being parent devices. A single parent node can have up to eight associated child nodes. For more than eight child nodes, one of the children 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. Here is an overview of the programs:

For node 1:

- Initialise the LCD display;
- Initialise the ZigBee component;
- Command the co-ordinator to connect to all devices;
- Clear the LCD;
- 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 ReceiveChar 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 ReceiveChar 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 ConnectToName 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 ReceiveChar 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 processor board 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 processor board 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 co-ordinator 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 additional minor acknowledgments being sent out by all the intermediate nodes.

Frame No.	Source Addr.	Destination Addr.	
1	0x0000	0xffff	Coordinator (0x000) Sends '1'
2	0x6741	0xffff	Node 2 Sends '2'
3	0x0d41	0xffff	Node 3 Sends '3'
4	0x6cea	0xffff	Node 4 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

The 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 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 sleep mode operation of the XBee device will be explored.

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

8.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- a target system configured as Node 3 (see section 3.2.2)
- a target system configured as Node 4 (see section 3.2.2)
- 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 joined 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 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 NodeConfigureSleep 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 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, 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 practice, this is achieved by assigning the same sleep timeout to all nodes on the network.

8.4.3 Polling for buffered data

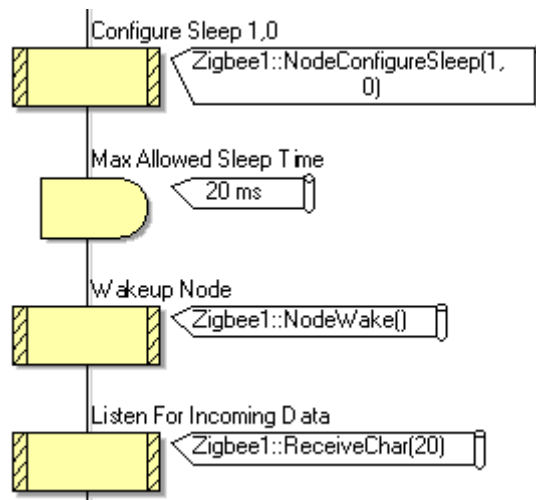
Data is collected from the parent node by issuing the 'force poll' AT command.

The Flowcode NodeWake 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 NodeWake 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 ReceiveChar 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, while still allowing the keypad and LED controller functionality. Whenever the keypad on node 2 is pressed, 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 processor board 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 ReceiveChar macro to look for new values of 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) NodeConfigureSleep macro, with Mode(BYTE), Timeout(INT) parameters of 2, 0;
 - Add a 5 second delay;
 - Insert a Component Macro that calls the ZigBee(0) NodeWake macro;
 - Loop back to the infinite loop
2. Compile the program and transfer it to node2.
3. Press reset on the node 2 processor board. 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 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 processor board.

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 processor board.

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 maintaining the node's functionality.
- 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 either 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 target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- a target system configured as Node 3 (see section 3.2.2)
- a target system configured as Node 4 (see section 3.2.2)
- 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”;
- display a list of nodes connected to the network and highlight those whose name begins 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 joined 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

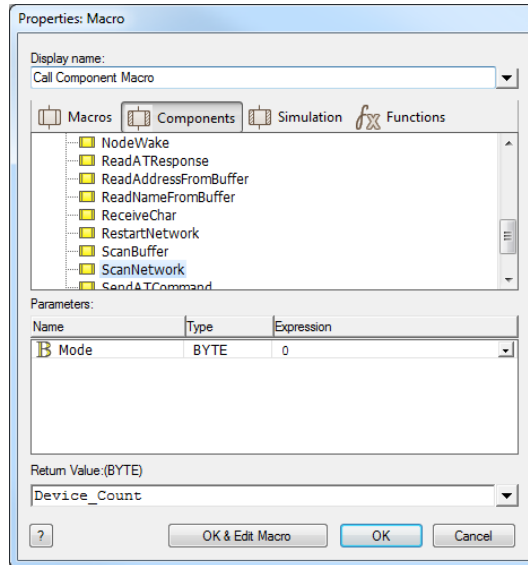
An identifier address is a character string of up to 8 characters. It could be just an arbitrary name or it could indicate the functionality of the node

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 can 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.

