

Telegesis™	 SILICON LABS	TG-ETRXn-UG-0512
ETRX2 and ETRX357		User Guide 2.00


Telegesis™ is a trademark of Silicon Laboratories Inc.








ETRX2 and ETRX357 WIRELESS MESH NETWORKING MODULES

R3xx Firmware User Guide






Table of Contents

1	INTRODUCTION.....	6
1.1	Radio Systems and ZigBee	7
1.2	OSI 7-layer Model	7
1.3	Radio Basics	8
1.4	ZigBee Radio Details	8
1.5	Propagation, Antennae	9
1.5.1	Maximum Distance between two Devices	9
1.6	Networking.....	10
2	ZIGBEE TECHNOLOGY	11
2.1	The ZigBee standard [1]	11
2.2	Network Topologies	12
2.3	Long & short addresses	14
2.4	Endpoints.....	14
2.5	Binding.....	14
2.5.1	End Device Binding.....	15
2.6	Device Types	17
2.7	Power modes	19
2.8	Route discovery	19
3	PUBLIC & PRIVATE PROFILES.....	21
3.1	Profiles and message clusters	21
3.2	The Telegesis AT command set	22
4	NETWORK MANAGEMENT AND MAINTENANCE	22
4.1	Overview	22
4.2	Channel mask.....	23
4.3	Network size limits, max number of children.....	24
4.4	Maintaining a Network	24
4.5	Network commissioning	25
5	THE AT COMMAND SET	25
5.1	Introduction	25
5.2	AT commands for module information and configuration	26
5.2.1	ATI - Display Product Identification Information.....	26
5.2.2	ATZ - Software Reset	26
5.2.3	AT+REZ - Soft Reset of Remote Device 	26
5.2.4	AT+BLOD – Enter the Bootloader Menu	26
5.2.5	ATS and ATREMS – Read and Write S-Registers	26
5.3	AT commands for configuring a network	26
5.3.1	AT+ESCAN – Scan the energy of all channels.....	26
5.3.2	AT+PANSCAN – Scan for all available channels	26
5.3.3	AT+EN – Establish Network	26
5.3.4	AT+JN – Join Network	27
5.3.5	AT+JPAN:cc,pppp - Join specific PAN	27
5.3.6	AT+DASSL – Disassociate Local Node	27
5.3.7	AT+DASSR – Disassociate remote device from PAN	27
5.3.8	AT+N – Display network parameters	27
5.3.9	AT+SN – Scan network for other nodes	27
5.3.10	AT+ANNCE – Indicate presence in the network.....	28

5.3.11	AT+NTABLE – Display Neighbour Table	28
5.3.12	AT+CCHANGE – Change the network's channel	28
5.4	AT commands for obtaining device information	28
5.4.1	Introduction	28
5.4.2	AT+IDREQ – Request network address	28
5.4.3	AT+EUIREQ – Request EUI64	28
5.4.4	AT+NODEDESC – Request Node Descriptor	28
5.4.5	AT+POWERDESC – Power Descriptor	29
5.4.6	AT+ACTEPDESC – Request Active Endpoint list	29
5.4.7	AT+SIMPLEDESC – Request Simple Descriptor	29
5.4.8	AT+MATCHREQ – Find nodes with specified clusters	29
5.5	AT commands for data transmission	29
5.5.1	Overview	29
5.5.2	Broadcasts – AT+BCAST	30
5.5.3	Raw data – AT+RDATA	31
5.5.4	Unicasts – AT+UCAST	31
5.5.5	Unicasts – AT+SENDUCAST 	31
5.5.6	S-Casts – AT+SCAST	31
5.5.7	Multicasts – AT+MCAST	32
5.5.8	The multicast command – AT+SENDMCAST 	33
5.5.9	Display Address Table – AT+ATABLE	33
5.5.10	Channels – AT+DMODE	34
5.5.11	Sending messages between PANs – AT+INTERPAN 	35
5.5.12	Messages to and from other endpoints	35
5.6	AT commands for binding	36
5.6.1	Introduction	36
5.6.2	AT+LBTABLE – Print binding table	36
5.6.3	AT+BSET – Write to binding table	36
5.6.4	AT+BCLR – Clear binding table entry	36
5.6.5	AT+BTABLE – Print remote binding table	36
5.6.6	AT+BIND – Write to remote binding table	36
5.6.7	AT+UNBIND – Clear remote binding table entry	36
5.6.8	AT+EDBIND – Send End Device Bind request 	36
5.7	Time-related commands	37
5.7.1	Introduction	37
5.7.2	AT+SETTIME – Set local clock 	37
5.7.3	AT+GETTIME – Read local clock 	37
5.7.4	AT+SYNCTIME – Sync with time server 	37
6	SECURITY	37
7	S-REGISTERS	40
7.1	Overview	40
7.2	ATS & ATREMS – Read and write S-registers	40
7.2.1	ATREMS – addressing modes	40
7.3	Radio Setup (S00-S03)	41
7.4	Module Setup (S04-S0A)	41
7.5	Security keys (S08-S09)	42
7.6	Module Setup (S0A-S11)	42
7.7	I/O-related registers (S12-S22)	43
7.7.1	Serial port	43
7.7.2	Input terminations	43
7.7.3	Alternate functions	44
7.7.4	I/O pin control – ETRX2	44

7.7.5	I/O pin control – ETRX357	46
7.7.6	PWM control	48
7.7.7	ADCs.....	48
7.8	S-Registers Defining the Functionality of the Module	49
7.8.1	Introduction	49
7.8.2	Interrupts (S23-S28)	49
7.8.3	Timers (S29-S38).....	50
7.8.4	Power Management (S39-S3A)	50
7.8.5	Functionality text (S3B-S3C).....	51
7.8.6	Supply voltage (S3D)	51
7.9	Advanced settings (S3E-S35).....	51
7.9.1	Predefined multicast table entries (S3E, S3F)	51
7.9.2	Message endpoint settings (S40, S41)	51
7.9.3	Message cluster ID settings (S42, S43).....	51
7.9.4	Message profile ID (S44, S45).....	51
7.9.5	Counter register (S46)	51
7.9.6	Power descriptor (S47)	51
7.9.7	Endpoint 2 Profile ID, Device ID and Device Version (S48-S4A)	52
7.9.8	Endpoint 2 input and output cluster lists (S4B, S4C)	52
7.9.9	Mobile End Device poll timeout (S4D)	52
7.9.10	End Device poll timeout (S4E)	52
7.9.11	MAC timeout (S4F)	52
8	BUILT-IN FUNCTIONS.....	52
8.1	Introduction	52
8.2	Triggering a function	53
8.3	Example functions.....	55
8.3.1	Introduction	55
8.3.2	0010 – If I am an end device Poll Parent for data.....	55
8.3.3	0014 – Check for other devices on the network.....	55
8.3.4	0015 – Scan and join a network.....	55
8.3.5	0017 – Allow joining via the local node for 60 Seconds.....	55
8.3.6	0018 – Copy local Inputs to Remote outputs	55
8.3.7	001E – Disassociate from the PAN if no coordinator or sink has been heard from	56
8.3.8	003x – ETRX2: Toggle I/Ox. ETRX357: Toggle PA0-7 (x=0-7) or PB0-7 (x=8-F)	56
8.3.9	004x – Flash I/Ox (pull low) for 250ms	56
8.3.10	005x, 006x – set an output high or low	56
8.3.11	0108, 0109 – send a text message to the sink	56
8.3.12	0110 – Send ADC readings	56
8.3.13	0111 – Send ADC readings	57
8.3.14	0112, 0113 – Location tracking.....	57
8.3.15	0114, 0115 – Location tracking.....	58
8.3.16	0116, 0117 – Location tracking.....	58
8.3.17	0118, 0119 – Location tracking.....	59
8.3.18	0130, 0131 – Send ADC readings	59
8.3.19	02xx – broadcast sink address	59
8.3.20	0400 – show network status on pin PA7	59
8.3.21	0401 – show network status on pin PB7	60
8.3.22	0402 – show network status on pin PA6.....	60
8.3.23	2000 – event counter	60
8.3.24	2001 – character counter	60
8.3.25	2100, 2101 – create your own function.....	61
8.3.26	24xx, 25xx, 26xx – control other timers	61

8.3.27	3xxx – set output states	61
8.3.28	4xxx – set I/O pin directions	61
8.3.29	53xx – Toggle output xx (R309) 	61
8.3.30	54xx – Flash output xx (R309) 	61
8.3.31	55xx, 56xx – Set output xx low or high (R309) 	61
9	PWM OUTPUT	62
9.1	Tone Generation at Pin I/O3 or PB7	62
9.2	PWM register settings	62
10	GETTING STARTED WITH TELEGESIS TERMINAL	63
11	REVIVING AN UNRESPONSIVE MODULE	70
12	APPLICATION EXAMPLES	70
12.1	A simple timer application	70
12.2	More complex timer functions	70
12.3	Simple Temperature Sensor	71
12.4	External interrupts and complex commands	71
12.5	Switch Application	72
12.6	Serial Port Replacement	73
12.7	Using a Host Microcontroller	73
12.8	Custom Functionality	73
12.9	Developing Your Own Firmware	74
13	HOW TO'S	74
13.1	Check firmware version	74
13.2	Check network status	74
13.3	Start a new network	74
13.4	Join a network	74
13.5	Change device from FFD to SED	74
13.6	Get received signal strength	74
13.7	Correct a wrong baud rate setting	74
13.8	Set power mode	74
13.9	Assign a different network address	74
13.10	Assign a different EUI64	75
13.11	Make a secure network	75
13.12	Use another ZigBee profile	75
13.13	Use the ZigBee Smart Energy profile	75
14	GLOSSARY	75
15	MODULE PINOUTS	76
16	REFERENCES	77
17	APPENDIX - STARTING A NETWORK WITH TELEGESIS TERMINAL	78

1 Introduction

Wireless connectivity today has become a vital component of embedded system design as more and more applications can benefit from the features offered by the latest wireless technologies.

With the evolution of these wireless technologies the communication protocols are becoming increasingly sophisticated with most designs pushing towards higher data rates and greater point-to-point ranges.

In contrast to this development, applications requiring only a low data rate, but nevertheless needing to rely on a proven industry standard protocol are actually outnumbering the classic high data rate wireless networks worldwide. These are simple applications like lighting control, HVAC control, fire/smoke/burglar alarms, remote doorbells, humidity monitors, energy/water/gas usage monitors, and countless other applications.

To fill this void the ZigBee® Alliance has released the ZigBee specification [1] defining an open standard for low data rate wireless connectivity based on the IEEE 802.15.4 specification [2]. In comparison to other wireless standards ZigBee offers an increased reliability due to its meshing capability in conjunction with a very low power consumption and low cost.

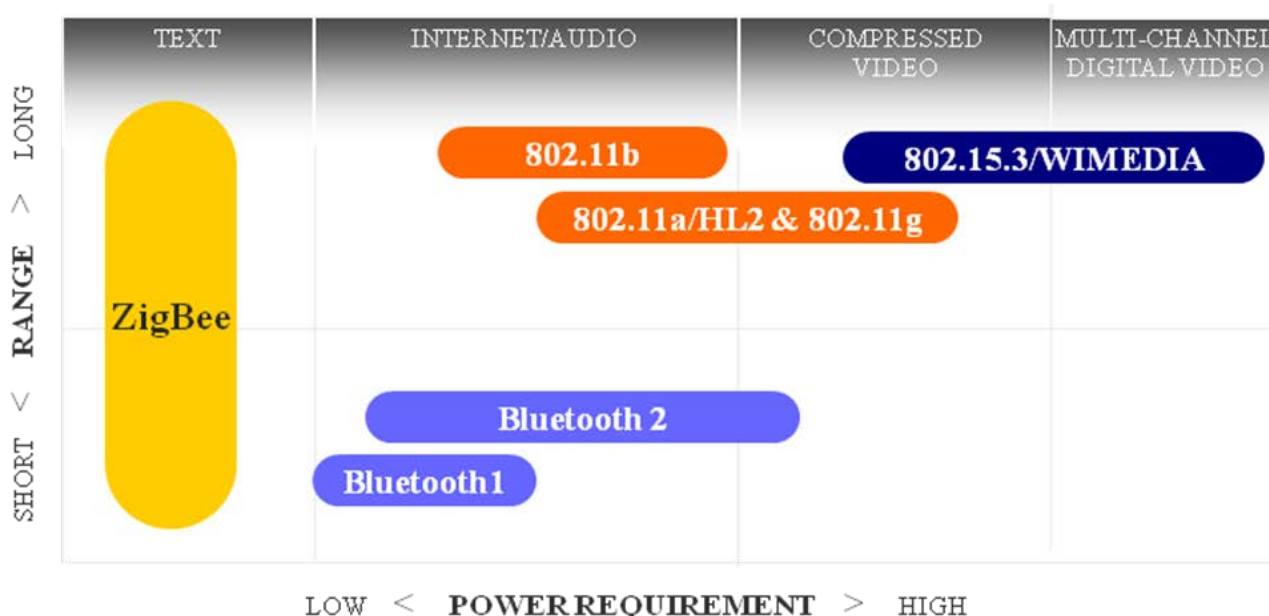



Figure 1. ZigBee Applications

When including wireless connectivity in a classical embedded system design, the development teams often don't have RF design expertise in house. Because of this the cost and time effort required for a custom RF solution including compliance testing and certification in many cases outweighs the cost of using a modular RF design even for high volume products.

Telegesis is now offering Ember based modular ZigBee transmitters since 2004, thus being one of the world's first companies offering an easy way of integrating ZigBee technology into any product. In addition to providing easy to integrate and compliance tested hardware Telegesis is also shipping all modules with the Telegesis AT command set which eliminates the need for custom firmware development for a lot of applications.

Chapters 1 to 4 give a general introduction into ZigBee and wireless mesh networking, whereas chapters 5 onwards describe the Telegesis AT command set R3xx based on EmberZnet in more detail.

 New features in version R309

Please note that when referring to the ZigBee standard and unless otherwise noted the reference is to the ZigBee PRO feature set, which was released with the latest revision of the ZigBee specification in December 2007 [1].

Do not use this guide if you are using the older R2xx firmware based on EmberZNet2.x as the AT Command set is different. Please check the Telegesis website www.telegesis.com regularly for updates.

1.1 Radio Systems and ZigBee

A wireless system is effectively a system in which the wires between interacting devices have been replaced by a radio link. The upside of a wireless network is that it eliminates the need for messy cables and enhances the mobility of the installation, but there are also downsides which need to be looked at when replacing a wire with a radio link.

First of all there is the potential for interference blocking the radio signal from passing between devices. This interference may be from other wireless networks or from physical obstructions that interfere with the radio link. The best way to avoid interference is to try and use a frequency channel not in use by any other wireless system. For this purpose ZigBee utilises a channel scanning mechanism on start-up of a network. Even when operating on the same frequency channel, standards based systems such as ZigBee and WiFi use channel sharing mechanisms to allow sharing of channels. Finally ZigBee uses mesh networking technology which offers alternative routes to avoid local interference.

Secondly it is important to understand that all devices operating on the same radio channel within range of one another are sharing the maximum available bandwidth of this channel. In a wired system this would be similar to all devices sharing the same cable (bus system).

Finally security is also a major concern of all wireless systems as breaking into a wired system requires tapping into cables, whereas a wireless system can be compromised without physically touching any hardware of the system itself. ZigBee is offering a set of security services around AES-128 encryption allowing the system designer to maintain high levels of network security.

1.2 OSI 7-layer Model

The transmission, reception and presentation of data through a communications network involve a large number of functions: A network must be established, a link set up between at least two points, the data must be gathered from the application program, suitably formatted, probably divided into chunks and transmitted, then it needs to be verified on receipt and sent to a file or application program. For transmission, the data has to be modulated to suit the physical channel and possibly multiplexed with other signals.

To simplify the software that is needed for all these operations, they are divided according to function so that each stage of the data processing can be carried out with little or no knowledge of the stages before and after it. This removes the need to have, for example, one web browser for an Ethernet link, one for a dial-up modem, and another for WiFi. A common representation of the data flow is the 7-layer Open systems Interconnection (OSI) model:

Layer	Functionality
7	Application
6	Presentation
5	Session
4	Transport
3	Network
2	Data link
1	Physical

Table 1: 7-layer OSI model

For a detailed description of the layer functions, see a reference such as en.wikipedia.org/wiki/OSI_model. Not all systems can be neatly divided in this way, and the Data Link layer is often sub-divided into a Logical Link Control (LLC) and an underlying Media Access Control (MAC) layer. In addition, you may sometimes come across references to Layer 0 (cables, connectors etc) and less seriously Layer 8 (ie. the user, as in “a Layer 8 problem”).

Ember’s ZNet protocol stack provides functions up to the Network layer, with the radio part of the Ember chipset providing the physical layer as well as parts of the MAC layer (lower part of the data link layer) in hardware. The Telegesis AT command set can be considered as a mixture of Network and Transport functions with additional build in functionality going up to the application layer.

1.3 Radio Basics

Radio is the wireless transmission of signals by modulation of electromagnetic waves with frequencies below those of visible light. Electromagnetic radiation does not require a medium of transport like sound waves. Data can be carried using electromagnetic waves by changing some properties of the radiated waves (modulation). The simplest modulation technique would be on-off keying (OOK) in which case the electromagnetic waves are simply switched on and off to transport information. Other modulation techniques include altering the amplitude or the frequency of the electromagnetic waves.

The use of the radio frequency spectrum is regulated by government agencies and by international treaties. Most transmitting stations require a license to operate defining the power levels, modulation types and whether the assigned frequency bands are reserved for exclusive or shared use. Most ZigBee implementations today use the 2400MHz frequency band, which allows worldwide license free operation as long as the local power limitations are not exceeded.

There are two additional bands defined by the ZigBee specification [1] (actually by the underlying IEEE 802.15.4 specification [1]), namely 868MHz which is license free in Europe and 913MHz, which is license free in the US. These two bands are not widely adopted for ZigBee applications due to the fact that they are not truly international and also because the maximum achievable data rate is much lower on these bands.

1.4 ZigBee Radio Details

In the 2400MHz band ZigBee uses Offset Quadrature Phase-shift Keying (OQPSK) as a modulation technique. This is a digital modulation scheme which effects the imposition of data on to the carrier frequency by changing its phase. A receiver will compare the received signal against its own

reference frequency and convert the phase changes of the received signal over a 16μs period into a pattern of binary 4-bit symbols. This constitutes a Direct Sequence Spread Spectrum technology, where the 250kbit/s data is coded as a 2Mchip/s data stream for improved immunity to interference.

All this happens within Layer 1 of the described OSI model and is out of the scope of the integrator of a modular solution. The modulation technique is just mentioned for informational purposes.

Of more interest to integrators will be the fact that within the 2400MHz band a ZigBee network can pick one out of 16 channels to operate on. The details of those 16 channels are shown below.

Channel No.	Centre Frequency (GHz)	Channel No.	Centre Frequency (GHz)
11	2.405	19	2.445
12	2.410	20	2.450
13	2.415	21	2.455
14	2.420	22	2.460
15	2.425	23	2.465
16	2.430	24	2.470
17	2.435	25	2.475
18	2.440	26	2.480

Table 2: ZigBee Channels and Frequencies

1.5 Propagation, Antennae

An antenna is an arrangement designed to emit or capture electromagnetic waves of a certain frequency span. There are two fundamental types of antennae, namely omnidirectional antennae radiating equally in all directions and unidirectional antennae radiating more in one direction than in the others. For ZigBee networks the desired antenna type is omnidirectional, because the intention is not to provide the best possible link between two fixed points, but to form a stable self-healing mesh network.

Please note that antennae can be detuned when brought into proximity of other objects like metal cases, the hand of an operator etc. Detuning means that the frequency range the antenna has been designed for is shifted resulting in reduced performance at the desired frequency band of 2400MHz. When designing in a ZigBee module it is highly recommended to stick to the design recommendations regarding to the antenna placement.

1.5.1

Also note that any obstructions between a transmitting antenna and a receiving antenna will have some impact on the maximum possible distance between the two devices.

Maximum Distance between two Devices

The maximum distance between any two radio enabled devices mainly depends on the transmit power of the transmitting device and the receive sensitivity of the receiving device. These power levels are generally expressed in dBm or decibels relative to 1mW.

Knowing the output power as well as the input sensitivity together with potential cable losses and antenna gains allows the link budget to be calculated as shown in the following table.

ETRX2HR TX Power (boost):	+	5dBm
Transmitter Antenna gain:	+	2dBi
Receiver Antenna gain:	+	2dBi
Transmitter cable loss:	-	2dB
Receiver cable loss:	-	2dB
ETRX2HR Receive Sensitivity (boost):	-	-99dBm
Link Budget:		104dB

The link budget is a good indicator on the maximum achievable range (a higher link budget is better). As a rule of thumb it can be said that the distance between two nodes can double with every additional 6dB link budget. As a starting point, the theoretical loss between two isotropic antennas 1 metre apart is about 40dB at 2.4GHz.

“But how many meters/feet does this result in”, is a question which gets asked very often and although this is a simple question the answer isn't quite as simple. The actual distance very much depends on the environment you intend to use the radio in as hardly any application can benefit from a line of sight scenario with the transmitter and receiver slightly elevated and no other obstacles in the way. This environment is usually used for range tests providing results which cannot be reproduced with the same setup being tested indoors for example in an office building.

Because of this it is highly recommended to do actual tests in the target environment and assuming that an additional 6dB will double the range, it is easy to then estimate the impact of a power amplifier or an antenna with a higher gain. However, due to reflections the inverse-square law of power vs distance may not be very accurate; in corridors and tunnels it may approach a more linear law best expressed in dB/m, and over longer distances the exponent may increase from the theoretical 2 to 2.5-3.

As a starting point it is safe to assume that a non-power amplified Telegesis ETRX2 can be used to bridge 100-300 metres outdoors, whereas with the power-amplified ETRX2-PA distances in excess of 1.2km can be attained. The range of an ETRX357 is typically around 400m.

1.6 Networking

For a device to be part of a network it has to be connected in a way allowing it to exchange data with other members of the network. The type of connection can be via an individual network cable running via a switch, or a single cable or wireless link shared by all participants of the network.

To allow data to be exchanged between devices on the network a set of rules has to be defined setting permissions for nodes to send data into the network and also defining addressing, error checking, acknowledgement and encryption schemes. All these details are defined in a protocol.

A *networking protocol* commonly uses the services of another, more fundamental protocol to achieve its ends. For example, the AppleTalk Data Stream Protocol (ADSP) uses the Datagram Delivery Protocol (DDP) to encapsulate the data and deliver it over an AppleTalk network. The protocol that uses the services of an underlying protocol is said to be a client of the lower protocol; for example, ADSP is a client of DDP. A set of protocols related in this fashion is called a protocol stack.

2 ZigBee technology

2.1 The ZigBee standard [1]

The ZigBee Alliance is formed by a group of companies working together to enable reliable, cost effective, low power, wireless networked, monitoring and control products based on an open global standard.

Any company using ZigBee technology in a commercial manner should join the ZigBee Alliance at one of the three available membership levels:

1. Promoting member
2. Participating Member
3. Adopting Member

Membership guarantees access to all draft documents of the individual working groups within the ZigBee Alliance and in addition to that the first two levels allow a member company to gain voting rights in any of the workgroups by regular participation.

Ember, a promoting member of the ZigBee Alliance from the founding, and Telegesis, a long standing participating member, both worked actively within the ZigBee Alliance to work towards an open interoperable standard, and will continue to do so as part of Silicon Labs.

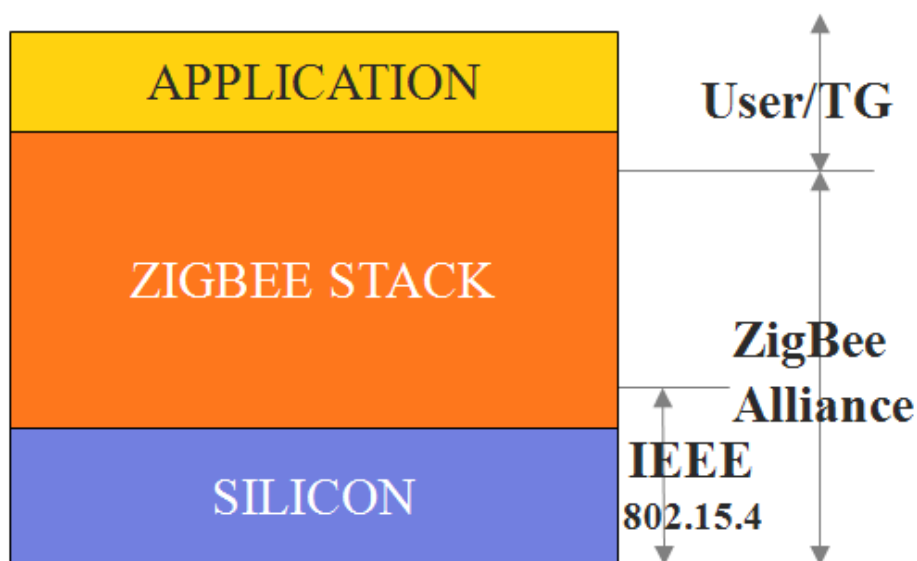


Figure 2: ZigBee

Because ZigBee is committed to open and interoperable devices, standards have been developed from the physical layer through the application layer as shown in Figure 2. On the lower layers of the OSI model (Physical Layer (PHY) and lower Data Link Layer (MAC)) ZigBee adopted the existing IEEE 802.15.4-2003 standard [1], adding the networking and transport layers as well as security services on top. On top of this Telegesis had developed an AT command interface, which ships with every module. This interface can firstly be used to interface with a host computer or microcontroller which is running the application, and secondly the AT command firmware can also be used to represent the application itself, so that simple use cases can operate without the need for an external controller.

At this point it should also be flagged up that it is perfectly possible to use the Telegesis range of ZigBee modules together with custom firmware which has been designed using the Silicon Labs suite of design tools.

2.2 Network Topologies

Before looking at the available topologies in detail it should be outlined that the main differentiator of ZigBee compared to any other conventional point-to-point or point-to-multipoint radio technology is that it is capable of forming a self-healing mesh network.

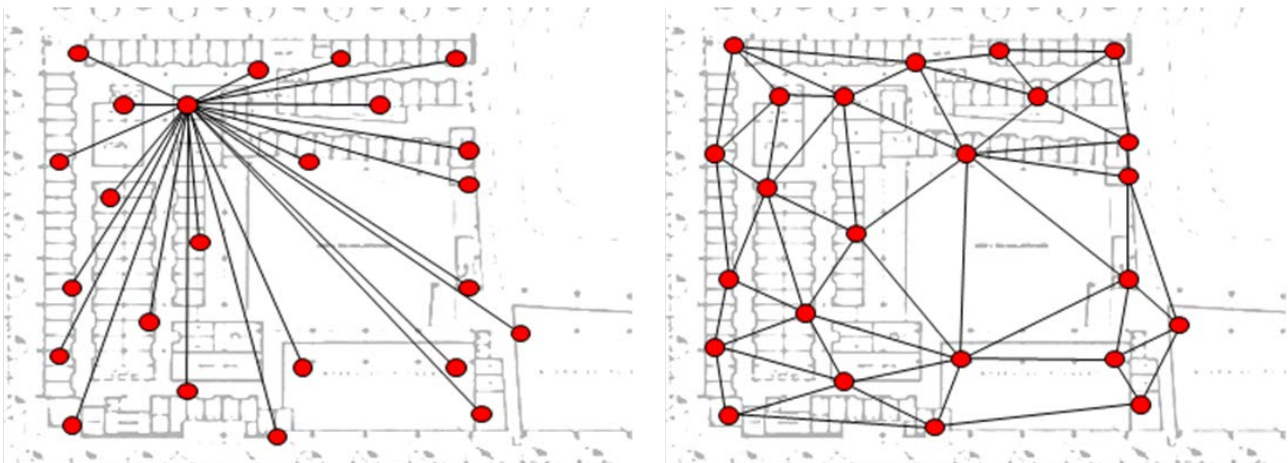


Figure 3. Point-to-Point vs. Meshing

Figure 3 shows a conventional point-to-point network like Bluetooth or WiFi on the left and a meshed ZigBee network on the right. Looking at Figure 4 it becomes obvious that breaking just a single RF link will cut individual devices off the network in case of the point-to-point network, whereas a broken link can be easily overcome by using another path of the wireless mesh network shown on the right hand side of Figure 3.

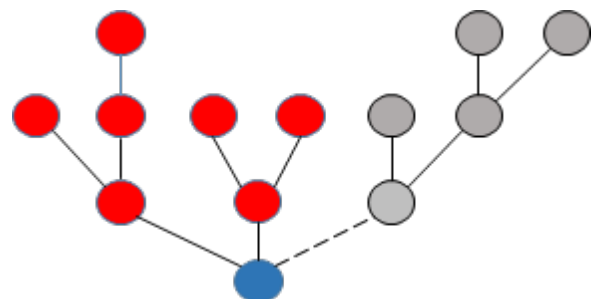


Figure 4. Tree topology

ZigBee supports two basic network topologies:

- Tree Topology
- Mesh Topology

The tree topology uses a simple routing mechanism in which nodes branch out in a tree like fashion from a central coordinator. Although the ZigBee specification allows a mesh to be superimposed on the tree structure this is not what one would call self-healing mesh networking. With tree routing breaking a single node, or a single RF link between two nodes can cut an entire branch of the tree network (without superimposed meshing).

For this reason Telegesis decided to only support the mesh topology with the Telegesis AT command set, which was introduced with the ZigBee PRO feature set as part of the 2007 release of the ZigBee Specification [1].

Within the mesh topology there is also a differentiation between a full mesh, a star and a hybrid mesh or cluster tree network. This can be misleading as any ZigBee PRO compliant router will automatically participate in a full mesh network. Referring to Figure 5, a mesh consists of routers with each node being able to exchange messages directly with any other node that is in radio range. It can also exchange messages with all other nodes as intervening routers will relay messages through the network. A true star consists of a router surrounded by end devices; the end devices are unable to communicate directly. A hybrid mesh has several star groupings with the parent devices forming a mesh. It is also possible for all these arrangements to occur in a network consisting entirely of routers, with the existence of each link determined by the quality of the radio reception. All this describes the potential data flow within a network; a network is not truly a star, for instance, just because all the data flows in and out of a central node; its routing protocol remains that of a mesh.

EmberZNet
does not use
beacons

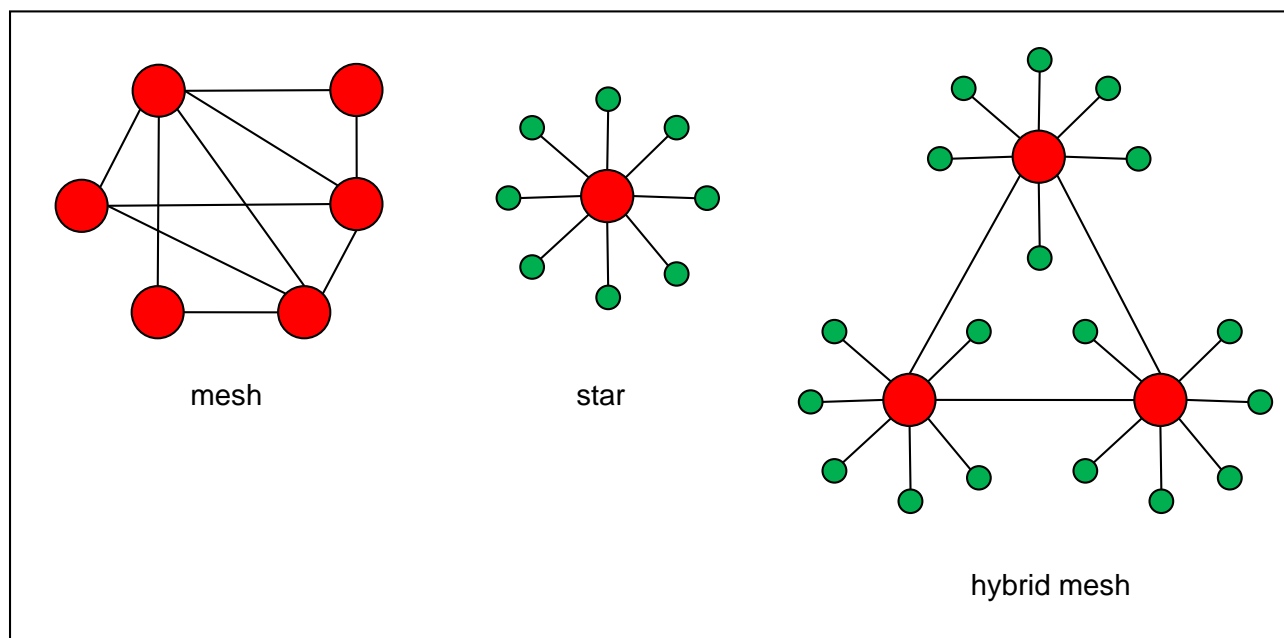


Figure 5. Network topologies

Due to the nature of wireless mesh networking the following limitations need to be understood:

- A ZigBee network is not real-time capable and the amount of time taken to send a message from one node in the network to another node is neither predictable nor measurable as the route the message takes may change.
- The maximum number of hops a message can travel is 30
- All nodes within radio range of one another share the available radio bandwidth, including those in other networks

2.3 Long & short addresses

Every node in a network requires an address. The destination address of a message can be specified as the 64-bit EUI64 which is flashed on to the EM250 chip and cannot be changed (the long address) or an ephemeral 16-bit short address. The short address or NodeID is assigned with a random value to the node when it joins a network, so if it leaves the network and rejoins again later on its short address is almost certain to be different. The exception is the coordinator which always has address 0x0000.

The user cannot assign or change a short address

The short and long addresses of a router's neighbours can be found from the AT+NTABLE command. Its own short and long addresses are in registers S04 and S05 respectively.

2.4 Endpoints

Each ZigBee device implements one or more endpoints within its firmware application, which can be considered a virtual device akin to ports associated with an IP address. Each is addressed by a number from 0 to 240; endpoint 0 is fixed and is referred to as the ZigBee Device Object. Messages to and from the ZDO use profile 0x0000 and are concerned with the basic network operations, with device and service discovery, and with binding. The Telegesis R3xx firmware uses endpoint 1 to handle its messages, so any radio packet received at endpoint 1 will be interpreted as though its message cluster conforms to the Telegesis manufacturer specific profile. It will be ignored if its header does not contain the profile ID 0xC091. Endpoint 2 can also be defined as active by setting bit B of register S0A; endpoint 2 can then reply to ZigBee Device Object requests from other devices performing device and service discovery. The content of endpoint 2's replies is defined in registers S48-4C.

2.5 Binding

Binding is a method of defining the destination address of a message without specifying it in the command that initiates the message. It relies on a device maintaining a local binding table; when the firmware is commanded to send a unicast message to an unspecified address, it searches through its binding table for entries that match the source endpoint and message cluster ID of the message to be sent. Each table entry defines a destination address and destination endpoint and the firmware transmits the message to each destination whose table entry matches the source endpoint and cluster ID. Binding table entries are created by using the AT+BSET command on the local device or the AT+BIND command on a remote device, which then sends a Bind Request message. Naturally, devices that do not use the Telegesis command set can accept the Bind Request messages according to the ZigBee specification [1] as AT+BIND uses standard ZCL commands, and similarly a device running Telegesis R3xx firmware can accept binding requests from devices running other firmware.

In Figure 6 a command is passed down with an unspecified destination address and endpoint. The binding table is searched until a match for that type of message is found, and the destination details can then be added into the message for further processing by the stack.

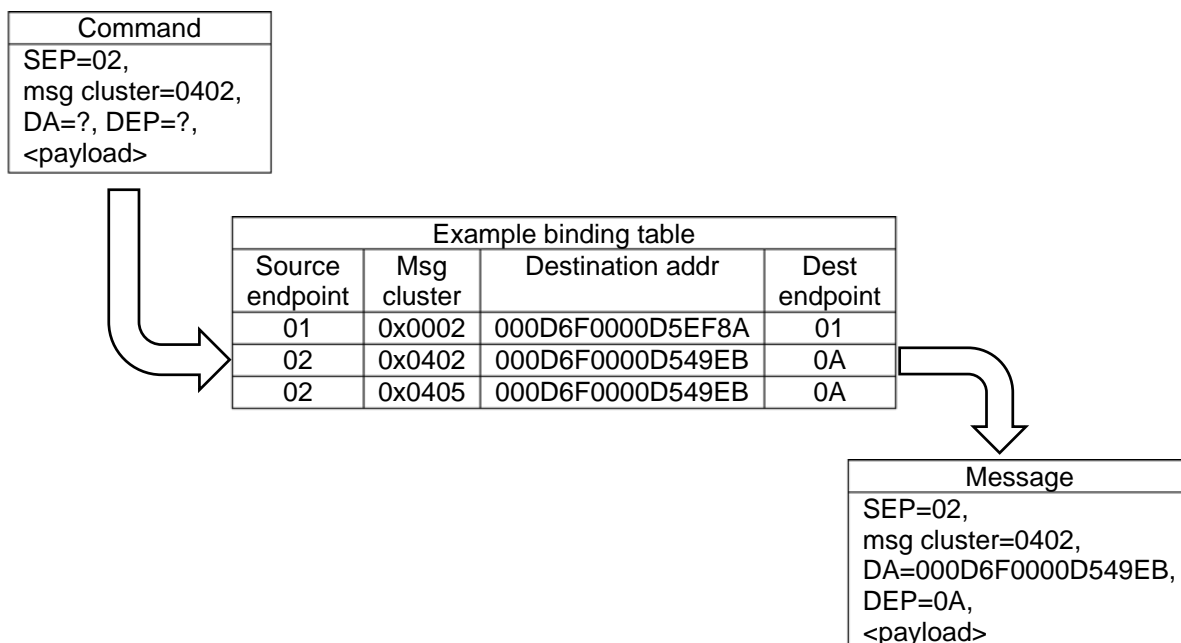


Figure 6. Example of sending a message via a binding table

While a device can add an entry into its local binding table, there is usually not much advantage in this since it can add the destination address directly into any unicasts it needs to send. More useful is to send a Bind Request message to another device, usually specifying the local device as the destination of the binding entry. The local device thereby subscribes to messages sent by the remote device without the latter needing to take any action or to know in advance the addresses of the devices that wish to receive its messages. A typical example would be a clock device operating as a time server, and sending regular reports of the time to other devices that need this information.

Source address	Source endpoint	Cluster ID	Dest address mode	Dest address	Dest endpoint
----------------	-----------------	------------	-------------------	--------------	---------------

Figure 7. Bind Request message

Figure 7 shows the contents of a ZCL Bind Request message. “Source” and “destination” are defined from the viewpoint of the device that contains the binding table, so “source address” is the device which receives the Bind Request message. Destination address mode allows for the destination to be a single device or a group, and we will not go into the details of that here. The destination address is usually the device that sent the request.

End Device Binding

While sending a Bind Request to another device is a useful facility, the local device may not always know the address of the other device, which might also be a Sleepy End Device with a slow response time which can lead to the Bind Request suffering a timeout error. Hence there is a more advanced protocol called End Device Binding. When two devices wish to bind to each other, each sends an End Device Bind Request to the coordinator stating which message clusters they wish to have bound to another device, as inputs or outputs. The coordinator compares the two requests and where their content is in agreement it forwards a Bind Request to either or both sending details of their partner. The requests are often sent in response to a button-press so they are not synchronous, and the coordinator matches up requests that arrive within a given time-window.

An End Device Bind Request contains this information:

- Binding target
- Source address
- Source endpoint
- Profile ID
- List of input message clusters
- List of output message clusters

(One of the cluster lists may be empty). The binding target is the address of the device to which the bound messages are to be sent; the source address defines the device which sent the bind request. These need not be the same as one device can issue a request on behalf of another.

When the coordinator receives two requests, it first compares the profile IDs and takes no action if they differ. Then it compares the input cluster list of each request with the output cluster list of the other, and notes the matching items. Finally it sends one or more Bind Request messages to set up the links between the two devices. For example, a coordinator receives two such requests as shown in Table 3:

End Device Bind Request	Binding target	Source address	Source endpoint	Profile ID	Input cluster list			Output cluster list		
From Device A	Short addr of A	Long address of A	SEP on A	Profile ID	CL1	CL 2	CL4	CL10	CL11	CL12
From Device B	Short addr of B	Long address of B	SEP on B	Profile ID	CL10	CL11	CL12	CL1	CL2	CL3

Table 3. End Device Bind Requests received at coordinator

It responds by sending a sequence of Bind Requests to Device A which meets A's search for targets for its output clusters (Table 4). A then knows where to send its messages. Destination address mode 3 means the 64-bit destination address and the destination EP are both present in the command.

Bind requests to Device A					
Source address	Source endpoint	Cluster	Dest address mode	Dest address	Dest endpoint
Address of A	SEP on A	CL10	03	Address of B	SEP on B
Address of A	SEP on A	CL11	03	Address of B	SEP on B
Address of A	SEP on A	CL12	03	Address of B	SEP on B

Table 4. Bind Requests to Device A

Then the coordinator sends similar Bind Requests to Device B (Table 5). Note that in this example there is no Bind Request for cluster CL3 because it was not in A's list of input clusters.

Bind requests to Device B					
Source address	Source endpoint	Cluster	Dest address mode	Dest address	Dest endpoint
Address of B	SEP on B	CL1	03	Address of A	SEP on A
Address of B	SEP on B	CL2	03	Address of A	SEP on A

Table 5. Bind requests to Device B

This is a complete example, but in practice it is more likely that an End Device Binding process between two devices will only set up message addressing in one direction. The term “End Device Binding” is a slight misnomer because it is not essential that any of the modules are end devices, they can equally be routers or a coordinator.

2.6 Device Types

A ZigBee network can consist of three different device types:

- ZigBee Coordinator (ZC)
- ZigBee Router (ZR)
- ZigBee End Device (ZED)

ZigBee Coordinators, Routers and End Devices were formerly known as Coordinators, Full Function Devices and Reduced Function Devices. For ZigBee End Devices Ember has added a further separation into Sleepy End Devices (SED) and Mobile (sleepy) End Devices (MED).

A ZigBee Router can route messages between ZigBee devices and forms part of the network’s routing infrastructure, whereas a ZigBee End Device does not provide any routing capabilities. The primary difference between the two is that a ZR needs to be constantly awake in order to fulfil its routing responsibilities, whereas a ZED can spend its life in sleep mode as it has no routing responsibilities.

ZRs consequently consume more power than ZEDs, therefore you must evaluate which type of device will suit your application best. See Section 12 for a list of example scenarios.

A Coordinator is effectively the Router which started the network and in most cases the coordinator also serves the purpose of the Trust Centre (ZigBee PRO security is explained in more detail in Section 6). By definition each network must have only one coordinator.

A ZigBee End Device has no routing responsibilities and joins the network via a ZigBee Router (its “parent”). The parent will buffer up incoming messages for its “child” if the latter is permitted to go into a sleep state. A child which may sleep should poll regularly to check if there are any new messages pending on the parent device to prevent loss of messages as any pending messages will time out on the parent after three seconds. The parent will also act as a relay point for its children’s outgoing messages. For example when a child wants to send a broadcast it sends the data to its parent and the parent in turn will initiate the broadcast.