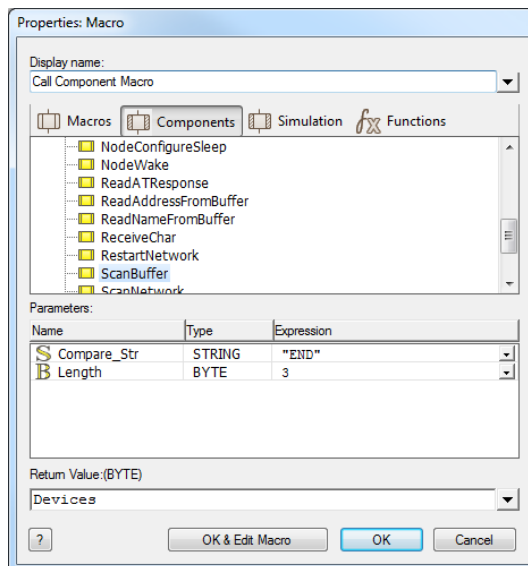
9.4.3 Polling for nodes

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

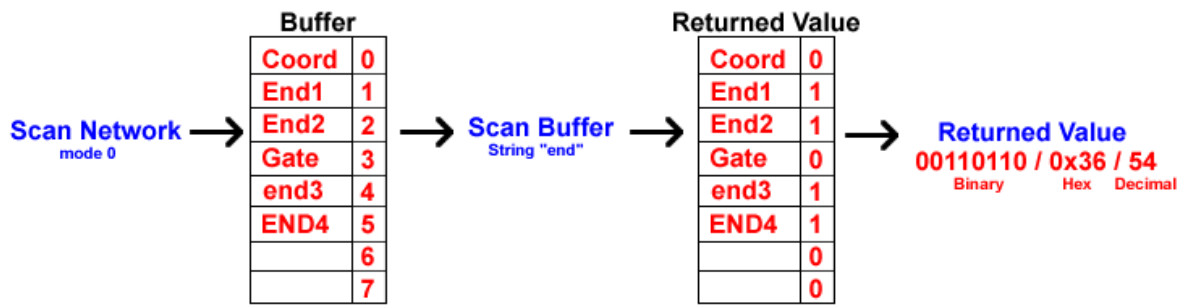


The ScanNetwork macro has a mode parameter that controls whether a detected device will be identified by its (numerical) address (mode1) or by its identifier address (mode 0). In the diagram above, the node will scan the network and store the identifier addresses of detected devices in a variable named 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 ScanBuffer 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 ScanBuffer 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) Start an infinite loop, called ‘Main Program Loop’;
 - (6) Clear the LCD;
 - (7) Print the message “Scanning for nodes”;
 - (8) Add a ScanNetwork Component macro, called ‘Scan for nodes’ using mode 0, and using the variable ‘Device_Count’ as the Return Value;
 - (9) Print the message “Scan complete”;
 - (10) Print the message “Nodes detected” in position (x2, y15);
 - (11) Add a gLCD(0) PrintNumber Component macro, called ‘Printnumber nodes detected’ in position (x110, y15), using Device_Count as the number;
 - (12) Add a ScanBuffer 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;
 - (13) Print the message “End Devices” in position (x2, y28);
 - (14) Add a gLCD(0) PrintNumber Component macro, called ‘Printnumber end devices’ in position (x110, y28), using ‘Devices’ as the number;
 - (15) Create a variable called ‘idx’, and set its value to zero using a Calculation icon called ‘Initialise index’ ;
 - (16) Create a loop called ‘Scan through Device Table’ that continues while the value of ‘idx’ is less than 8;
 - (17) Create a variable called ‘Flag’ and add a calculation icon called ‘Mask one device at a time’ to perform the calculation ‘Flag = Devices & 0x01’;
 - (18) Add a Decision icon to test if the variable ‘Flag’ is true;
 - (19) In the ‘Yes’ loop, add a gLCD(0) SetForegroundColor macro, with parameters 7,0,0 for red, green and blue;
 - (20) In the ‘No’ loop, add a gLCD(0) SetForegroundColor macro, with parameters 0,0,0 for red, green and blue;

- (21) Create a variable called 'idx2', and set its value to zero using a Calculation icon called 'Initialise index 2';
- (22) Create a loop called 'Scan through Device Name' that continues while the value of 'idx2' is less than 8;
- (23) Add a ReadNameFromBuffer Component macro, called 'Read name from buffer' using 'idx' and 'idx2' as device and index parameters respectively, and TextString[idx2] as the Return Value;
- (24) Add a Calculation icon called 'Increment index 2' to increment 'idx2';
- (25) End the loop 'Scan through Device Name';
- (26) 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';
- (27) Add a gLCD(0) PrintNumber Component macro, called 'Print device index' in position (x2, y= Ycoord), using 'idx' as the number;
- (28) Add a gLCD(0) PrintNumber Component macro, called 'Print device flag' in position (x10, y= Ycoord), using 'Flag' as the number;
- (29) Print the message "Print Device ID Name" in position (x20, y=Ycoord), using TextString as the string parameter;
- (30) Add a Calculation icon called 'Move to next Device' to increment the variable 'Devices';
- (31) Add a Calculation icon called 'Increment index' to increment 'idx';
- (32) End the loop 'Scan through Device Table';
- (33) Add a ten second delay;
- (34) End the loop Main Program Loop
- (35) End the program.