To change a device type it must leave the PAN and join again

There are three different types of ZigBee End Devices defined in EmberZNet:

- sleepy end device
- mobile (sleepy) end device
- ZigBee End Device (unqualified)

The sleepy end device will usually remain in its parent's child table for about five minutes after the most recent poll (or until the parent is reset or leaves the network by itself). This implies that the sleepy end device should not physically move away from its parent and join a new parent, as it will still block a new child entry at its previous parent if that parent's child table is full. The five-minute decay time is a typical value and is the default setting of the Telegesis R3xx firmware.

A mobile sleepy end device will time out and be erased from the parent's child table if it does not poll its parent within 3 seconds. When a mobile end device cannot find its parent it assumes that it has moved and will search for a new parent. A device need not be defined as "mobile" merely if it changes its physical location; the key point is it that moves enough within the area covered by a PAN to need a new parent. If, for example, the PAN only contains one parent and a number of end devices (a "star" topology), there is no point in defining the ZEDs as mobile since there is no other parent to adopt them.

A ZigBee End Device, without being defined as an SED or ZED, joins the network as a leaf node that cannot function as a router, and on joining notifies its parent that it will remain permanently awake (technically, it has the constant RxOnWhenIdle set to TRUE). Therefore it is not permitted to enter a sleep state as its parent does not buffer incoming messages but instead forwards them immediately. Regular fast polling for incoming messages is not required and so this feature can be turned off in the Telegesis R3xx firmware, but slow polling is needed to keep the parent's child table updated. The Telegesis R3xx firmware normally does this once a minute as a result of built-in function 8104 in Timer/Counter 2.

In the Telegesis AT command software each routing device can support several end devices, in any combination of sleepy end devices, mobile end devices and ZigBee End Devices, but the exact number varies between firmware revisions so you should consult the relevant AT command manual [3].

A module becomes a coordinator by the application issuing AT+EN as described above. Whether a device joins as a ZigBee Router or sleepy/mobile end device is determined by the settings in S0A at the instant the device joins a PAN, so if a device is currently joined as a router and you want it to become an end device you have to complete the following steps:


1. Leave the network
2. Change S0A accordingly
3. Re-join the network

An end device will always keep its radio switched off (unless used) and therefore consume less power than a router, even when fully awake.

In power levels 0 to 2 (see section 7.8.4) by default an end device will poll its parent for new data once every second via the built-in functionality. This timing can be altered or disabled altogether if required. Polling takes less than 10ms, so the battery life of end devices can be maximised. One incoming broadcast is stored on each parent until overwritten by a new one. Unicasts can only be stored for as long as the timeout period on the sending device, so when expecting to receive unicasts you are required to poll at least once every three seconds, unless you increase the S4F timeout in the parent or are prepared to rely on message repeats.

When sending a unicast from an end device the unit will keep polling for the acknowledgement until it has been received or the transmission has timed out regardless of the configured polling interval.

2.7 Power modes

In order to save power, Sleepy end Devices and Mobile End Devices may be put into a low-power state where various functional blocks are turned off. There are five power modes, 0 to 4, with mode 0 consuming the most power (power mode 4 is a new feature in R309) . The default state is power mode 0 where the device is fully functional. In power mode 2 the radio and microcontroller core are off but the timers are still running, and in power mode 3 the timers are turned off as well. This means that in power mode 3 a device can only be woken up by an external interrupt. Power mode 4 behaves as mode 3, except that when the device goes to sleep it does not wait for any pending acknowledgements, as an unacknowledged message would leave it in its wake state. The power mode is determined by the value of register S39.

If a device in power mode 2 has a timer running that causes it to send periodic transmissions by means of a built-in function (eg function 0108), it will first return to power mode 0, send its data, check for any acknowledgement and incoming messages, then return automatically to power mode 2. It is not necessary for the user to manually change the power mode each time. A device in power modes 3-4 that is commanded to transmit by an external interrupt will behave similarly.

There is as yet no provision in the ZigBee specification or in our firmware for synchronised sleeping and wake-up of an entire network of routers. Individual routers should not be put to sleep because this can interfere with the normal meshing functions of the network, and ZigBee End Devices (not sleepy or mobile) must not be put to sleep because their incoming messages are not buffered in their parent.

The actual power consumption in the various sleep states is affected by the way the protocol stack wakes up the various functional blocks of the EM357 chip, so it is somewhat dependent on the firmware version. For this reason the detailed figures are presented in the AT command manual [3] rather than in any of the hardware manuals.

2.8 Route discovery

Where it is not possible to send a message between two nodes because of poor reception it can be routed via intermediate nodes. This is done automatically by the protocol stack and the user does not usually need to pay any attention to it. As a brief description of the route discovery process, a unicast message to another node is at first preceded by a Route Request packet targeted at the destination, which is repeated by all the other nodes in the network. When the destination receives it, it responds with a Route Reply, which again propagates back to the original source. As each node receives the replies from the destination or other repeaters, it is able to compare the quality of each individual link and the cumulative path cost back to the destination, and thereby build a routing table for each destination. The routing table is not wide as it only contains the next hop address for each destination, but it may become very long if traffic passes between many different nodes. The memory space for the tables is limited, so old routes are discarded to make room for new ones and may need to be refreshed by a new route discovery process. Routes are set up as required, so a device does not automatically know a route to every other device in the network. Where a router is in direct radio range of the local node routing is obviously not needed, so before sending a message a device refers first to another list of nodes known as the neighbour table. The neighbour table is populated from messages that each router broadcasts every few seconds which contain the contents of its own neighbour table. If a destination is found in the neighbour table a message can be sent immediately; if not, the routing table is checked, and if the destination address is not found there then a route discovery process is started. The routing table therefore contains the addresses of the first hop along the route to each recently-accessed router that is not in range, plus the first hop or parent of end devices that are not the children of the local device.

In situations where the data flow is one-to-many, ie a central node transmits to many others, the routing table can become full and the oldest entries have to be overwritten; this is especially likely if the destinations are end devices because they may need an entry even if they are in range. If the sender cycles round all its destinations, it can find that it has lost the routing table entry for each device and needs to re-discover the route for every single message. This needs a broadcast message to detect a device by its network address, which may itself need a broadcast to turn an EUI64 into a network address. However, a device is limited to sending about eight broadcasts in an eight-second period so in this situation the rate at which unicast messages are sent can be severely limited.

This many-to-one situation can often be foreseen by considering the network topology and the traffic flow, but other symptoms are frequent lost messages and large numbers of unexpected AddrResp and SR prompts. The solution is to ensure that the sender is either the coordinator or a sink, and store the information from the SR prompts to recreate source routes, as described below. Coordinators and sinks are normally defined as low-RAM concentrators, ie they are not expected to be able to store much source routing data and source routes are set up very frequently. When the source routing data is off-loaded in to a host processor to overcome the many-to-one traffic problem, these nodes can be defined as high-RAM concentrators which results in less network traffic.

The AT+SR command can prevent lost messages in large networks

A common situation is a many-to-one network of sensors or controllers communicating with a central node, usually the coordinator. This would force the coordinator and possibly its immediate neighbours to maintain routing table entries for nearly the entire network, with continual churning of the table and repeated route discoveries as described above. To avoid this, such a system will use a one-to-many routing procedure known as source routing, the implication being that the source dictates the route rather than the network having to find one. A message sent to the coordinator (or to a sink) is preceded by a Route Record packet which accumulates the addresses of all the nodes it passes through. The coordinator therefore has a ready-made route for its response, and will send an outgoing message with all the routing data included in it. This shortens the available payload but reduces the amount of network traffic.

If there is a significant time between the incoming and outgoing messages at the coordinator, it will have discarded the routing data. It is possible for a host processor to capture and store the routing data by saving the SR: prompts that are produced when a Route Record packet arrives. Just before a message needs to be sent from the coordinator, the route can be re-inserted by using the AT+SR command. This technique can also be used to overcome the limitations of the routing table in one-to-many situations.

3 Public & private profiles

3.1 Profiles and message clusters

802.15.4 [1] only provides for four types of radio packet: beacon, data, acknowledgement and command. All the various ZigBee messages have to be transported as data which requires a header that can distinguish them, so each ZigBee data packet header contains a Profile ID and Message Cluster ID. Associated messages and their attributes are grouped into a cluster; for example Message Cluster ID 0x0006 is the On/off cluster, which is sent with the parameter 0 or 1. A profile is a set of defined devices which provide the functions for an industry sector, with a list of mandatory and optional message clusters for each device type. The format of the message clusters is defined in the ZigBee Cluster Library specification, along with some non-profile specific commands, mainly to do with the reporting of cluster attributes. Some profile specifications also define message clusters that have been added to meet the requirements of that profile, but it is expected that they will eventually migrate to the cluster library specification. The various profiles share many message clusters as shown in Figure 8 to simplify the task of defining them, writing the applications and performing compliance testing. A profile specification, for example Home Automation, defines which messages clusters a compliant device shall or may generate or recognise. The message cluster specification can be considered as a code-book; only when the recipient of a message knows the profile and message cluster ID's can they know how to interpret a message, since the binary content of a message will have a different significance in the context of each application. The ZigBee Alliance publishes various public profiles and message clusters, but manufacturers can also write their own message set and request a Manufacturer Specific Profile ID. The Telegesis AT command firmware is such an MSP which allows a message to be interpreted, for instance, as either a unicast text message to go to the serial port or a text string to be written into an S-register.

Mixing different profiles in the same PAN can result in messages that are not understood

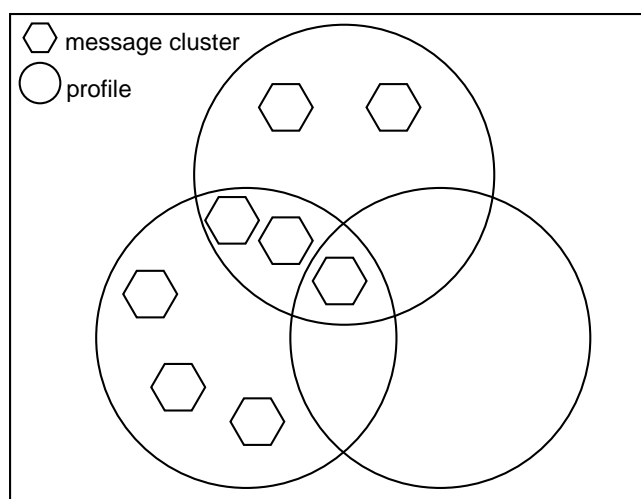


Figure 8. Profiles and message clusters

3.2 The Telegesis AT command set

In order to relieve users of the need to write their own firmware and to understand the fine details of ZigBee networking, Telegesis have developed a command set that can be used either from a host program or interactively through a terminal application such as Telegesis Terminal or HyperTerminal. An interactive session is the easiest way to understand the usage and effects of the various commands. The commands described in this manual relate to version R3xx of the firmware (currently R309). The commands are based on the now-traditional Hayes modem commands, but do not conform to any particular standard.

Each command is a string of ASCII characters which starts with “AT” and is terminated by a <cr>. The commands cannot be concatenated into one string. Where a command sends a text message (eg AT+UCAST) the payload of the message can contain any character that the application program can produce and is not limited to printable characters, provided that it is compatible with the UART settings (eg you obviously cannot send the whole of an 8-bit character if the UART uses only 7 data bits). The exception is the <cr> character 0x0D which will terminate the command and leave the remainder of the string to be interpreted as the next AT command, almost certainly leading to an error message. If your application is generating random characters or if <cr> must be included in the message, you can use the so-called binary version of the command AT+UCASTB. In this case the command’s parameter is not the string itself but its character count; the string is entered after the ‘>’ prompt returned by the firmware and needs no terminating character. (Note that if your terminal software uses xon-xoff handshaking, it will strip out characters 0x11 and 0x13 from your data stream.)

The main features of the firmware are the AT commands, the S-registers and the built-in functions. Each of these three will be described in this document, but to supply all the details of each command and register would only duplicate the contents of the R3xx AT Command Manual [3]; the latter manual should therefore be consulted in conjunction with this user guide.

This document was written when the latest firmware version was R309 but the general principles and most of the details of the firmware also relate to earlier versions. R309 is not available for the ETRX2 module, for which the latest version is R308.

4 Network Management and Maintenance

4.1 Overview

A ZigBee network is formed from the inside out. This means that a device which is instructed to start a network becomes a so-called coordinator and can scan all available radio channels, pick the one with the least traffic and generate a random PAN ID in order to start a network; other nodes can then join in.

Another scenario is that a node scans for an available network to join, and if no network is found the node can choose to become a coordinator and start its own network as described above. Remote nodes are then capable of finding this newly established network and joining it. As more and more remote nodes join, the network grows from the inside out. If the network is spread over a wide area, a new node may be out of range of the coordinator and will join the PAN via another node, which will act as a router. Messages may then pass across the network in more than one “hop”.

The Telegesis AT command-driven software supports point-to-point unicast messages and broadcast messages travelling up to thirty hops through the network.

The principal commands used to establish and maintain a network are described in section 5.3, and refer to the appendix for a typical sequence. For a list of all available commands please refer to the AT-Command dictionary document. With a set of modules which are not in a PAN and have their S-registers in the default state, it is sufficient to send the AT+EN command to one device to create a network; all the others will join automatically in a minute or two.

A device is commanded to leave its own network with the command AT+DASSL; it can command another device to leave with the AT+DASSR command. It is important to note that the coordinator is the sole trust centre in a ZigBee network, ie only the coordinator can grant permission for another device to join. Therefore if the coordinator leaves the network no new devices can join, nor can the old coordinator regain access to the network. It may then be necessary to dissolve the network and reform it around a new coordinator. Since a network is stable without a coordinator, built-in function 001E can be used to make devices leave automatically if no coordinator is detected.

Register S0A is fundamental to the formation of a network, as it defines the security configuration and whether a device joins as a router or an end device.

4.2 Channel mask

Register S00 holds a 16-bit word in which each bit represents one of the available 802.15.4 channels 11-26 (MSB=26, LSB=11). Setting a bit to 1 enables a channel, and setting a bit to 0 disables it. It is therefore possible to mask out channels which are known to be noisy in the chosen application environment. Setting the mask to fifteen 0s and a single 1 will force operation on one radio channel. Some typical channel patterns and masks are shown in Table 6 and Table 7.

	Channel																Note	Applies to			
	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26		ETRX2	ETRX2-PA	ETRX357	ETRX357-LRS
PA																	1				
PA																	2				
Crosstalk																	3				
WiFi																	4				

Table 6. Channels to avoid

Mask	Channel																Note	Application	
	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26			
0xFFFF																	5	Detect all networks	
0x7FFE																	6	ETRX2-PA: avoid band edges	
0x0FFF																	7	Prevent crosstalk in ETRX2	
0x1FFE																			
0x3FFC																			
0x7FF8																			
0xFFF0																			
0x6319																	8	Avoid WiFi	
0x0040																	10	Select channel 13	

Table 7. Example channel masks

Notes on Table 6 and Table 7:

1, 2, 5. A power-amplified device has restricted performance on the highest and (with the ETRX2-PA) lowest channels, so they are best avoided.

3, 7. It has been known for the ETRX2 to detect a network 60MHz (12 channel spacings) above or below its true frequency. This can lead to an unstable network and can be prevented by blocking the outermost channels of the band. These masks (apart from 0xFFFF0 and 0x0FFF) also block the highest and lowest channels so suit the power-amplified devices as well. The phenomenon is referred to as crosstalk, and it does not occur with the ETRX357.

4, 8. WiFi devices may use one of three carrier frequencies which overlap with a block of ZigBee channels. They do not exactly coincide with ZigBee channels so the best channel mask cannot be easily determined from the WiFi spectrum, but this channel mask is recommended in the ZigBee Smart Energy and Home Automation profile specifications.

5. A mask of 0xFFFF allows a device seeking a network to detect everything in the vicinity.

9. A mask of 0x6318 avoids WiFi, prevents crosstalk, and can be used with the power-amplified devices.

10. Starting a network on channel 13 is the first step to recovering a failed over-the-air firmware cloning operation.

4.3 Network size limits, max number of children

The theoretical limit to the size of a network is almost 65535, set by the 16-bit network address of each module, but this is clearly unattainable. A tree network has a limit set by its pre-determined width and depth parameters, but a mesh is constrained by the degradation in performance as the available radio bandwidth becomes congested. Routers exchange network maintenance messages every few seconds, and end devices have to send regular data requests at a rate chosen by the user, which each device having to wait for a quiet channel. However, a large network might consist of islands of nodes with little radio contact between the islands, so the traffic within each group may have little effect on the capacity of the other groups. The maximum size is therefore indeterminate but users' experience suggests that networks of a several hundred nodes can operate satisfactorily.

4.4 Maintaining a Network

A network is self-healing, which means that if a certain path between two nodes breaks, the mesh will find a new path to make communication possible, unless one of the nodes has simply gone out of range of the entire network.

Each node periodically checks it has not lost its neighbouring devices; if it has lost connectivity to its PAN, it leaves the network (default function of Timer 2). A module that is not part of a network periodically searches for one that it can join (default function of Timer 3); hence a node may temporarily drop out of the net and come back in again, possibly via a different parent node if it has moved. If there is a risk that a node may drop out and be re-captured by a different PAN, security modes should be invoked (see section 6).

Once the network has been established you may wish to keep this network private so other nodes do not interfere with it. The first method is to prevent more nodes from joining (register SOA, bit 5), the second is to only permit secured joining from the beginning (see section 6 for security). Attempts to achieve privacy by setting radio channel masks or preferred PAN IDs may not be reliable since the network's environment cannot be predicted. This is discussed more fully in section 4.5.

4.5 Network commissioning

In most cases it is sufficient to command one device to establish a PAN, after which the other devices will join it automatically. As mentioned in section 5.3.6, it may be necessary to disassociate devices from a pre-existing PAN first. When some users use a pre-defined radio channel and PAN ID a particular situation can arise which they did not anticipate, as follows:

In this scenario, all the user's devices are given a chosen radio channel and PAN ID. At each installation, one device is commanded to establish a PAN. In the factory, there is also a PAN for configuring the devices; modules join this PAN, are powered down without being disassociated from the PAN and are delivered to site in that state. It is assumed that when powered up they will immediately communicate with the local coordinator, since they are already in a PAN with the correct parameters, but in fact they are found to be completely out of contact with the rest of the network.

The cause of this problem is the network security key in register S08. When left with its default value of all-zeros, the key chosen by the coordinator is not all-zeros but a random value. Therefore devices commissioned in the factory are almost certain to have a different key from the ones at each site and so cannot communicate with the local coordinator. The key is always used for message encryption even if secured joining is not in effect.

The solution is to specify a common key value in S08; it is not necessary to alter the contents of S0A that relate to secured joining. In most cases it is sufficient to command one device to establish a PAN, after which the other devices will join it automatically. However, it is sometimes necessary to configure the S-registers of each device to suit your application and it may be more convenient to do this at the manufacturing stage rather than on-site for each individual system. Commissioning is usually done in one of four ways:


1. Send AT commands from a PC to the equipment through a serial port connector of some sort
2. Send AT commands from an on-board host processor
3. Use a bed-of-nails tool or similar to configure the ZigBee modules before PCB assembly
4. Create a commissioning radio network and configure each device over the air. When simple devices with no serial port access are being processed, this may be the only option

Since a commissioning radio network must start with the devices in their default state, the network must be unsecure, ie have no preconfigured link key. One option is to give all the devices a chosen radio channel and PAN ID to distinguish them from other networks; it is then important to disassociate them from the commissioning network since each PAN has a unique network key, so devices cannot simply be moved from one PAN to another. A better option is to write a link key into register S09 and write a suitable value into S0A, then disassociate the device. This method gives a better level of security and permits the local coordinator to select its own channel to avoid interference.

5 The AT command set

5.1 Introduction

Commands are sent to a device in the form of text strings that start with "AT" and end with a carriage return character. Only one command can be sent on each line, and it is recommended that the application should wait for a suitable response, usually "OK", before sending the next command.

This list of commands is not exhaustive as several need little or no explanation. Consult the AT command manual [3] for a full list. Commands and features marked  are new to R309.

5.2 AT commands for module information and configuration

ATI - Display Product Identification Information

Almost the simplest command, it returns a description of the module type and its EUI64 address.

ATZ - Software Reset

- 5.2.1 Resets the processor and restarts all the timers from zero, but the device does not leave the network. The neighbour table, address table and multicast table are cleared.

5.2.2 AT+REMZ - Soft Reset of Remote Device

As for ATZ, but resets a remote device.

5.2.3 AT+BLOAD – Enter the Bootloader Menu

- 5.2.4 The bootloader runs at 115200 baud, and can be used to update the R3xx firmware or to load a completely different application file. The firmware is not altered by merely starting the bootloader, so it is a safe operation – just select option 2 to run the existing firmware. For more information about AT+BLOAD and AT+RECOVER, refer to the ETRX357 Development Kit Product Manual.

5.2.5 ATS and ATREMS – Read and Write S-Registers

Refer to section 7.2 (ATS & ATREMS – Read and write S-registers).


5.3 AT commands for configuring a network

5.3.1 AT+ESCAN – Scan the energy of all channels

- “AT+ESCAN” scans all channels which are not masked out in register S00 and displays their energy levels, enabling the user to manually choose a quiet channel on which to establish a PAN. This is the level of background noise which will include ZigBee networks, but it is not the RSSI of the devices in those networks. As with the AT+PANSCAN command, the module can remain in a PAN while it performs the scan.

AT+PANSCAN – Scan for all available channels

- “AT+PANSCAN” will produce a list of all available channels from which you can manually choose the one you wish the unit to join. To perform this you need to use the “AT+JPAN” command. R3xx firmware can perform the channel scan while the module is still a member of a PAN (with R2xx the module had to be disassociated from the PAN first). “AT+PANSCAN” causes an active scan which triggers replies from the other devices, rather than a passive scan which merely listens.

In R309 the user can define the scan time and a channel mask, and filter the results by joining permission. 

AT+EN – Establish Network

When the local module is not a member of a network, it can initiate a new one and become that network’s coordinator. Issuing the “AT+EN” command scans all the available channels, looking for

the one with the lowest energy level; it then generates a random PAN ID, checks to ensure that a network with this PAN ID does not already exist and then starts a PAN. A unit on which this command has been executed automatically becomes the network's coordinator and remains the coordinator until it leaves the network using "AT+DASSL". Neighbouring nodes can now join this newly established network using "AT+JN".

Register S02 allows you to select a preferred PAN ID. By default this register is set to FFFF, which causes the coordinator to pick a random PAN ID when establishing a PAN. Any number between 0000 and 3FFF will cause the unit to start a PAN with the specified PAN ID unless that particular PAN ID is used by another network. In this case, a randomly generated number is allocated instead.

AT+JN – Join Network

5.3.4 The node will scan all available channels for a PAN which allows units to join. The available channels are defined by the channel mask, and they are scanned in numerical order. When the device finds a channel with available networks, it discards those with a poor signal quality and attempts to join the network whose responding router is closest to its coordinator. If this fails it tries the remaining networks, and if necessary moves on to the next channel. This is a very convenient way of joining a network; it does not, however, give the local node a choice of which network to join if there is more than one network available.

AT+JPAN:cc,pppp - Join specific PAN

5.3.5

cc is the channel number and pppp is the PAN ID of the PAN which is to be joined. The channel number overrides any filtering applied by the S00 channel mask.

5.3.6

AT+DASSL – Disassociate Local Node

5.3.7 To join a different network, or to change a device's type from FFD to SED and vice-versa, a node must first disassociate itself from its current network using the "AT+DASSL" command. Do not disassociate the coordinator unless you are closing down its network and have removed all the other devices, because the coordinator cannot join the network again.

Think twice before
disassociating a
coordinator

AT+DASSR – Disassociate remote device from PAN

5.3.8

To move a node from one PAN to another, it must first be disassociated. This may be necessary when it is intended to swap a coordinator unit for another (which will initiate a new PAN).

5.3.9

AT+N – Display network parameters

"AT+N" reports the type of the local node (see section 2.6), the channel number and PAN ID, and the transmitter power.

AT+SN – Scan network for other nodes

"AT+SN" can take a parameter to specify the number of hops the search should cover (the radius; 00 to 30). The entire network will be scanned if the parameter is set to 00 or 30 or omitted). Optionally, the RSSI and LQI readings between the remote device and the local device can be displayed if bit C of register S10 is set as described in section 7.6. When measuring the RSSI it is advisable to set the scan radius to 1, as described under register S10.

AT+ANNCE – Indicate presence in the network

A remote device can indicate its presence in the network by pinging out its ID when instructed by the “AT+ANNCE” command. All devices receiving the ping that have bit C of register S10 set will display the RSSI and LQI.

AT+NTABLE – Display Neighbour Table

5.3.10

Shows a list of all the routers that are within one hop, plus (for a router) all the end devices that are its children. It also shows the Link Quality Index of the received signals, which is a measure of the chip error rate. By using AT+NTABLE to examine the neighbour list of each remote device it is

5.3.11 possible to build up a map of the network connectivity.

AT+CCHANGE – Change the network’s channel

The network manager (which is normally the coordinator) can command the entire network to change to a different channel. The process will take around 10 seconds in order that the command can propagate to all the end devices. If the network manager receives a lot of NM:ES prompts indicating a high incidence of message failures, this command can be used in conjunction with AT+ESCAN to move the network to a quieter channel.

5.3.10

5.4 AT commands for obtaining device information

Introduction

5.4.1

The basic ZigBee specification [1] provides several ZigBee Device Object (ZDO) commands that request information about a device. These are of greatest use when the R3xx firmware is configured to interact with devices that use another manufacturer’s firmware or another profile such as Home Automation, as they enable you to verify that a device is of the correct type and communicate with it using the right endpoint and message clusters.

5.4.2

AT+IDREQ – Request network address

5.4.3 Request Node’s Network address, given its EUI64. If the device has children, an optional index parameter will cause them to be listed as well.

AT+EUIREQ – Request EUI64

5.4.4 Request Node’s EUI, given its network address. As with most of these ZDO commands, the address has to be given twice. This is because the ZigBee specification allows for the request to be sent to one device to obtain information about another device; although this behaviour is not currently implemented in this firmware it makes the commands fully compatible with any other device. Again, an index parameter will list the target’s children.

AT+NODEDESC – Request Node Descriptor

Request a device’s Node Descriptor. This gives basic information about the node type (FFD/SED), manufacturer code and other capabilities. Refer to the ZigBee specification or the AT command manual [3] for more details.

AT+POWERDESC – Power Descriptor

Request Node's Power Descriptor. The power descriptor is displayed as a 16 bit hexadecimal number as described in section 2.3.2.4 of the ZigBee specification.

AT+ACTEPDESC – Request Active Endpoint list

5.4.5 Request Node's Active Endpoint list. It is important that messages sent to a device are directed to the correct endpoint. Also, this enables you to query each endpoint in turn and obtain a list of the message clusters they support.

5.4.6

AT+SIMPLEDESC – Request Simple Descriptor

5.4.7 Request Endpoint's Simple Descriptor. This returns its profile ID, Device ID, and lists of its input and output clusters. Input clusters feature on the server side of a device, and output clusters on the client side.

AT+MATCHREQ – Find nodes with specified clusters

5.4.8 Find nodes which match a specific descriptor. The command requires a Profile ID and a list of input and/or output message cluster IDs, and is sent as a broadcast. Every device in the network that supports at least one of the clusters responds with its network address and relevant endpoint number.

5.5 AT commands for data transmission

5.5.1

Overview

The purpose of a network is - of course - to send data between nodes. The data (formatted as successive bytes) can be of predefined or undefined length, or may be a single 16-bit or 32-bit word controlling the ETRXn's I/O ports. It can be sent to a single node, or broadcast across the network.

ZigBee has been designed for control and automation applications, therefore the maximum payload for any message type mentioned in this section is 82 bytes, which reflects the small and efficient messaging structures of ZigBee (the exception is a packet of raw data sent by the AT+RDATAB command, which can hold 114 bytes).

The strength of a meshed network compared to traditional point-to-point radio systems is that data can be sent between nodes which are out of range of one another. The packets containing the data dynamically route through the network finding the best possible path. In doing so the mesh network can dynamically react to a changing RF environment and self-heal broken links, given that an alternative route from the source to the destination of the data is present.

The integrity of message transmission is enhanced by the use of application layer end-to-end acknowledgements. The R308 firmware allows 1.6 seconds for a unicast message to reach its destination and for an acknowledgement to arrive back at the sender, which corresponds to a delay of about 20ms through each node of a network with the maximum size of 30 hops, plus around 200ms to encrypt and decrypt the message. Although the propagation delay through an ETRX357 can be only a few milliseconds, there is an initial random backoff before transmission and there may be further backoffs if the radio network is congested so 20ms is representative of an average delay. If a message is not acknowledged at the application layer it is sent a total of three times, and after

4.8 seconds a NACK prompt is reported. If a message is not acknowledged even at the MAC layer, it is sent four times at about 4ms intervals according to the 802.15.4 specification [1], and the stack sends each burst five times at about 50ms intervals. Therefore typically sixty attempts are made to send a message before a NACK is reported. Figure 9 shows the sequence of the acknowledgements at the two layers.

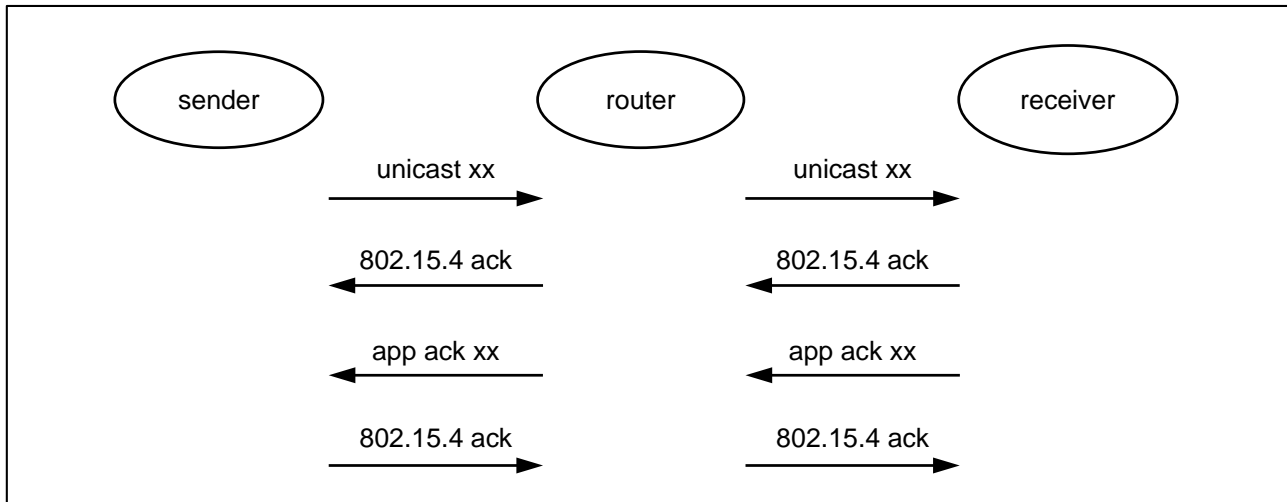


Figure 9. Acknowledgements for unicast message number xx

It can happen that an outgoing unicast is properly delivered but the application layer acknowledgement does not arrive back at the receiver. The consequence is that the sender sees a NACK response and the receiver gets three copies of the message. The most common cause of this effect is the use of a high-power ETRX357-LRS module for the sender and a lower-power ETRX357 for the receiver, which creates an assymetric link where the receiver can hear the sender's radio packets but not vice-versa. The effect can also be created by Sleepy End Devices with unsuitable timeouts and polling rates, or premature switching between the wake state and the sleep state.

Radio packets are protected against errors by a 16-bit CRC. If a CRC error is detected the entire packet is rejected; there is no MAC layer acknowledgement and the packet is not passed up to the higher layers of the stack, because there is no guarantee that the PAN ID or destination address are correct.

5.5.2

The destination address in the AT command set can be defined in several ways, as described in 7.2.17.2.1 (ATREMS – addressing modes).

Broadcasts – AT+BCAST

“AT+BCAST:nn,<data>” sends the string <data> to all nodes within nn hops. Broadcasts are not acknowledged, so they are sent three times to improve the likelihood of reception. However, reception cannot be guaranteed. Each receiving node sends the message “BCAST:<EUI64>,xx=<data>” to its serial port (<EUI64> being the transmitting node).

“AT+BCASTB:xx,nn” will generate a ‘>’ prompt in your terminal window, where a number of xx characters may be typed. The character string can include <CR>, <LF> and non-displayable bytes. Each receiving node sends the message “BCAST:<EUI64>,xx=<data>” to its serial port (<EUI64> being the transmitting node and xx the string length).

As each node hearing a broadcast within the specified amount of hops is also repeating it, it becomes obvious that broadcasts consume a lot of bandwidth. Because of this the ZigBee specification limits the amount of broadcasts which can be used to a total of 8 in every 8 seconds.

The maximum number of hops a broadcast can bridge is 30 hops.

Raw data – AT+RDATAB

If you need to send data as rapidly as possible at the expense of some reliability and security, you can broadcast it using the AT+RDATAB command. Up to 114 bytes can be sent in a packet that bypasses the ZigBee layers of the protocol stack, and hence has no encryption or error checking.

5.5.3 AT+RDATAB:xx" is similar in format to "AT+BCASTB:xx,nn". This must be used with care as it is completely non-ZigBee compliant.

Unicasts – AT+UCAST

5.5.4 "AT+UCAST:<address>,<data>" and "AT+UCASTB:xx,< address >" function in a similar way to their broadcast equivalents. However, the data is only sent to the specified node and "ACK" or "NACK" is sent to the sender's serial port. Note that "NACK" may result from a positive "ACK" response being lost in transmission. <address> can be an EUI64, a 16-bit network address, the index of an address table entry, or the index of a binding table entry. Address FE causes a search of the table for the first entry whose source endpoint and cluster ID matches registers S40 and S42. Address FF refer to the local device for some commands. With address FE the device ought to send a unicast to every address in the binding table that matches S40 and S42, but in fact only the first matching entry is used. This is a minor deviation from the ZigBee specification.

To allow for multiple messages in flight, a transmission number is issued when successfully sending a unicast and at this point it is possible to make further transmissions. The follow-on ACK or NACK prompt is also followed by that transmission number.

5.5.5 If you prefer to wait until the ACK or NACK has been received simply set bit B of S10 and no transmission number will be issued. Instead the response will be either "OK" if an ACK has been received, or an error code where a NACK has been received or no feedback was received at all.

Unicasts – AT+SENDUCAST



5.5.6 When it is desired to use a profile ID, message cluster or endpoints that differ from the default parameters, storing the new values in S-registers is convenient when they only need to be changed infrequently. When several different parameters are in use it can be more convenient to incorporate them in the AT command itself, so AT+SENDUCAST and AT+SENDUCASTB take as parameters the values that would otherwise be written into S40, S42 and S44.

S-Casts – AT+SCAST

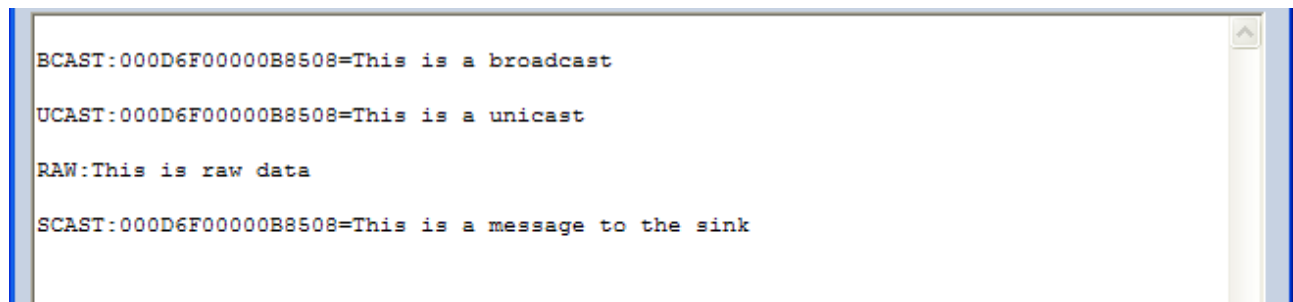
A sink (sometimes called a concentrator) is a node intended as a central receiver of data. A node other than the coordinator can be selected to operate as a data sink. A node may be changed to be a sink by setting bit 4 of S-register S10. With the R3xx firmware there may be more than one sink in a network, but the users cannot predict which sink will be selected by each device unless they examine its address table.

In order to send data to the network's sink an individual node needs to know the sink's address. This can be done by searching for a sink using the "AT+SSINK" command, alternatively the sink can

advertise its services in regular intervals as set by the sink's Timer/Counter 1 (see explanation of the built-in functionality in section 8). When a sink has been found its address is written into entry 5 of the address table and can be inspected with the command AT+ATABLE. This entry can be overwritten to select a different sink.

If a sink is known "AT+SCAST:<data>" sends a message to the sink and returns "ACK" or "NACK" as with a unicast. S-casts have the advantage that the identity of the sink does not need to be known by the user in advance, which simplifies the writing of application software.

"AT+SCASTB" allows the sending of a predefined number of characters including <CR>, <BS> in the same way as AT+UCASTB and AT+BCASTB.



```
BCAST:000D6F00000B8508=This is a broadcast
UCAST:000D6F00000B8508=This is a unicast
RAW:This is raw data
SCAST:000D6F00000B8508=This is a message to the sink
```

Figure 10. Examples of received messages

5.5.7 Multicasts – AT+MCAST

5.5.7.1 Introduction

There may be occasion when you want to send messages to sub-sets of a network, for example to control different groups of devices. To do that you can define multicast groups and use the AT+MCAST(B) command. Each device in the network can retain a multicast table that controls which groups it belongs to; the messages are sent as broadcasts with a multicast group ID in their header, and each receiving device filters the message and either ignores it or passes the text to its serial port. It is important to remember that multicasts are a form of broadcast and so restricted to the same traffic limit of eight in any eight-second period. Each device can belong to up to five multicast groups, so various sets of devices to be controlled can overlap. Multicast messages cannot be sent to end devices.

The multicast table is erased whenever the device boots up, so entries 00 and 01 may be pre-defined in registers S3E and S3F.

5.5.7.2 The multicast table – AT+MSET

For a device to filter multicast messages there need to be suitable entries in its multicast table. You can view the table with the command AT+MTABLE:

AT+MTABLE		
No.	ID	EP
00	D0D0	01
01	FEED	01
02	4567	01
03	0000	00
04	0000	00

Figure 11. Multicast table

This table was created with the three commands:

AT+MSET:00,D0D0,01

AT+MSET:01,FEED,01

AT+MSET:02,4567,01

which makes the device a member of three groups. Note that any 16-bit ID number can be assigned to a group, but the endpoint EP must be 01 in order for the incoming messages to be processed by the AT command firmware. The AT+MSET command can only edit the table of the local device, but registers S3E and S3F can create entries 00 and 01 automatically at boot-up and these registers can be written from a remote device.

5.5.7.3 The multicast command – AT+MCAST

Having set up the multicast tables as required, we can now send multicast text messages, for example:

AT+MCAST:00,4567,Hello world

5.5.8 Since this is a broadcast spanning the maximum number of hops it goes to every device in the network. Only devices belonging to group 4567 will accept it and send the text string to their serial port, but they will not send an acknowledgement as it is a broadcast message.

The multicast command – AT+SENDMCAST



5.5.9 As with AT+SENDUCAST, AT+SENDMCAST and AT+SENDMCASTB take the profile ID, message cluster ID and endpoints as parameters, which is more convenient than writing to the relevant S-registers if they change frequently.

Display Address Table – AT+ATABLE

The user can create entries in the address table and use the index of an entry as an abbreviated destination address. Entry 5 is automatically populated with the address of the network sink, if one is known. Entry 6 contains the address of the device which last sent a unicast message to the local node, enabling the user to easily send replies.

If an address is written manually into entry 5, the device assumes that this denotes a sink and will use that address for AT+SCAST messages and for the built-in functions that transmit to the sink. It

is not necessary for the destination node to have declared itself to be a sink, it receives the messages anyway.

Channels – AT+DMODE

A unicast has a maximum payload of 82 bytes, so if you want to send more than that, or if you want to send continuous data in both directions, a so-called channel is needed.

5.5.10 To the outside world a channel appears like a wireless cable replacement, but please note that you should not expect high data-rates, as ZigBee has been designed for sensor and control applications for distributed networks rather than for high data throughput. When using a channel you may need to implement XON/XOFF or hardware handshaking, otherwise you will risk losing data due to a buffer overflow.

To open a channel use the “AT+DMODE:” command followed by the address of the device to which you want to open that channel. Alternatively you can use the “Open” button in the “Channel” section of the Telegesis Terminal Software. By default the remote node will automatically accept the channel if bit E of S10 is cleared. Once a channel has been established, data can be sent bi-directionally. To close the channel send “+++” on either end with a pause of at least 500ms preceding it (“Close” button).

Figure 12 shows a sequence of messages when using a channel. **Heavy text** is the data entered at the keyboard, normal text is the responses. Note that the channel is bidirectional, and there is no local echo of the data (since it would be indistinguishable from data received). Normal prompts and error messages are also suppressed.

At node 000D6F0000594B00	At node 000D6F0000594C81
AT+DMODE:000D6F0000594C81	
SEQ:07	
OK	
ACK:07	
DataMODE:07C8,000D6F0000594C81,00	DataMODE:0000,000D6F0000594B00
OPEN	OPEN
asdad	asdad
123123	123123
+++	
CLOSED	CLOSED

Figure 12. Channel messages

This command corresponds to the AT+OPCHAN command of R2xx, but the name was changed as “channel” can be ambiguous in a radio network. R2xx also had the AT+OPLCHAN command, but there is no longer any need for an equivalent as the +++ sequence can be included in the data stream provided it is sent more than 250ms after the preceding character.

Sending messages between PANs – AT+INTERPAN



5.5.1 Normally there is no communication between different PANs because received messages are only accepted when they have the correct PAN ID. However, unencrypted messages between PANs on the same radio channel can be sent by using the AT+INTERPAN command. They only travel one hop, broadcasts are only sent once and are not echoed by the receivers, and interpan messages cannot be sent to end devices. The format is:

AT+INTERPAN:<AddressMode>,<DstAddress>,<DstPAN>,<ProfileID>,<ClusterID>,<Payload>

Addressmode can be 00, 01 or 02 to select a network address, group ID number or EUI64. To send the message as a broadcast, use mode 01 and group ID FFFF.

Payload is written as a sequence of hexadecimal values.