2. Compile the program and transfer it to node1.
3. Press reset on the node 1 processor board 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 processor board.

For node 3:

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 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 processor board.

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 processor board.

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 ConnectFromLibrary 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 the receiving node knows where incoming messages originated and where they are destined for.

A typical situation where problems can occur is 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 to 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 target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- a target system configured as Node 3 (see section 3.2.2)
- a target system configured as Node 4 (see section 3.2.2)
- 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;
- announced itself on the network by sending its node number;
- 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 sensor 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 ZigBee course CD-ROM.

- Download 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 to generate and transmit 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 node 3, use the second data byte to send the data from the LDR on the sensor board. The LDR potentiometer 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 than 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 UART to USB virtual comm port 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 the 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 target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- a target system configured as Node 3 (see section 3.2.2)
- a target system configured as Node 4 (see section 3.2.2)
- 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;
- 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;
- announced itself on the network by sending its node number;
- 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 sensor reading periodically and convert it 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 is received 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.

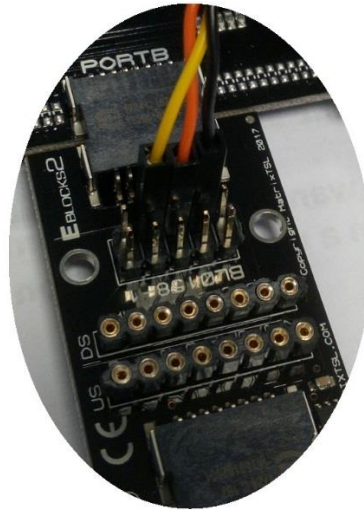
ASCII data has a different format, encoding each digit or character as a byte of data. For example, the letter 'A' is represented by the decimal number 65, whereas 'a' is represented by the number 97. A full list of ASCII characters can be found at <http://www.asciitable.com/>

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 second relates to a character or control command.

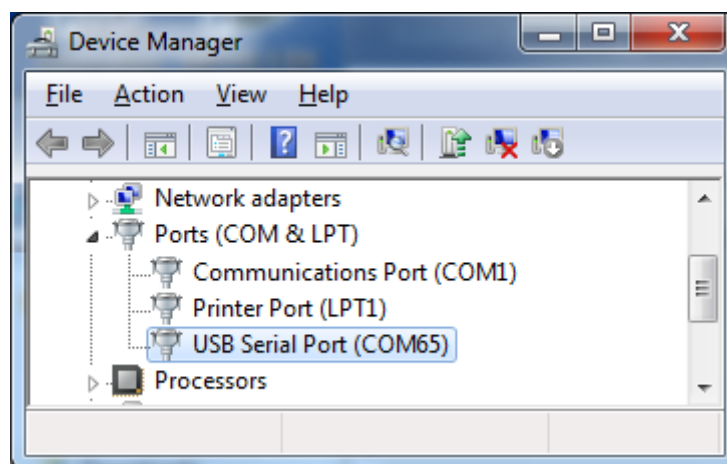
HyperTerminal accepts bytes in ASCII form so there needs to be a way of converting to this from pure number format. The Flowcode function named "BYTE_2_NUMBER" performs this transformation, and can be imported into a Flowcode program to allow this conversion to take place.

11.4.4 Connecting the USB Serial port

The Connect the USB FTDI Serial cable to the processor board of Node 4 via the BL0158 interface board as shown below.

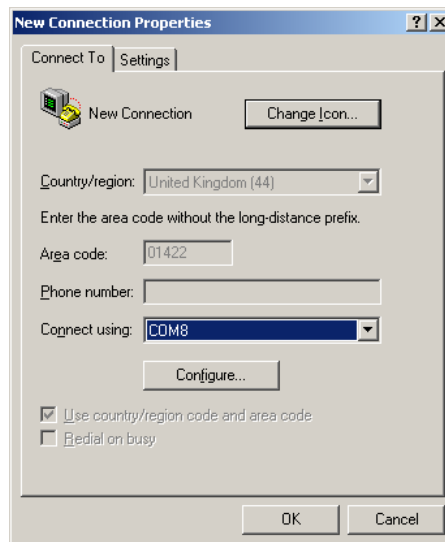


When the USB cable is plugged into the PC the FTDI drivers will be installed. The associated COM port can be found by looking in the Device Manager under the heading Ports (COM & LPT).

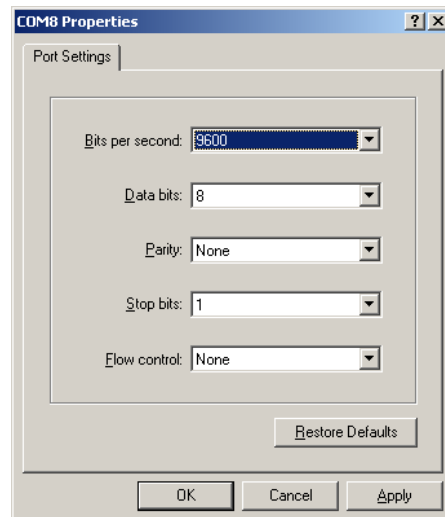


That done, the terminal emulation software can be configured to use this hardware, for example Windows HyperTerminal by following the steps outlined below:

1. Open the HyperTerminal application.
2. Give the new connection a name, such as 'ZigBee'.
3. Select the COM port associated with the USB Serial port 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 a serial communications port, which uses 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 has only 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 USB serial cable.

The data is then in a form allowing 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.

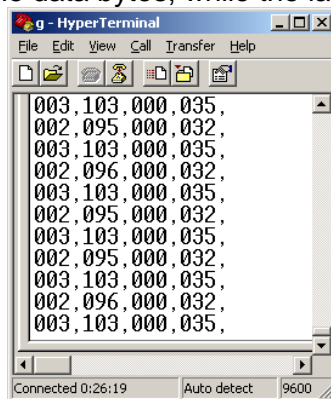
The program for node 4 is modified by the addition of:

- 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 “GATE”.
- Then they begin sampling the sensor voltages and converting them to digital signals.
- These are then transmitted to node 4, the gateway node.
- This node then forwards each packet to node 1 and also sends the data to the computer via the bit banged USB232 connection. There, 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 target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- a target system configured as Node 3 (see section 3.2.2)
- a target system configured as Node 4 (see section 3.2.2)
- 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 board on Node 3 acts as temperature sensor.

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 also to the computer.

Once the network has been established, nodes 2 and 3 begin searching for a device with the name "GATE", and then node 3 samples the output of the on-board potentiometer. 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 banded 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 node 3 has an analogue sensor E-Block connected to it equipped with a potentiometer. This will be used to simulate temperature changes at the end device node. 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 controller, 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 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 banded USB port.

The diagram shows a typical output from the system. The data is arranged in five columns, conforming to 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 next 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 ZigBee course CD-ROM.

- Download 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 "GATE".
- Once these have found the correct address for the gateway, node 3 begins sampling the potentiometer voltage, connected to analogue channel AN1.
- After each sample, a packet will be generated and transmitted to node 4, the gateway node.
- Once the gateway has received a complete transmission, it forwards the packet to node 1 and to the computer via the USB serial 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, to prevent hackers from gaining access to the network, while still allowing future expansion by the addition of legitimate nodes.

13.3 Requirements

This exercise requires:

- a copy of Flowcode running on the PC
- a target system configured as Node 1 (see section 3.2.2)
- a target system configured as Node 2 (see section 3.2.2)
- a target system configured as Node 3 (see section 3.2.2)
- a target system configured as Node 4 (see section 3.2.2)
- 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.
- Node 3 acts as a temperature sensor.
- Node 3 uses the Switch E-Block as a PIR sensor.
- 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 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.

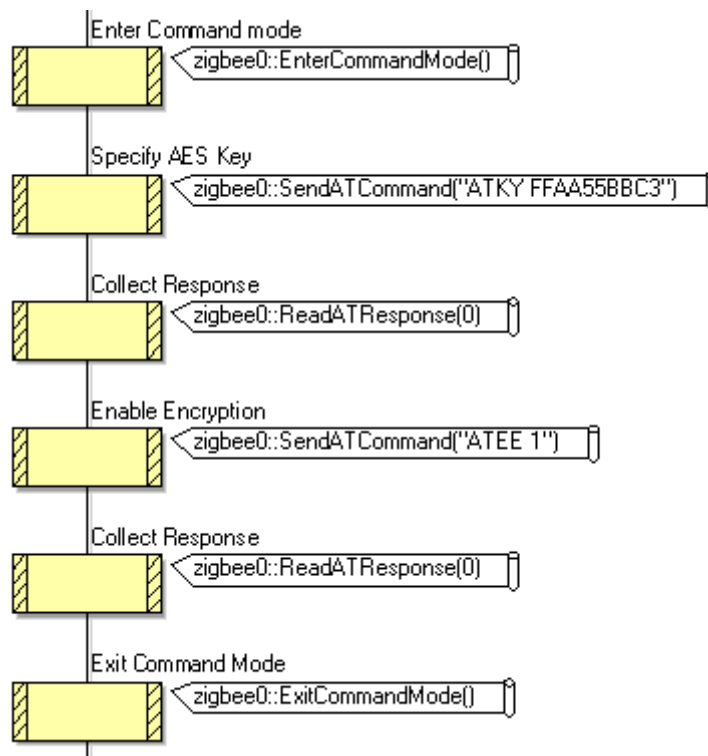
ATEE	Encryption Control (0 = off, 1 = on)
ATKY	AES Encryption Key from 0 to FFFFFFFFFFFFFFFF

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 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!