The other parameters have their obvious meanings. For example, to broadcast the string “Hello” within PAN 1234 you can send

AT+INTERPAN:01,FFFF,1234,C091,0002,48656C6C6F

At the receiver, the profile ID and message cluster are merely displayed as part of the prompt, and there is no source or destination endpoint. The example above results in the prompt:

INTERPAN:C091,0002,08,0002,,412A,000D6F0000D5EF8A,05:48656C6C6F

5.5.1 08 signifies a broadcast, 412A is the PAN ID of the source. Refer to the AT command manual for the full details of the command and prompt parameters.

Messages to and from other endpoints

When working with a network that contains devices with firmware other than R3xx, it will probably be necessary to communicate with endpoints other than 1. Register S40 sets the source and destination endpoints in the message headers; the source endpoint is not usually important but devices sometimes take it as a guide as to which endpoint they should use as a destination.

Similarly, registers S42 and S44 set the message cluster ID and profile ID respectively in the message headers. This enables you to send messages that conform to any profile, such as Home Automation. For more details refer to [1].

When the R3xx firmware receives a message that is not addressed to endpoint 0 or 1, or does not specify profile ID 0xC091, the message is usually ignored. A suitable setting of bits 1 and 8 of register S0F causes these unidentified messages to be passed to the serial port as an “RX:...” prompt which can be handled by a host processor. The exact behaviour of these two register bits varies a little between firmware versions so be sure to refer to the appropriate AT command manual [3].

5.6 AT commands for binding

Introduction

Please refer to section 2.5 for a description of binding between ZigBee nodes.

AT+LBTABLE – Print binding table

5.6.1 This command simply prints the binding table of the local device, and has no parameters.

AT+BSET – Write to binding table

5.6.2

AT+BSET creates an entry in the next free space of the local binding table.

AT+BCLR – Clear binding table entry

5.6.3

AT+BCLR deletes one or all of the local binding table entries.

5.6.4

AT+BTABLE – Print remote binding table

5.6.5

AT+BTABLE prints the binding table of a remote device. As with AT+NTABLE the radio packet has limited capacity and may not be able to contain all the entries, so the command has an offset parameter to specify the first entry in the list.

AT+BIND – Write to remote binding table

5.6.6

AT+BIND is the principal command for creating a binding table entry on another device. The target of the binding need not be the local device though that is the most common usage. See section 2.5 for the effect of a binding table entry.

5.6.7

AT+UNBIND – Clear remote binding table entry

This command removes an entry from a binding table by reference to the entry's contents, rather than by its table index number as with AT+BCLR. It is the only way to remove a binding on a remote device.

5.6.8

AT+EDBIND – Send End Device Bind request



AT+EDBIND sends an End Device Bind request to the coordinator as described in section 2.5.1. The full command with its parameters is

AT+EDBIND:[<target>,<SrcEP>,<ProfileID>,<NumInClusters>,<InClusterList>,<NumOutClusters>,<OutClusterList>

<target> is the network address of either the local device or its primary binding cache. For simplicity the address and the following comma can be deleted, to signify the local device.

The parameters NumInClusters and NumOutClusters must be 2-digit hexadecimal numbers; if either is 00 then the following cluster list will be empty as in these examples:

AT+EDBIND:01,C091,00,,01,0002

AT+EDBIND:01,C091,01,0002,00,

Successful binding results in a prompt such as “End Device Bind:0000,00”, where 00 indicates success.

5.7 Time-related commands

Introduction

The ETRX35x module can maintain an internal clock, which can be set by a local command or by synchronising it with a ZigBee time server. The R3xx firmware cannot act as a time server itself.

5.7.1

AT+SETTIME – Set local clock



The command takes a time as a parameter and initialises the internal clock to this value. The format is either:

<year>,<month>,<day>,<hour>,<minute>,<second> in decimal numbers, for example
AT+SETTIME:2014,04,01,23,59,59. All except the year must be 2-digit numbers

or

<seconds> as a 32 bit hexadecimal number representing time in UTC format (number of seconds since 01.01.2000 00:00), for example AT+SETTIME:18C3FD62 for 01/03/2013 23:59:30.

5.7.3

AT+GETTIME – Read local clock



The command returns the number of seconds since 01.01.2000 00:00 as a 32 bit hexadecimal number.

5.7.4

AT+SYNCTIME – Sync with time server



The format is AT+SYNCTIME:<Node ID>,<End Point>[,Profile ID]. If the profile ID is omitted it defaults to 0x0104 (Home Automation). Before using this command it is possible to seek a time server by using the command “AT+MATCHREQ:0104,01,000A,00” or similar. This requests a response from any device that uses the HA profile and has Time (0x000A) as an input cluster. The response will tell you which network address and endpoint to synchronise with.

6 Security

In ZigBee 2007 PRO [1] there are four principal security modes:

- No security. All messages are sent unencrypted.
- Standard (or Residential) security. Messages are encrypted with a network key. The Trust Centre sends the network key to new members of the network unencrypted.
- High (or Commercial) security. Messages are encrypted with a network key. The Trust Centre sends the network key to new members of the network encrypted with a Link Key (Master Key).

- APS layer security. (APS: Application support sub-layer) Standard and High security modes operate at the network layer so that a message is re-encrypted each time it passes through a router along a multi-hop route. APS layer security protects the payload of the message and encryption/decryption are only applied at the source and destination. Encryption uses a Link Key that is unique to each pair of modules and is set up using Certificate-Based Key Establishment. The primary use of this security mode is in ZigBee Smart Energy networks.

The exchange of messages when an FFD joins a network is shown below for the two available security levels. The process is similar when an FFD or SED joins via another router instead of the coordinator except that more messages are required to transfer the keys.

Direction	Message	Note
FFD b/cast	Beacon request	
COO b/cast	Beacon	Indicates whether joining is permitted
FFD→COO	Association request	
FFD→COO	Data request	COO waits for Data Request in case new device is an SED
COO→FFD	Association response	
COO→FFD	Network key	Not encrypted
FFD→COO	Request key	
COO→FFD	Link key	Encrypted with Network Key
FFD b/cast	Device announce	Causes “FFD” prompt in other devices

Table 8. Unsecure joining process (Standard security)

Direction	Message	Note
FFD b/cast	Beacon request	
COO b/cast	Beacon	Indicates whether joining is permitted
FFD→COO	Association request	
FFD→COO	Data request	COO waits for Data Request in case new device is an SED
COO→FFD	Association response	
COO→FFD	Network key	Encrypted with Link Key
FFD b/cast	Device announce	Causes “FFD” prompt in other devices

Table 9. Secure joining process (High security)

Using the Telegesis AT command interface software 128-bit AES encryption is always enabled and the settings of the main S0A function register decide whether new nodes trying to join the network need to have an identical network key (S08) or can obtain the key over the air from either a neighbour or the coordinator. For High level security all nodes must have a preconfigured Link Key in register S09. APS layer security is not a feature of this firmware. The coordinator is the sole Trust Centre and its role cannot be taken on by any other node.

With the older meshing stack EmberZNet2.x the coordinator was free to leave the network once it is established. With ZigBee PRO a coordinator must be present or else no new nodes can join the PAN. Therefore a coordinator cannot be replaced. In the R3xx firmware the user has the option of starting a PAN in distributed Trust Centre mode where any device can grant joining permission, but this is a deviation from the ZigBee specification.

A coordinator cannot be replaced. The user should consider the effect of this on network maintenance

Alternatively joining can be simply disabled by altering the settings of the main S0A function register. As the main function register contains security relevant settings, this register is password protected in the same way as the encryption key, the OEM word and the password itself. These registers can be written locally as well as over the air provided the user knows the password. The relevant bits in S0A are:

Bit	Meaning	Relevant to	
		COO	Others
A	When joining don't ask for Trust Centre link key		✓
9	Don't use central Trust Centre (distributed TC Mode)	✓	
8	Use Pre-Configured Trust Centre Link Key when joining		✓
7	Trust centre uses hashed link key	✓	✓
5	Don't allow nodes to join (TC setting)	✓	
4	Send Network key encrypted with the link key to nodes joining	✓	
3	Don't allow nodes to re-join unsecured	✓	
2	Send Network key encrypted with the link key to nodes re-joining unsecured	✓	
0	Don't allow other nodes to join the network via this node	✓	

Table 10. Security settings

The default setting of S0A is 0x0000 which results in Standard level security.

Bit A. A ZigBee 2006 network does not use a Link Key, so when a ZigBee 2007 PRO device joins a ZigBee 2006 network the link key request can be disabled.

Bit 8. When the Trust Centre imposes High level security each new node must apply its preconfigured Link Key in order to receive the correct value of the Network Key.

Bit 4. To impose High level security the Trust Centre must encrypt the Network Key.

Bit 2. Nodes can rejoin a network, either to obtain an updated network key in the event that they have missed a key update, or in the case of an end device to re-establish contact when it has decayed from its parent's child table. The device is not considered to have left the network, so it retains its network address and does not need to receive the keys. Following an unsecured rejoin request, the network key is normally sent unencrypted, but to remove this security risk bit 2 requires the Trust Centre to send the network key encrypted.

Bits 8, 4 and 2 are all that are required to set up High level security, so the simplest configuration is to set S0A=0114 in all devices.

Bit 0. A network can be closed to new members by setting bit 0 in all routers, including the coordinator. Beacon requests from a new device are answered with a beacon that contains the field "Permit association: false", so the new device does not attempt to join. Bit 0 can be set or cleared

selectively to control which routers will adopt end devices as their children. Bit 0 has the same effect as the ZigBee Device Object command `Mgmt_Permit_Joining_req`.

Bit 5. A network can also be closed to new members by setting bit 5 in the coordinator, so that it does not send the network key to new nodes. This does not prevent unwanted devices from trying to join, though, so the coordinator still has to handle the association requests.

Bit 7. When the Trust Centre hashes its preconfigured Link Key with the EUI64 of a new node, it creates a unique Link Key. The new node has to hold the hashed key in its S09 register. This mode appears to be rarely used.

Bit 9. To work around the restriction that only the coordinator can grant joining permission and therefore can never be removed from the network, bit 9 allows all routers to grant joining permission. This is like a ZigBee 2006 network, but it is not strictly compliant with ZigBee 2007.

As a final comment, if your system requires a high level of security and you have written a private ZigBee application, you should not use over-the-air cloning to distribute the firmware file as the radio packets are completely unencrypted and a sniffer could capture your entire file.

7 S-Registers

7.1 Overview

The ETRXn modules contain a number of items of configuration data (the “S-registers”) which are under the control of the user. Commands “ATS” and “ATSREM” read and write the register contents locally as well as remotely. They define, for example, the radio network parameters, I/O status, textual prompts and responses and timer operations. The timed functions are the key to much of the versatility of the ETRXn and detailed study of the AT command dictionary document is recommended.

For a reference of all S-Registers please refer to the AT command dictionary document. Most of the registers are non-volatile, retaining their values when the module is powered down. Note that some register changes take effect immediately while others are deferred – typically until a reset. The registers fall into groups, shown below:

7.2 ATS & ATREMS – Read and write S-registers

“ATSxx?” returns the contents of local register xx, while “ATSxx=XXXX” sets it to XXXX. The corresponding syntax for register xx of remote node <EUI64> is “ATREMS:<EUI64>,xx?” and “ATREMS:<EUI64>,xx=XXXX”; alternatively use any of the address modes described below. The S-registers control the module’s functions and I/O; for example “ATS18=00000000” clears all the bits of the output register S18, and for a module on a Devkit board lights up all the LEDs. “ATSxxy” or “ATREMS:<EUI64>,xxy” accesses bit y alone of register xx. The registers fall into groups, shown below.

ATREMS – addressing modes

Many of the AT commands take a device address as a parameter, which can usually be expressed in several different formats.

EUI64. 16 hexadecimal characters. This is flashed on to the chip at manufacture and cannot be changed by the user. This can be compared to the permanent MAC address of an IP-based device.

Network address. 4 hexadecimal characters. This is allocated to the device when it joins the PAN and cannot be changed or preset, except that 0x0000 is always the coordinator. It is analogous to a temporary IP address. Otherwise known as the Node ID.

Address table entry. Range 00-06. Entry 05 is a sink address, entry 06 is the source address of the last received UCAST, SCAST or MCAST.

Binding table entry. Range 10-24 (hexadecimal). Entry FE causes a search of the table for the first entry whose source endpoint and cluster ID matches registers S40 and S42.

FF. In many commands address FF represents the local device.

7.3 Radio Setup (S00-S03)

S00

The channel mask as described in section 4.2.

S01

The RF transmitter power level in dBm. This is the level out of the EM250 or EM357 chip, so consult the ETRX2-PA or ETRX357-LRS manual to see how this relates to the final output when a power amplifier is included in the module. An ETRX357-LRS is limited to -7dBm into the power amplifier which corresponds to 17.5dBm at the antenna, in order to comply with the unwanted emissions limit of the FCC Part 15 requirements.

S02, S03

The preferred PAN ID and Extended PAN ID. These can be used when establishing a PAN to give it a distinct identity and to ensure that devices join the correct PAN. The drawback however is that if the coordinator detects a PAN on the channel it has selected that already uses its preferred ID, it will choose a random value and that will prevent any device that also has S02 or S03 predefined from joining its PAN. This situation can arise when a coordinator has been disassociated from an existing PAN in order to start a new one, and the other members of the old PAN still remain. Secured joining using a predefined link key may be a more reliable method of distinguishing one PAN from another.

7.4 Module Setup (S04-S0A)

S04-07

The various long and short network addresses of the device and its parent (if it is an end device). These are pre-assigned; the EUI64 is programmed at the factory and the user cannot choose the network address.

7.5 Security keys (S08-S09)

S08

The network key, used to encrypt the messages. There is usually little point in assigning a value to S08 as the coordinator can select a random key and may update it at a later time. For security, registers S08 and S09 cannot be read back.

S09

The link key, used to encrypt transmissions of the network key in a secure system. Assignment of a link key also requires the setting of bits 8, 4 and 2 in register S0A.

7.6 Module Setup (S0A-S11)

S0A

Used to define whether a node is a router or end device, and to control the use and distribution of link and network keys and whether or not other devices can join the PAN. See section 6.

S0B

A user-definable test string that can be used to distinguish devices.

S0C

A user-definable password used to protect the important registers against inadvertent changes.

S0D

Device and firmware information extracted from factory settings.

S0E, S0F

Registers that allow the user to suppress various messages and remove unwanted information in order to simplify the parsing of the text strings produced by the device. The radio messages passed between the devices are not affected, merely the presentation of the text strings on the local module. S0F also activates the display of messages that arrive at endpoints 0, 2, 3 etc and are not handled by the firmware. Most of the messages arriving at endpoint 0 are sent by other devices to control the local module and extract information from it, so they will not be passed to the serial port. In addition, when S0F bit F is set the RSSI value is appended to all incoming text messages generated from AT+UCAST/BCAST/SCAST commands.

S10

Various settings relating mainly to the network sink:

Bit D defines the device as a high-RAM concentrator; see section 2.8 (Route discovery) for an explanation of this.

Bit C activates the display of the RSSI of the responses to the AT+SN command, which is useful in range tests and diagnosing unreliable links (note – when a message arrives through multiple hops, the RSSI is the signal strength of the nearest hop. Use AT+SN:01 to avoid ambiguities).

Bit B inhibits the transmission of a unicast message until the previous message to that destination has been acknowledged, instead of permitting several to be in flight at once; there is then no message sequence number.

Bit 3 allows changes to be made to the RF output power without the need to leave the PAN and join again.

Bit 1 causes unicast messages to be sent unacknowledged; message delivery failures are therefore not detected, but the reduction in network traffic may be beneficial.

Bit 0 causes the EUI64 of the source device to be omitted from unicast messages. This increases the available payload by 8 bytes but the prompt displayed at the receiver no longer identifies the source.

S11

Bit F activates the PWM output which can be used as an audible tone, or as a DAC output by altering the duty cycle and filtering the waveform. See section 9.1 (Tone Generation at Pin I/O3 or PB7).

Bit 9 enables the UART to return a device from Power Mode 2 to Power Mode 0 without the need for a built-in timer or an interrupt pulse on one of the digital inputs. The first character of a text string is used to generate the interrupt and so is lost, hence for a string to form a valid command it must be preceded by a dummy character.

Bits 8-0 control the input debounce and the sense of the edges of the pulses that are detected as interrupts.

7.7 I/O-related registers (S12-S22)

Serial port

S12

7.7.2 This register sets the serial port parameters including command echo and flow control. 19200 baud is usually adequate as the maximum achievable data rate between two devices is rarely more than 20kb/s, but where many devices send data to a central node it may be necessary to raise the baud rate of the receiver.

Input terminations

S13

On the ETRX2 this register is the pull-up enable register. By default all pull-ups are disabled. For current sensitive applications it is recommended not to use the built-in pull-ups. On the ETRX357 this register is used in conjunction with S16 and S18 to configure the I/Os.

S14

On the ETRX2 this register is the pull-down enable register. By default all pull-downs are disabled. S14 is not used on the ETRX357.

Alternate functions

S15

Not used on the ETRX2.

7.7.3 On the ETRX357, the digital I/O ports are by default simple inputs or outputs with nominal voltages of 0 or Vcc. Various other functions can be assigned to some of the ports by setting the appropriate bit of S15 (the index of each bit corresponds to the index of the I/O pin):

Bit 8: present Vref on pad PB0 during an ADC measurement. (This is the equivalent of setting bit D of S11 on an ETRX2.)

Bits 9, 10: activate the serial port connections TXD and RXD. This is the default state of the R3xx firmware.

Bits 17, 15, 14, 13: activate ADC inputs ADC3, ADC2, ADC1, ADC0 on pads PC1, PB7, PB6 and PB5 respectively. No ADC readings are available unless the relevant ADC has been enabled here.

Bit 21: activate the TX_active signal on pad PC5, which then goes high when the RF circuit is in the transmit mode. This signal is not available on the ETRX357-LR as it is used internally.

7.7.4 I/O pin control – ETRX2

(See below for notes on the ETRX357.)

Each module has three volatile registers representing the current state of its I/O, namely:

- S16 Data Direction
- S18 Output Buffer
- S1A Input Buffer

These registers can be read and written from a remote device as well as locally, so they are the key to remote sensing and control. S16 defines the data direction of each individual pin, where 1 represents an output and 0 represents an input, e.g. setting bit 7 to 1 will turn I/O7 into an output. By default S16 is defined to be 0x00F8 which suits the devboard having three inputs (Buttons 2-4) and five outputs (LEDs 1-4 and Beeper). Table 11 shows the mapping of the devboard pins to the I/O of the ETRX2 module.

Pin	Default direction	Default level	Devboard functionality	Default
I/O0	I	0	Button4	connected
I/O1	I	0	Button3	connected
I/O2	I	0	Button2	connected
I/O3	O	0	Button1 or Beeper	Beeper connected
I/O4	O	1	LED4	connected
I/O5	O	1	LED3	connected
I/O6	O	1	LED2	connected
I/O7	O	1	LED1	connected
I/O8	I	0	None	n/a
I/O9	I	0	None	n/a
I/O10	I	0	None	n/a
I/O11	I	0	None	n/a

Table 11. Development board pins

S18 defines the output level which is driven by the pin.

S1A reflects the digital reading of all the I/O pins, both inputs and outputs.

Example for an ETRX2 development kit board:

The four LEDs on the devkit are by default all switched off (the I/Os sink current through the LEDs). If we now wanted to switch on LED1 locally we need to take the following steps:

- Make sure the Data direction in S16 is set correctly (it is by default)
- Optionally, read output register S18
- Set bit 7 of S18 to 0 (note we can access each bit individually)

The described operations can be executed using the ATS commands as shown below (blank lines are omitted):

```
ATS16?
00F8
OK
ATS18?
00F0
OK
ATS187=0
OK
```

Table 12. Read/write operation

The same action can now be performed on a remote node using the “ATSREM” command. As you will see, you do not need a remote host processor in order to switch the external I/Os.

We can now check the status of the I/O by reading S1A. Try to read S1A on the local node and then read it again while pressing button 4 on the devkit. The results should appear similar to this:

ATS1A?
0F77
OK
ATS1A?
0F76
OK

Table 13. Change in input values

It can be seen that bit 7 of register S1A is still set to 0 since we had switched on LED1, which is connected to I/O7 previously (if you simply need to read the status of this bit you could also use ATS1A7?). You can see when pressing button 4, bit 0 of S1A changes to zero as pressing a button on the devboard forces the respective I/O to ground.

I/O pin control – ETRX357

Each module has three volatile registers representing the current state of its I/O, namely:

7.7.5

- S16 Data Direction
- S18 Output Buffer
- S1A Input Buffer

S16 defines the data direction of each individual pin, where 1 represents an output and 0 represents an input, e.g. setting bit 7 to 1 will turn I/O7 into an output. By default S16 is defined to be 0x000142CC which suits the devboard having three active inputs (Buttons 1-3) and four outputs (LEDs 1-4). Table 14 shows the mapping of the devboard pins to the I/O of the ETRX357 module and carrier board (CB).

Name	Index	Pad	Default direction S17= 0142CC	Devboard functionality	Alternate function
PC7	17	4	In		
PC6	16	3	In		
PC5	15	2	In		Enable TX_active on ETRX357
PC4	14	24	In		
PC3	13	23	In		
PC2	12	22	In		
PC1	11	26	In		ADC3 (light sensor)
PC0	10	27	Out	LED2	
PB7	F	28	In		ADC2, PWM
PB6	E	29	Out	LED1, Button 4, IRQ3	ADC1
PB5	D	30	In		ADC0 (temp sensor)
PB4	C	8	In		
PB3	B	6	In		
PB2	A	18	In		RXD
PB1	9	17	Out		TXD
PB0	8	25	In	Button 3, IRQ2	
PA7	7	5	Out	Red LED on CB	
PA6	6	16	Out	Green LED on CB	
PA5	5	15	In	(Bootload)	
PA4	4	14	In		
PA3	3	12	Out		
PA2	2	11	Out		
PA1	1	10	In	Button 2, IRQ1	
PA0	0	9	In	Button 1, IRQ0	

Table 14. Module pads and functions

S18 defines the output level which is driven by the pin.

S1A reflects the digital reading of all the I/O pins, both inputs and outputs.

Example for an ETRX357 development kit board:

The four LEDs on the devkit are by default all switched on (the I/Os sink current through the LEDs). If we now wanted to switch off LED1 locally we need to take the following steps:

- Make sure the Data direction in S16 is set correctly (it is by default)
- Optionally, read output register S18
- Set bit 7 of S18 to 1 (note we can access each bit individually)

The described operations can be executed using the ATS commands as shown below:

```

ATS16?
000142CC
OK
ATS18?
00000000
OK
ATS18E=1
OK

```

Table 15. Read/write operation

The same action can now be performed on a remote node using the “ATSREM” command. As you will see, you do not need a remote host processor in order to switch the external I/Os.

We can now check the status of the I/O by reading S1A. Try to read bit 0 of S1A on the local node and then read it again while pressing button 1 on the devkit. The results should appear as below:

```

ATS1A0?
1
OK
ATS1A0?
0
OK

```

Table 16. Change in input values

If you read the entire S1A input buffer it may be difficult to see whether the correct bit changes when you press a button, because with floating inputs several may change at the same time.

7.7.6

PWM control

S1B-S1E

These registers control the top value and compare value of the PWM generator and the default values that apply at boot-up. The top value sets the frequency of the square-wave output and the compare value adjusts the duty cycle. See section 9 for more details of the PWM output.

7.7.7

ADCs

S1F-S22

Reading of the A/D ports A/D1-4 (ETRX2) or ADC0-3 (ETRX357). The allowed input range is 0-1200mV on both devices.

ETRX357:

1LSB=100μV

Bits 17, 15, 14 and 13 of S15 must be set to activate all four ADCs

ETRX2:

1LSB=1mV

Bits C and B of S11 must be set to activate A/D3 and A/D4. A/D1 and A/D2 are always on

7.8 S-Registers Defining the Functionality of the Module

Introduction

7.8.1 The R3xx firmware is capable of performing a variety of functions without the need for an AT command each time an action is required. Functions can be carried out at boot-up, on joining a network, on the detection of an external interrupt pulse, or under the control of a built-in timer (either one-shot or repeated). In order to program a function, first select a suitable register according to how the function is to be triggered, then find the function code in the table “Built-in functionality” in the R3xx AT Command Manual [3] and write it into the chosen register. For a built-in timer the registers are arranged in pairs: the first is the delay to the first execution of the function (in 250ms increments) and then the repetition rate thereafter, and the second is the function code itself. The codes as listed in the AT command manual implement a one-shot operation, which is converted to a repeated function by setting the MSB of the code. For example, function 0037 (toggle I/O7 once) becomes 8037 (toggle I/O7 repeatedly), while function 2100 becomes A100.

Some of the counter/timer pairs are pre-configured, but the timing and function may be altered if desired.

7.8.2 Interrupts (S23-S28)

S23-S26

The external interrupts. Each register contains the code for a function to be executed on detection of an interrupt pulse on the appropriate input. Register S11 determines whether each interrupt occurs on a rising or falling edge, or both.

Register	ETRX2		ETRX357	
	I/O	Pad no.	I/O	Pad no.
S23	I/O0	26	PA0	9
S24	I/O1	25	PA1	10
S25	I/O10	30	PB0	25
S26	I/O11	31	PB6	29

Table 17. Interrupt registers and inputs

By default, S23 is configured with function 0001 (change to power mode 0). Always ensure that there is some way to wake up a sleepy device, as it will not respond to AT commands on its serial port when in a low-power state.

S27

A function that is executed at boot-up which can be used to configure the local device. The protocol stack is not running at this point so S27 cannot be used to send messages or report network-related information.

S28

A function that is executed when the device joins a network which can be used to configure the local device. Powering up or resetting the device causes it to go through a network joining process, so register S27 is invoked at boot-up and can be used to send a message.

Timers (S29-S38)

Registers S29-S38 define the functions of Timer/Counters 0-7. As described above, each timer/counter is a pair of registers counting 250ms increments, so the fastest rate is 4 events per second and the longest interval is 0xFFFF ticks, or about 4½ hours. The default settings with R309 firmware are:

Timer	Registers		Effect
	Interval	Function	
0	0004	8010	End device polls parent
1	00F0	821E	Sink advertisement
2	00F4	8014	Leave network if alone
3	00F2	8015	If not in a PAN, seek and join one
4-7	0000	0000	None

Table 18. Default timer functions

To remove a timer function, set either or both of the registers to 0x0000. The user can change the operation of the pre-defined timers, but you may need to familiarise yourself with some of the deeper working of the ZigBee PAN in case there are unexpected side-effects. The commonest is to slow down the polling rate of an end device to save power, to the point where it loses contact with the network because its parent has deleted it from its child table on the assumption that it has been lost.

7.8.4 Refer to section 8 for more information about the function codes themselves.

Power Management (S39-S3A)

Sleepy and Mobile End Devices do not take part in any routing functions, so they can be put into a low-power state with the radio switched off in order to save power. Synchronised power-down of an entire network is a feature that is not yet implemented in the ZigBee specification.

Since an end device may be asleep when a message is sent to it, the message must be buffered in another device, referred to as the end device's parent. In R308 each parent can support 16 (ETRX2) or 30 (ETRX357) children, but this can vary with other firmware versions. An end device must send a data request or poll message to its parent, which will then download all the pending messages that it has stored, though note that very old messages will have either timed out or been overwritten so polling must not be too infrequent. The polling mechanism results in a delay of the message, so there is a trade-off between latency and power consumption. There is no delay when an end device transmits a message, though.

Functionality text (S3B-S3C)

The text in these registers is usually used as a parameter for one of the built-in functions. Its main uses are as a text message to send to the sink, the target address when cloning digital inputs, and a string to be interpreted as an AT command.

Supply voltage (S3D)

The supply voltage in mV.

7.9 Advanced settings (S3E-S35)

Predefined multicast table entries (S3E, S3F)

Multicast table entries 00 and 01. The use of these registers partly avoids the need to rewrite the multicast table whenever the device is reset.

Message endpoint settings (S40, S41)

When sending a message to a node that uses an endpoint other than 01, it is necessary to define the destination endpoint in S40. Other devices may send back their responses to the source endpoint defined in S40. S41 specifies the boot-up value of S40.

Message cluster ID settings (S42, S43)

Messages sent as a result of an AT+UCAST, AT+BCAST, AT+SCAST or AT+MCAST command normally use the Telegesis manufacturer specific profile with message cluster 0x0002. It is possible to change the message cluster ID in order to communicate with devices that use a different profile.

Message profile ID (S44, S45)

When sending messages to a device that interprets them in the context of a different profile ID, the profile and cluster ID must be attached to the messages as specified in S42 and S44. The use of profile ID 0x0000 in conjunction with destination endpoint 00 (in S40) enables you to send ZigBee Device Object commands.

Counter register (S46)

S46 is a 32-bit counter. It can be set to any value with the ATS or ATREMS command, and modified by built-in functions 0300-0302. It can be transmitted to the sink by function 0130.

Power descriptor (S47)

S47 contains the value that the device will return in response to a query of its Node Power Descriptor. This is defined in section 2.3.2.4 of the ZigBee specification [1] and describes the device's power modes and power sources. It cannot be set by the firmware automatically as it relates to the power supply, so the user must enter the appropriate value if desired, although it is not a property that is often used. The fields of the Node Power Descriptor are sent low-order first, so the characters in S47 need to be interpreted in reverse order in relation to the ZigBee specification. Thus the default value C110 is:

0: Current Power Mode

1: Available Power Sources: Constant (mains) power

1: Current Power Source: Constant (mains) power

C: Current Power Source Level: 100% charge

Endpoint 2 Profile ID, Device ID and Device Version (S48-S4A)

7.9.7 S48 defines the profile ID of endpoint 2, S49 defines the device ID of endpoint 2, and S4A defines the device version of endpoint 2, which are sent in response to various ZigBee Device Object queries from other devices when endpoint 2 is activated in S0A.

Endpoint 2 input and output cluster lists (S4B, S4C)

7.9.8 The input and output cluster lists for endpoint 2 can be defined in order that another device can detect the local device when it performs a Match Device request for the purposes of service discovery. Messages received at endpoint 2 can only be passed to the serial port for interpretation by a host processor.

Mobile End Device poll timeout (S4D)

7.9.9 S4D defines the time that a parent device will wait after a poll (Data Request) before erasing a Mobile End Device from its child table. The default value of five seconds means that an MED that has moved out of radio range will be removed from the table quite quickly, making room for another entry in the table. MEDs must therefore poll at least this often in order to maintain the network connections. Note: this register is defined in the parent, not the child.

End Device poll timeout (S4E)

7.9.10 Similarly to S4D, S4E defines the time that a parent device will wait after a poll before erasing a Sleepy End Device from its child table. The default is five minutes, but it is not necessary to explicitly send a poll this often because Timer/Counter 2 normally sends one every minute to check that the end device still has contact with its parent. Note: this register is defined in the parent, not the child.

MAC timeout (S4F)

The length of time in milliseconds during which a parent will store a message that is addressed to a child device. If the child does not poll within that period the message is erased.

Also, this is the time after which a NACK will be reported when the device sends a unicast but does not receive an acknowledgement.

8 Built-in functions

8.1 Introduction

In order that operations that are required frequently can be performed without the need for AT commands, the R3xx firmware has a wide range of predefined functions that can be executed in response to an external interrupt or an internal software timer. The functions are listed in section 5

of the AT command manual [3] and only the most common ones will be described here, so it is recommended that you consult an up-to-date command manual to see if there is a function that meets your needs. By using these functions it is often possible to implement a simple device such as a switch or sensor without the need for a host processor.

8.2 Triggering a function

Each function has a 16-bit code which must be written into an S-register, and the choice of register determines what triggers the event. There are six ways to implement each function, as described below. It is not possible to trigger an event when an analogue input passes through a certain value as this would require continuous operation of an ADC, which is not consistent with a low-power device.

1. An external interrupt (IRQ0-3)

Write the function code into one or more of registers S23-26. Each of these registers is associated with an input pin (PA0, PA1, PB0, PB6 for the ETRX357; I/O0, I/O1, I/O10, I/O11 for the ETRX2). Register S11 determines whether a rising or falling edge on each of these pins generates an IRQ.

Examples:

S23=0001 when triggered by a transition on PA0 the module goes into power mode 0

S24=0017 when triggered by a transition on PA1 the module permits joining for 60 seconds

2. On boot-up

Write the function code into S27, when it will be executed when the device is power up or reset. Note that at this stage it may not be part of a network.

Example:

S27=0047 flash an LED connected to PA7 for 250ms on power-up

3. On joining a network

Write the function code into S28, when it will be executed when the device joins a network. Also, when the device is reset it restores its network state and so executes the code in S28.

Example:

S28=021E if the device is a sink, broadcast its address

4. After a delay

When a function is to be executed after a delay or at regular intervals it must be written into one of the timer/counter registers S29-38. The timer/counters are numbered 0 to 7 and each consists of a pair of registers: the first determines the time in 250ms intervals, and the second holds the function code. Some of the timer/counters are preconfigured and you can refer to their contents as examples, but none of them are dedicated to any particular purpose and the preconfigured ones can be modified or overwritten.

When the function code is used as given in the AT command manual [3] it is executed once after the timer delay. The timer starts when the module is reset or when a value is written into either of the two registers (note: this happens even if the new value is the same as the old one).

To cancel a timer/counter, write 0000 into either register.

Example:

S30=0004 after 1 second
S31=0003 go to power mode 2

5. At a regular time-interval

Write a time interval and a function code into a timer/counter exactly as for a time delay, but set the MSB of the code to 1. The function is then executed after the initial delay and at the same interval thereafter. For instance, code 0014 becomes 8014 and 2100 becomes A100. All the timers can be synchronised by resetting the module.

The timers are software functions, not hardware blocks, so hardly any power is saved by disabling unused functions.

Example:

S29=000C every 3 seconds
S2A=8010 if an end device, poll the parent

6. Triggered by another function

Three of the functions (24xx, 25xx, 26xx) can start, stop or toggle other timer/counters. In this way you can have nested timers or trigger more than one event from the same interrupt. The final byte of these functions is a bit mask where bit n corresponds to timer n, and determines which timers are controlled.

Examples:

ATS31=0028 every 10 seconds
ATS32=A520 toggle timer/counter 5 (S33 & S34)
ATS33=0004 every second
ATS34=8037 toggle PA7 (turn an LED on and off)

ATS23=2430 on IRQ0, start timers 4 and 5 (S31-34)
ATS31=0001 after 250ms
ATS32=0047 flash LED connected to PA7
ATS33=0001 after 250ms
ATS34=0108 send the contents of S3B to the network sink node

It is recommended that the timer which holds the function 24xx, 25xx or 26xx has a lower number than the timers it controls, eg timer 4 controls timer 5 in the example above.

8.3 Example functions

Introduction

As mentioned above, there are too many functions to describe them all in detail, and their effects are described in the AT command manual [3], so only those that are most-frequently used are listed here.

8.3. 0010 – If I am an end device Poll Parent for data

8.3.2 Sleepy End Devices and Mobile End Devices only receive messages from their parent when they send a data request (poll), because only then does the parent know that the child is awake. Frequent polling gives a shorter maximum propagation delay but raises the average power consumption of an SED/MED; a long interval between polls lowers the power consumption but may result in messages timing out and being deleted from the parent's memory before they are forwarded. Further side-effects are that the SED can be deleted from the parent's child table which requires it to rejoin the network, or even that the SED leaves the network. You should be familiar with the timers that control leaving the network and with the effects of timeout registers S4D-4F if you want to use slow or manual polling.

0014 – Check for other devices on the network

8.3.3 Function 0014 is in a preconfigured timer, and makes the device leave the network after a few minutes if it cannot find any neighbours (ie its neighbour table has no entries or as an SED it cannot detect a parent). Functions 0012 and 0013 are similar but result in slightly different operations.

This function can be disabled if it is intended that the device will remain in isolation for long periods and only come into contact with its network intermittently.

8.3.4

0015 – Scan and join a network

8.3.5 Function 0015 makes a device that is not in a network try to find and join one; it has the same effect as the AT+JN command and is preconfigured in one of the timers. It can be disabled if the user prefers to keep control over the joining process.

0017 – Allow joining via the local node for 60 Seconds

8.3.6 When bit 0 of S0A is set in a router and it receives a Beacon Request command, it sends a beacon with the field "Permit association: false". The device that is searching for a network therefore cannot request permission to join via this node, although it may send an Association Request command to another device in the network that does permit joining. When function 0017 is triggered the device behaves for 60 seconds as though bit 0 of S0A was cleared to zero though the contents of S0A are not changed. Function 0017 only affects the local device.

0018 – Copy local Inputs to Remote outputs

Function 0018 creates a remote switching device. Each time it is triggered it reads the local input buffer S1A, and if it has changed it copies its contents to the output buffer S18 of another device. The address of the remote device must be written into S3B by the user.

This function uses the feature that any I/O pin can be configured as an input or output, so that PA7, for example, can be an input on one device and an output on another device. Using function 0018 the state of an input can therefore be copied to a remote output.

To use function 0018, either implement it in a timer and tolerate the lag from input to output, or connect the switch input(s) to an input that can generate an IRQ.

001E – Disassociate from the PAN if no coordinator or sink has been heard from

8.3.7 Functions 0012, 0013 and 0014 check for the presence of a neighbour device, which may be a router or an end device. However, the situation can arise that the coordinator has been removed from the network by accident or as a result of an equipment failure, which can leave a stable set of devices with no means of access by which they can be controlled. Since no new nodes can join the network in the absence of a coordinator, it can be very difficult to restructure the network. Function 001E listens for regular network maintenance messages sent once a minute by a coordinator, and makes the device leave the network if no such message is received. 001E should therefore only be executed every few minutes or the network may become unstable. It only affects routers, but function 0014 makes SEDs and MEDs leave the network if no parent router is available.

003x – ETRX2: Toggle I/Ox. ETRX357: Toggle PA0-7 (x=0-7) or PB0-7 (x=8-F)

8.3.8 Each time the function is executed the level of a single output pin is toggled, eg 0037 toggles I/O7 or PA7.

8.3.9 004x – Flash I/Ox (pull low) for 250ms

A single output pin is pulled low for 250ms then set high, typically to flash an LED.

8.3.10 005x, 006x – set an output high or low

8.3.11 Using the same notation as function 003x, a single output is set low (005x) or high (006x).

0108, 0109 – send a text message to the sink

8.3.12 The contents of register S3B (0108) or S3C (0109) go to the sink, in the same manner as the AT+SCAST command.

0110 – Send ADC readings

A message is sent to the sink containing:

- The lower 16 I/O values
- The readings of the two ADCs A/D1 and A/D2 or ADC0 and ADC1
- An 8-bit message counter which increments each time
- Vcc

The sink presents the data as the text string:

SDATA:[<EUI64>,<ioread>,<A/D1>,<A/D2>,<sequenceNo>,<Vcc>

If bit 8 of S10 is set and the sender does not have a sink address in entry 05 of its address table, the first message will fail then the sender will send a request for the sink address.

Use function 0130 if you want to use more than the first two ADCs.

0111 – Send ADC readings

The same as function 0110, but output I/O3 or PA3 is briefly set high then goes to a high-impedance state. This allows the charging of a capacitor whose voltage can slowly decay and give a longer time delay than is possible with the R3xx timers.

8.3.13

0112, 0113 – Location tracking

Function 0112 is typically run periodically in a device that moves around within a network. It broadcasts a message which can be picked up by nearby routers, which measure the received signal strength and forward this reading to the sink. At the sink a series of prompts is displayed in the form

TRACK:<EUI64 R>,<EUI64 S>,<RSSI>, <i/o read>,<AD1>,<AD2>,<Vcc>,<S46>

The items of data are:

EUI64 R	The address of the fixed node that received the tracking signal
EUI64 S	The address of the moving node that sent the tracking signal
RSSI	The signal strength in dBm
i/o read	The state of the input pins of the moving node
AD1,AD2	ADC readings from the moving node
Vcc	Supply voltage of the moving node
S46	Count register of the moving node

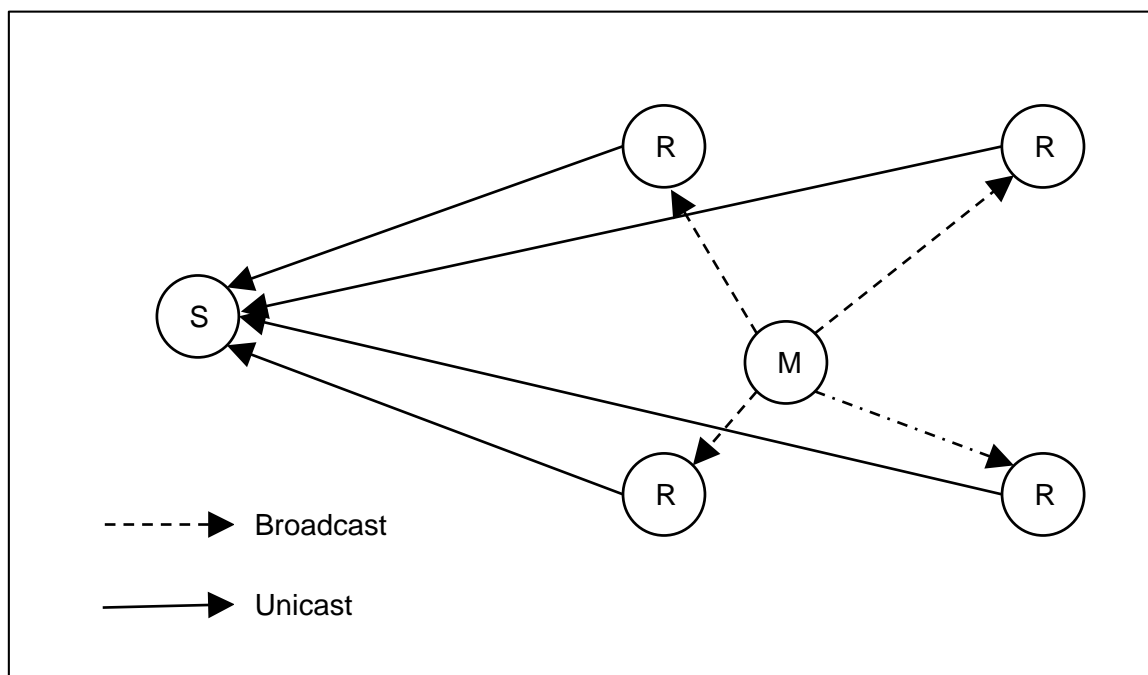


Figure 13. Tracking message from moving node M via routers R to sink S

There is no need to set anything in the routers apart from defining one device as a sink.

Computing a device's location from the signal strengths is a complex topic that requires knowledge of the nature of the propagation path between the devices, and Telegesis does not offer any algorithms for this.

As with function 0111, 0113 can charge a capacitor from I/O3 or PA3.

0114, 0115 – Location tracking

Function 0114 does not include ADC readings which reduces the power consumption of the moving device. Again, function 0115 can charge a capacitor. The prompt at the sink is:

8.3.15
 TRACK2:<EUI64 R>,<EUI64 S>,<RSSI>, <I/O read>,<S46>

The items of data are otherwise the same as for function 0112.

0116, 0117 – Location tracking

8.3.16 Functions 0116 and 0117 add a further refinement to the tracking messages by using a threshold level to filter them. In this case it is necessary to set a value in register S3B in all the routers with the format snnxxx, for example -50abc. The first three characters define an RSSI level; if the signal from the moving device is stronger than this level a message is sent to the sink which appears as a TRACK prompt. If the signal is below the threshold no action occurs. The last three characters are used by function 0118.

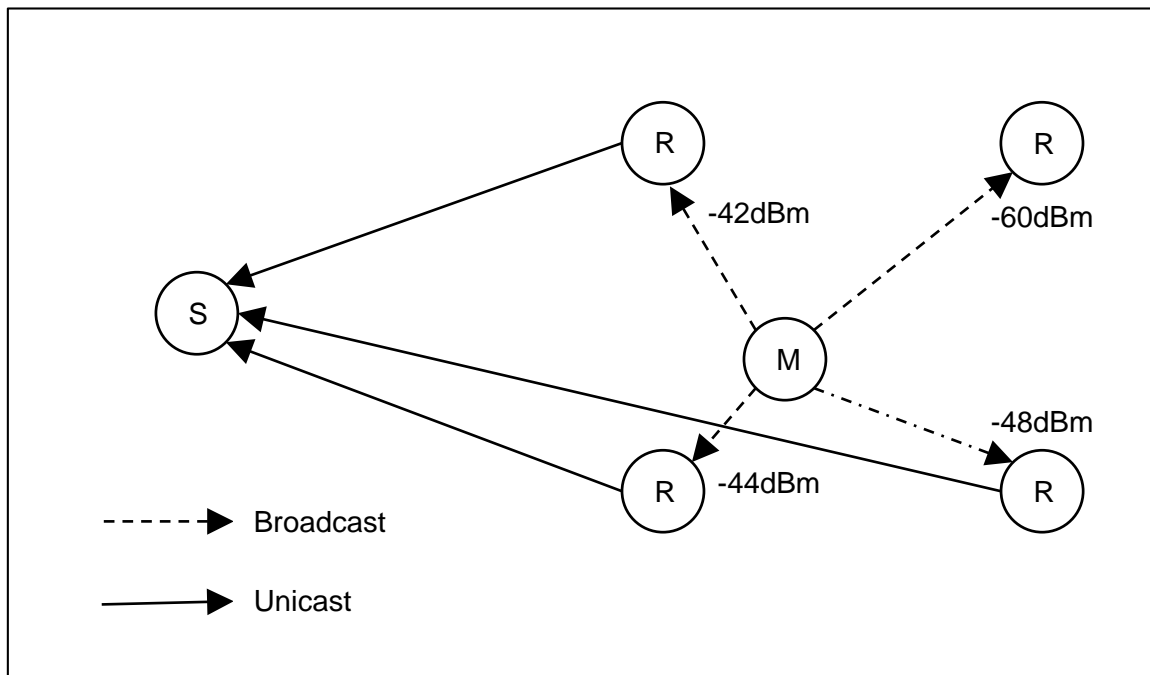


Figure 14. Tracking message with -50dBm threshold

A typical use of this function would be a proximity detector, where the moving device only triggers a tracking message when it approaches one fixed node.

Again, 0117 can charge a capacitor from I/O3 or PA3.

0118, 0119 – Location tracking

Function 0118 combines the features of 0114 and 0116: a threshold is set in register S3B of all the routers in the form xxxsnn, using the last three-letter group, the moving device does not include its ADC readings and the sink shows a TRACK2 prompt. 0119 can also charge a capacitor.

8.3. 0130, 0131 – Send ADC readings

Function 0110 was suitable for an ETRX2 with only two ADCs, but function 0130 allows you to use all four ADCs of the ETRX357. Readings are only sent from those ADCs that are enabled in register S15 and the prompt at the sink is

8.3.18

```
FN130:[<EUI64>],<NWK addr>,<ioread>,<sequence no>,<S46>,<ADC0>], [<ADC1>], [<ADC3>], [< ADC3>]
```

The data items are:

- The EUI64 of the sender (if included in the packet header)
- The node address of the sender
- The I/O values
- An 8-bit message counter which increments each time
- The S46 counter register
- The readings of any of ADCs that are enabled

An example with two ADCs is:

```
FN130:000D6F00024CBBE1,15E5,00FCDF33,13,00000000,4B2E,0501
```

8.3.19 Function 0131 is the same as 0130 but can charge a capacitor through I/O3 or PA3.

02xx – broadcast sink address

If the device is a sink (ie if bit 4 of register S10 is set), function 02xx causes it to broadcast its EUI64 over xx hops (maximum 30). The propagation of the sink address message to end devices is not guaranteed in all versions of the R3xx firmware as attempts have been made to reduce the amount of network traffic, and this is especially true if an end device has been recently reset as its parent may not forward the sink address more than once. It is sometimes helpful to set bit 8 of S10 so that a device wishing to send a message to a sink actively seeks one when necessary.

8.3.20

The default register settings implement this function in timer 1 in the form 821E.

0400 – show network status on pin PA7

Typically, this function is used to flash an LED according to the device's network state:

LED on:	device is not in a network
Fast blink:	device is scanning for a network
Slow blink:	device is in a network

When the function is used in a timer, the repetition rate in the associated timer register determines how often the network state is examined, but it has no effect on the blink rate.

Output PA7 is active low, so the LED must be connected in pull-down configuration. PA7 must be set as an output in S16.

0401 – show network status on pin PB7

Exactly the same as function 0400 except it uses PB7.

0402 – show network status on pin PA6

8.3.21

Exactly the same as function 0400 except it uses PA6, which is connected to the LED of an ETRX3USB stick.

8.3.22

2000 – event counter

8.3.23 Function 2000 causes a message to be sent to the sink after a preset number of interrupts. The message contains a transmission counter and the reading of the analogue and digital inputs. The function requires the use of an external interrupt and a counter/timer. Three registers must be set:

Count register:	the first register of a counter/timer pair. It holds the count threshold
Function register:	the second register of a counter/timer pair. It holds the value 2000
External interrupt:	register S23, S24, S25 or S26. It holds the function 24xx where xx defines the number of the counter/timer

Example:

S31 = 0005
S32 = 2000
S23 = 2410

S31/S32 constitute timer/counter 4. The bitmask 0x10 in register S23 is 00010000, so only bit 4 is set and the function will therefore control timer/counter 4. Each time an IRQ on I/O0 or PA0 occurs timer/counter 4 starts, and on the fifth event the sink receives a message such as:

8.3.24 **SDATA:000D6F00000AD97B,0FF5,0226,036F,04,3296**

which is the same format as a message generated by function 0110.

2001 – character counter

When enabling this action the command line is disabled and as soon as a number of bytes in excess of the number N specified in the accompanying timer/counter register is received on the serial port, a SCAST containing these characters is sent to the network's sink.

Notes:

The number of characters sent is N+1

If N=0, every single character is sent to the sink

$N \leq 64$

N must be written as a 4-character hexadecimal number

No AT commands can be entered into the serial port, so this function can only be stopped by overwriting the register that contains code 2001 from another node.

The code must be 2001, not A001. It always repeats every N characters.

2100, 2101 – create your own function

Although the range of built-in functions is very diverse there will always be occasions when there is nothing that is quite right for your application. If you can write an AT command that is suitable then you can create a function that has the same effect. Just remove the “AT” prefix and write the remainder into register S3B, and use function code 2100 in the same way as the other codes. Each time it is triggered the contents of S3B are prefixed with “AT” and sent to the command interpreter.

Function 2101 is the same except that it uses register S3C.

24xx, 25xx, 26xx – control other timers

- 8.3.26
- 24xx:** start timers masked in xx
 - 25xx:** toggle timers masked in xx
 - 26xx:** stop timers masked in xx

The use of these functions is described in section 8.2 (Triggering a function)

3xxx – set output states

Function 3xxx sets or clears the lower 12 bits of the output buffer which will control all the output pins of an ETRX2 but only pins PA0-PB3 of an ETRX357. xxx is a bitmap with the LSB controlling PA0 or I/O0 in the same manner as register S18. When function 3xxx is executed the lower 12 bits of S18 hold the updated value.

4xxx – set I/O pin directions

- 8.3.29
- Function 4xxx is similar to 3xxx but it determines whether each of the lower 12 pins is an input or output. Register S16 holds the updated value.

53xx – Toggle output xx (R309)

Function 53xx toggles a single output pin. For example, 5300 controls PA0, 5308 controls PB0 and 5317 controls PC7.

54xx – Flash output xx (R309)

In similar fashion to functions 004x and 53xx, 54xx flashes output xx low for 250ms.

55xx, 56xx – Set output xx low or high (R309)

Function 55xx sets output xx low, 56xx sets it high.

9 PWM output

9.1 Tone Generation at Pin I/O3 or PB7

Instead of acting like a normal I/O, I/O3 (ETRX2) or PB7 (ETRX357) can drive a PWM signal or a variable-frequency signal, for example to play a tune on a beeper (as with the devboard using “AT+IDENT”) or to control a dimmer. To set up the signal use the following registers:

- S11: set bit F to enable special function pin
- S1B: controls PWM signal frequency (volatile)
- S1C: initial value of S1B (non-volatile)
- S1D: controls PWM signal duty cycle (volatile)
- S1E: initial value of S1D (non-volatile)

9.2 PWM register settings

S1B represents the top value of a 16-bit counter incrementing at 12MHz. Whenever the top value is reached the counter restarts from zero and I/O3 (ETRX2) or PB7 (ETRX357) is set.

S1D represents the compare value of the 16-bit counter described above. When reaching the compare value I/O3 (PB7) is cleared.

To produce a waveform of 800Hz and a 50% mark/space ratio on the PWM output we need to divide the 12MHz clock by 15000d = 0x3A98. This value is the top value of the counter at which the output is set high. In order to achieve a 50% mark/space ratio the compare value needs to be half the top value, namely 7500d = 0x1D4C. These are the default values of S1C and S1E.

The PWM output also serves as a DAC. If, as before, S1B=3A98 and S1D=1D4C, then the mean level is $0.5V_{dd}$. If S1D is halved to 0EA6 then the mean level is $0.25V_{dd}$ though the frequency is unchanged. The exact value and accuracy of V_{dd} depends on the user's power supply.

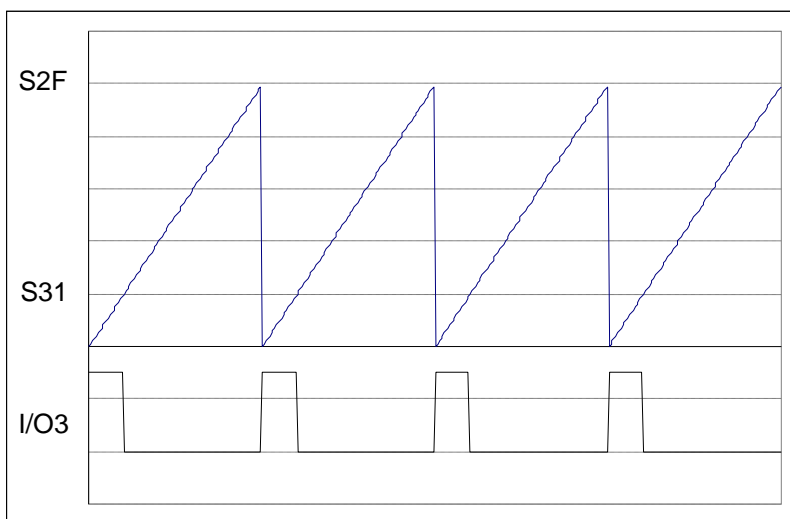


Figure 15. PWM output and compare levels

Note: the use of S1B varies a little between firmware versions. The PWM frequency is either $12\text{MHz}/\{S1B\}$ or $12\text{MHz}/\{S1B+1\}$ though the difference is often not significant.

The range of frequencies achievable with the 16-bit S1B register goes from 6MHz down to 183Hz. When bit C of S0A is set, the clock is prescaled by 256 so the range is from 23.4kHz down to 0.72Hz. If the PWM output is connected to one of the IRQ pins it is possible to trigger functions faster than the 250ms of the built-in timers.

10 Getting Started with Telegesis Terminal

The ETRXn sends and receives commands and data from its host computer through a UART as ASCII strings. You can use your own application software, a tool such as HyperTerminal or PuTTY, or our own Telegesis Terminal which is a free download from our website at <http://www.silabs.com/telegesissoftware>.

Note: Telegesis Terminal is not a GUI that interprets the AT commands and sends instructions to the ETRXn in another format. It does not alter the AT commands in any way – the ETRXn receives them exactly as described here and in the AT Command Manual [3]. Likewise, if you are writing your own terminal application, it should send and receive data as formatted in our manuals. Do not include any of the quotation marks around the commands described here.

In order to get started, plug an ETRXn module on to each development board which forms part of your set-up and connect at least one development board to a PC using the serial or USB cable provided.

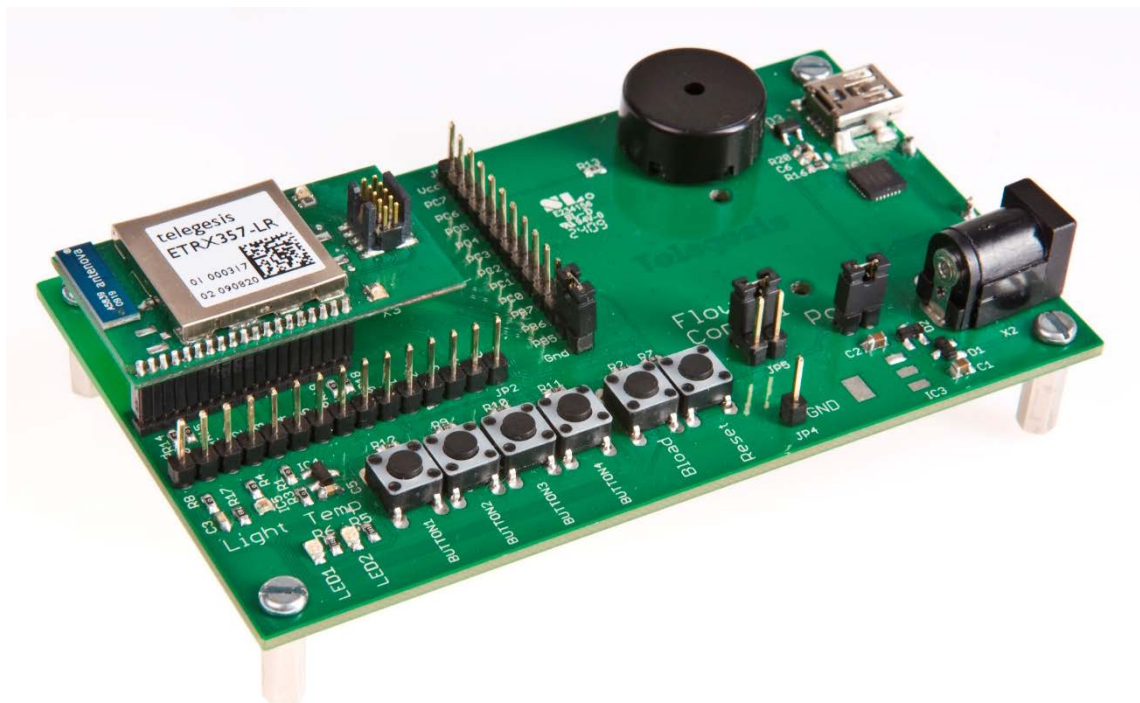


Figure 16. Development board with module mounted

Power up all of the development kit boards and start the Telegesis Terminal application on the PC. You need to know which of your computer's COM ports is connected to the development board, which may not be obvious if you are using a USB-to-serial adaptor, for example. To check, open your PC's Device Manager; there are several ways in Windows, but the simplest is to right-click on

“My Computer” and select “Manage”, then “Device Manager”. In this example there are two USB-to-serial adaptors in COM3 and COM4 and a USB stick in COM5:

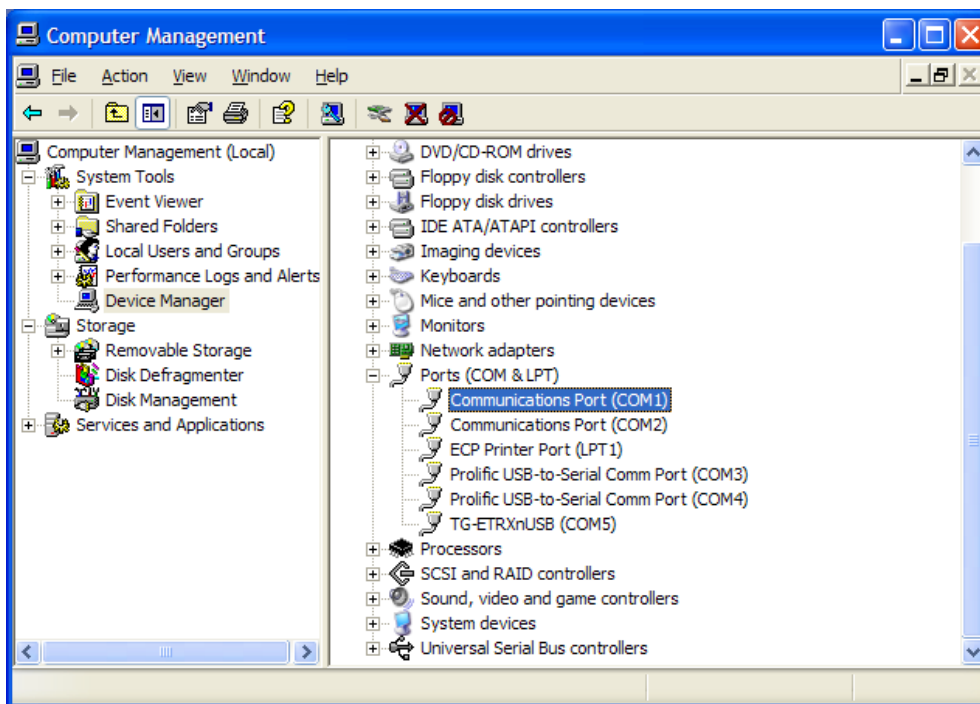


Figure 17. Windows Device Manager

The connection should be set up using the default values as shown in Figure 18. By default the module uses a bit-rate of 19200bps, no parity, 8 data bits, 1 stop bit and no flow control. Optionally XON/XOFF as well as hardware flow control is supported. (See the description of register S12 in the AT command manual [3] for more details). Pressing the “Connect” button establishes the connection to the ETRXn module on the development board.

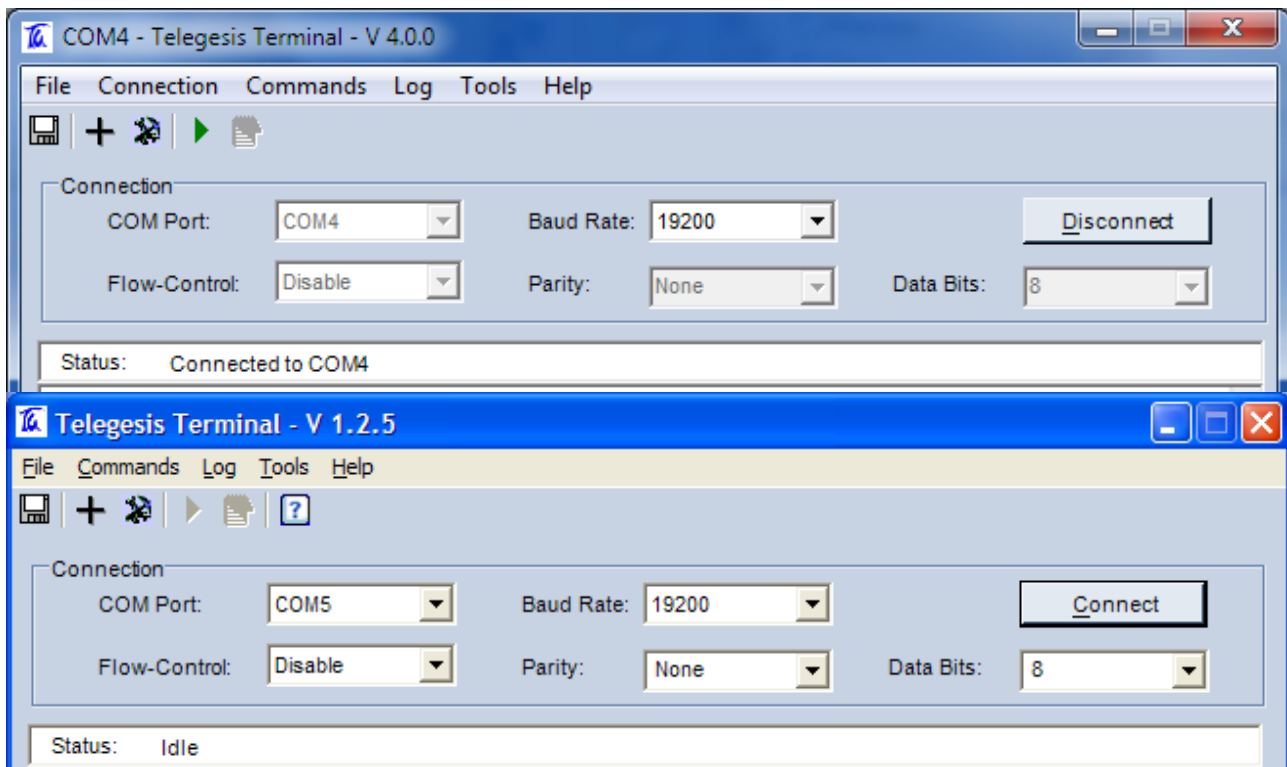


Figure 18. Setup of the Serial Connection

The module will accept commands starting with the “AT” prefix after it has booted up successfully. Booting up can take 1-2 seconds and on completion the module will prompt “OK”.

Entering the “ATI” command into the terminal window, followed by <Enter>, will cause the module to display its manufacturing information. Alternatively the “Info” button in the “Module Control” section can be pressed, causing the “ATI” command to be sent to the local module. After executing a command the module will prompt “OK” or an error code as explained in the AT command dictionary.

In order to communicate with other modules a module must be part of a PAN (Personal Area Network). To find the status of the local module simply issue the “AT+N” command and this will show you the module’s network status. If the module is not part of a PAN (response “+N:none”) it can be instructed to join an existing PAN using the “AT+JN” command, or to start a new PAN for other units to participate in using the “AT+EN” command. In order to exchange messages all units need to be on the same PAN, i.e. have the same channel and the same PAN ID. The response “+N=<devicetype>,<channel>,<power>,<PID>,<EPID>” displays the channel number plus the PAN ID and Extended PAN ID of the current PAN, the type of device (COO, FFD, SED etc) and its RF transmitter power. If you have a module which is part of a separate PAN simply use the “AT+DASSL” command to leave the current PAN and try joining the correct PAN using the “AT+JN” command. Network establishment and maintenance is described in more detail in Section 4.

In order to find other nodes which are part of the same PAN press the “Scan Pan” button which sends the “AT+SN” command to the local module. (Your terminal software may expect a parameter after the command, namely the number of hops to interrogate. Just type a suitable two-digit number e.g. “08”). The module will now transmit a request to all the modules within eight hops asking them to identify themselves; “AT+SN:00” scans the entire network. When modules are found the “Discover

Devices” window will pop up (if not already open) and display a list of all of the modules which have reported in. See Figure 19.

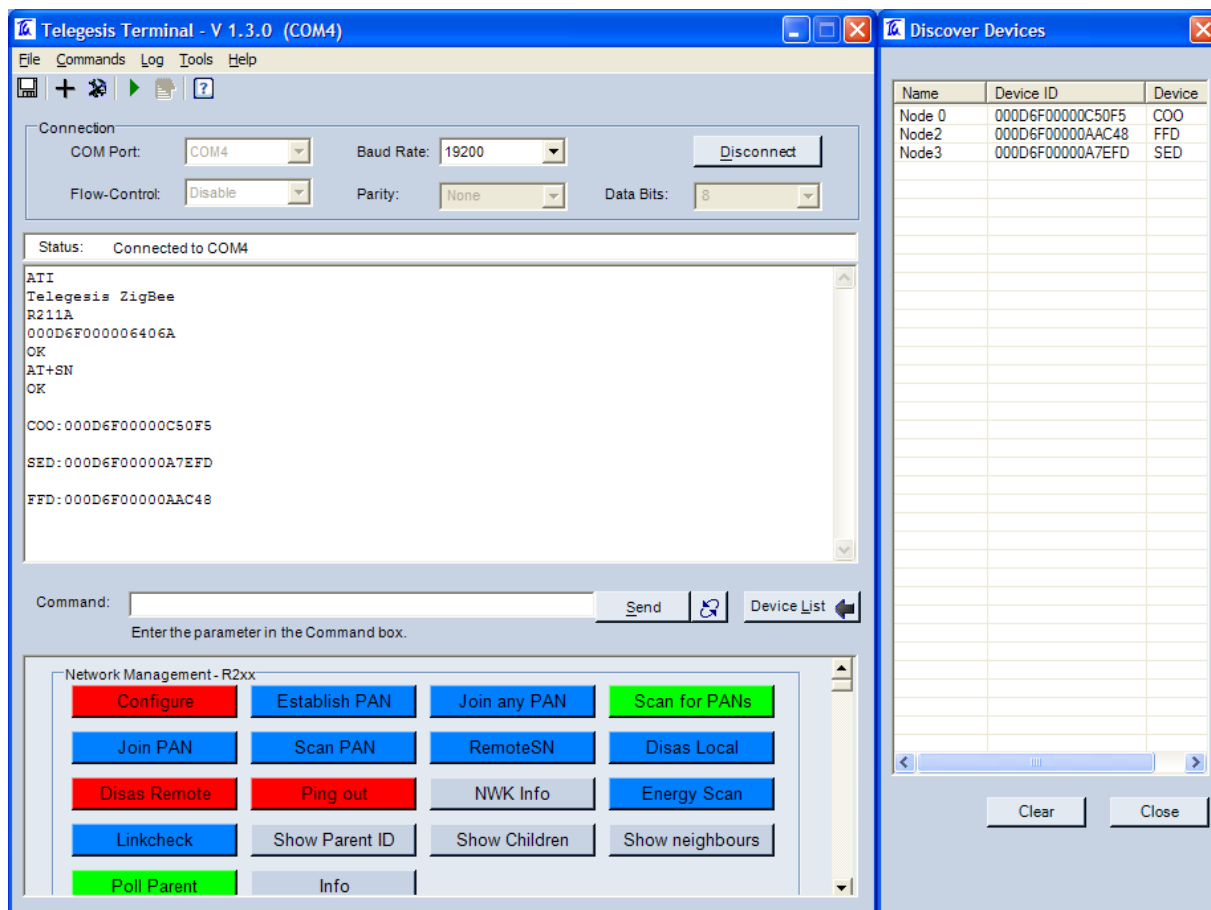


Figure 19. Results of a network search

In this example three additional modules were detected: one ZigBee Coordinator (COO), one ZigBee Router (FFD) and one Sleepy End Device (SED). For an explanation of the device types see Section 2.6.

An alternative way to search the PAN for other modules is to hit the “Configure” button. As well as scanning the PAN as before, it adds extra buttons to control the buzzer and LEDs on each remote module.

Depending on the previous usage of your development kit, units may either immediately communicate with each other on the same PAN, or they may be off-line. If the unit connected to your PC does not report that it has joined a PAN within a few seconds of power-up, a suitable procedure would be:

1. issue the command AT+JN to join any existing PAN or
2. issue the command AT+PANSCAN followed by AT+JPAN:<channel>,<PAN ID> to find and join a specific PAN or
3. Wait for the module to automatically join a PAN as described in section 4.1.

4. If the steps above do not reveal an existing PAN, issue the command AT+EN to initiate a new PAN. Others units searching for a PAN will then attach to it within a few minutes, and be reported with a "NEWNODE:<EUI64>" prompt.

A common problem occurs when the user establishes a PAN and waits for the other devices to join it ... and waits ... and waits. This usually means that the other devices are already in a PAN of their own (set up when they were last used), and as long as they can communicate amongst themselves they will ignore the new network. If you use the AT+PANSCAN command, this will reveal the pre-existing PAN. If it has a coordinator you can join it and then either carry on using it or disassociate the remote nodes and start afresh; if it has no coordinator you cannot join it so each device must be disassociated manually or forced to become orphaned by powering it up in isolation and waiting for it to leave.

In order to find out which of the detected modules is which, press the "Ident Node" button in the "Test Commands" section of the buttons. This command requires an additional parameter, therefore instead of sending a command straight to the local node the command is displayed in the command bar with the text "<Enter Parameter here>" highlighted.

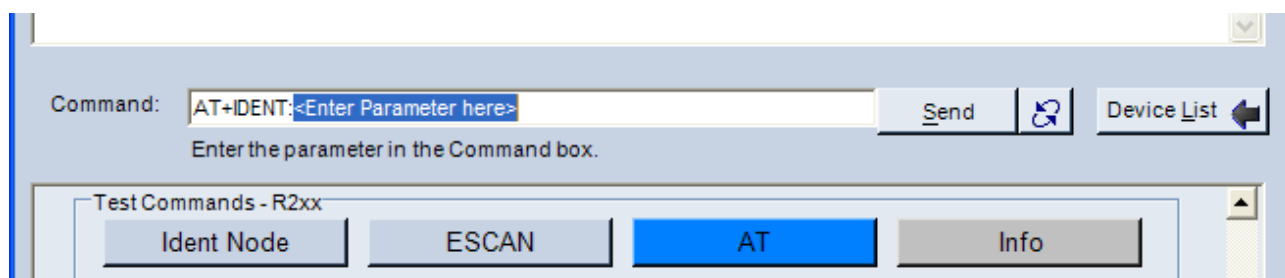


Figure 20. Entering parameters

Now, double click on any entry in the "Discover Devices" window and the serial number of the selected device will be added as a parameter. The command is executed by either pressing "Enter", or by clicking the "Send" button.

When executing this command the local module will send a request to the module with the specified serial number asking it to identify itself. A module can identify itself by playing a tune on a buzzer which is connected to I/O3 or PB7. If you are using an ETRX2 devkit and you do not hear a tune check that the jumper for I/O3 is set so that I/O3 is an output (factory default); on the MCBs the buzzer is directly connected to I/O3.

Having identified which module is which you can now optionally give each module a name using the Telegesis Terminal software. To do this, right click on the entry in the "Discover Devices" window (as shown in Figure 21) and enter the name when prompted to do so. Please note that this name is not written to the module itself, it only represents a temporary name which can be used to simplify the evaluation process using the Telegesis Terminal software.

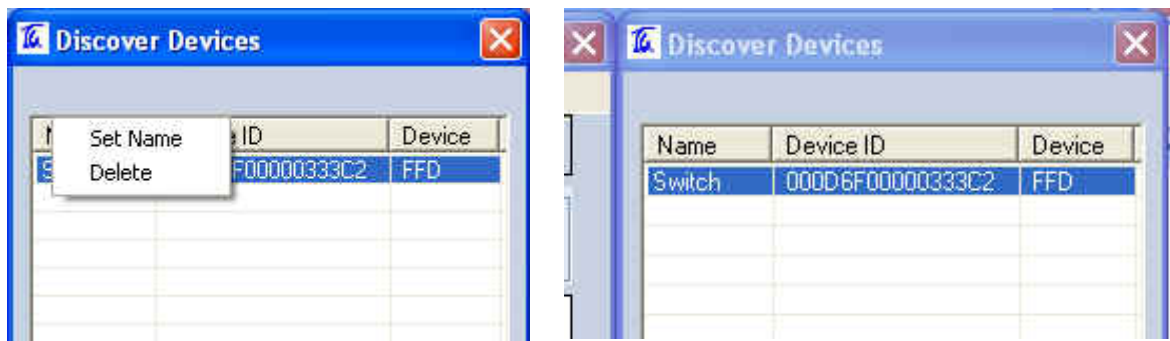


Figure 21: Device Naming

If you have used the “Configure” button, your button window will resemble Figures 22 and 23 and the device table Figure 24.

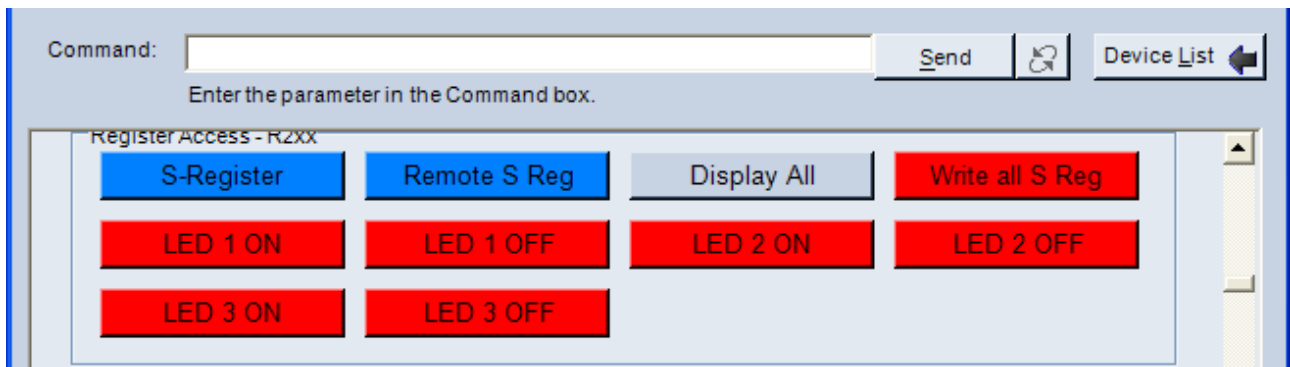


Figure 22. "LED" buttons after a “Configure” operation

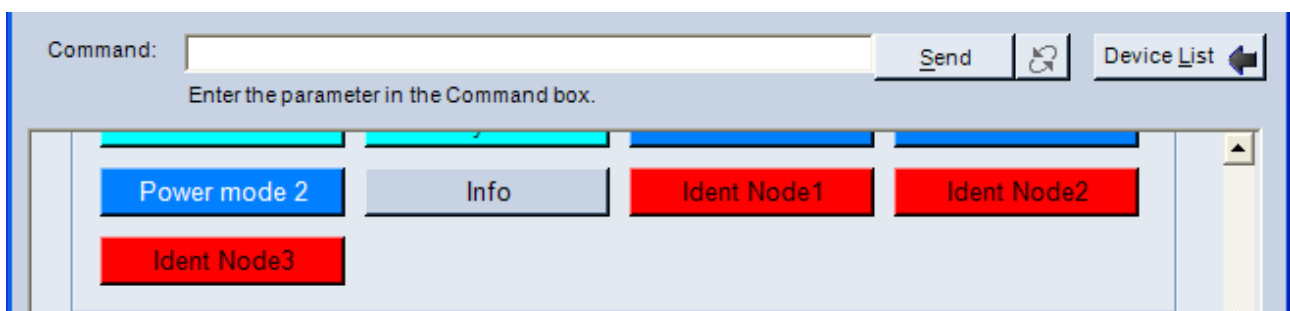
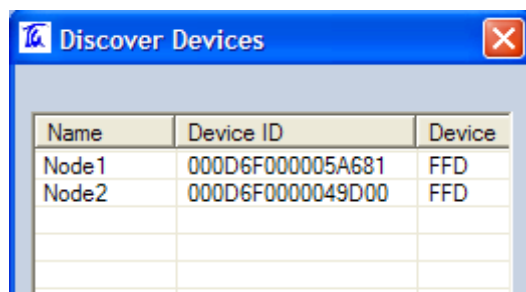


Figure 23. "IDENT" buttons after a “Configure” operation



Name	Device ID	Device
Node1	000D6F000005A681	FFD
Node2	000D6F0000049D00	FFD

Figure 24. Device table after a “Configure” operation

The “Config” button generates an AT+SN command and uses the responses to create new buttons as shown in Figure 22 and Figure 23. These allow you identify the various development kit boards and module carrier boards in your network. If you save the button layout file (or accept the option to save changes when you close Telegesis Terminal) then the same buttons will reappear the next time you start Telegesis Terminal, though the address associated with each button may no longer be valid. The “Config” button does not work properly with the ETRX2 development kit because the output pins connected to the LEDs differ from the ETRX357 version. The LED buttons will therefore be created but only send the right command to an ETRX357. The AT+IDENT command remains the same so those buttons work properly.

After identifying all of the devices in the network we can also send messages to any devices within range. In order to send a broadcast message to all of the devices, use the “AT+BCAST:” command followed by the number of hops you want the broadcast to travel, then the text you wish to broadcast, or simply press the “Broadcast Button” and enter parameters into the Command line. (E.g. “AT+BCAST:01,Hello World”)

It is also possible to send data to an individual node using a Unicast. To send a Unicast press the “Unicast” button and then double click on the entry of the recipient in the “Discover Devices” window and enter the data you wish to transmit preceded by an equal sign. If you do not know the syntax of a command, hold the mouse pointer over the respective button and the tool tip text will display the format of the command as shown in Figure 25.

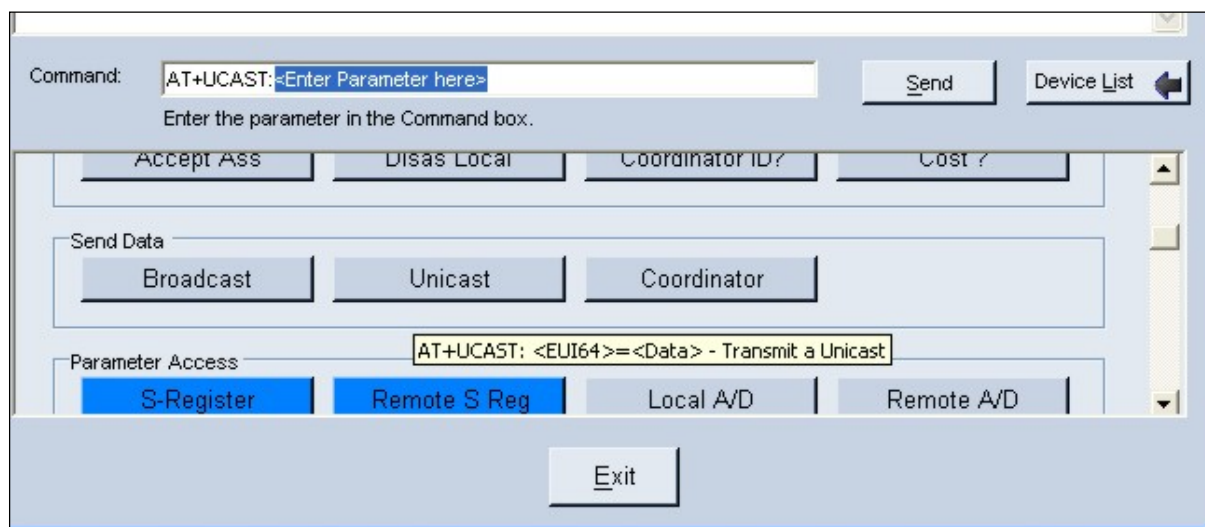


Figure 25. Tool tip texts

So far we have set up all of the hardware required for a meshing network, identified and named the individual modules using the Telegesis Terminal Software and actioned a few simple commands in order to learn the functionality of the AT-Style command line interface. From this point you can experiment with the AT+UCAST and AT+BCAST commands to send text messages, the ATREMS command to monitor and control remote devices, and the use of the built-in functions to create autonomous actions.

11 Reviving An Unresponsive Module

If the module is completely unresponsive, it may be in power mode 3 or the serial port settings may have been altered. Go to the “Download software” page of the Telegesis website and download the [Factory Default Utility software](#). Connect the module’s serial port to your PC and run the Factory Reset Tool; when prompted execute a hard reset by pulling the reset pin to ground or pushing the reset button on a Development kit board. (The screening can is a convenient earthing point.) If the serial port settings have been changed, resetting the contents of register S12 over the air may solve the problem.

12 Application Examples

In this section there is a list of suggested applications using the ETRXn module. This list will never be exhaustive, but should give you an idea about the great flexibility and versatility of the Telegesis ETRXn Wireless Meshing Module.

12.1 A simple timer application

Each timer consists of a pair of registers: the first defines the time interval in 250ms ticks, the second contains the function code to be executed at the appropriate time. The code as given in the table of built-in functions in the AT Command Manual [3] results in a single-shot event at the end of the time interval, but if the MSB of the code is set to 1 the event repeats indefinitely. As an example, look at the default settings of Timer/Counter 0:

S29=0004 4 250ms ticks, so event repeats every 1 second

S2A=8010 Function 0010 “End devices poll parent”, with the MSB set to make it repeat

For instance, to flash an LED connected to pin PA7 on a devkit board:

1. Choose a spare timer such as S30/S31
2. Look up the function code in the AT command manual, in this case 003x
3. Replace ‘x’ with the index of the selected I/O, to give 0037
4. Set the MSB of the function to make it repetitive, to give 8037
5. Set S30=0004 to make the LED toggle once a second, for example
6. Set S31=8037

12.2 More complex timer functions

Functions 24xx, 25xx and 26xx allow one timer to control other timers. xx is the hexadecimal equivalent of a bit-map which defines which timers are to be controlled, eg to repeatedly toggle timers 4 and 6 the bit-map is 01010000 (0x50) and the code is 25xx, so with the MSB set the

complete function code is A550. As an example we can use the LED connected to pin PA7 on an ETRX357 development kit board:

```

ATS31=0014
ATS32=A510
ATS33=0002
ATS34=8037

```

Timer 4: toggle Timer 5 every 5 seconds

Timer 5: flash LED on and off every 500ms

The LED will repeatedly flash for a time then pause. Since the bit-mask allows one timer to control several others, you can use these functions to trigger several different events synchronously or when triggered by a single external interrupt pulse. The most reliable way to use functions 24xx, 25xx and 26xx is to put them in a lower-numbered timer so that they control higher-numbered timers (in order that the controlling function is executed first).

12.3 Simple Temperature Sensor

Let us assume we have a simple temperature sensor on a remote node and a data-gatherer node which sinks the data transmitted by the sensor, possibly talking to one another via several relaying FFDs.

Various sensors could be attached to up to four ADC inputs. Note that these must provide analogue readings in the range 0-1.2V; sensors that provide data in a text format must communicate through the serial port.

- The simplest approach would be to poll the sensor every now and then from the data-gatherer using the "ATREMS:<EUI64>,1F?" command (or similar). The sensing node could be a sleepy or mobile sleepy end device as well as a router. The advantage of the latter would be that every sensing node is an FFD and can therefore relay messages forming a reliant and failsafe meshed network. A host controller would have to initiate the polling operation on the data sink.
- An alternative is to use one of the ETRXn's built-in timers and select a suitable function from the list of built in functionality which is further explained in section 8, such as 0110 or 0130 (send I/O and ADC readings to sink). This will make the sensor send its temperature information to the sink periodically. As in the previous option the sensor could be an end device in power mode 2, thus being battery operated. See the notes on timers below.

12.4 External interrupts and complex commands

A digital transition applied to certain I/O pins can trigger an interrupt. The ETRX2 and ETRX357 can each sense four interrupt inputs:

	ETRX2	ETRX357	Register
IRQ0	I/O0	PA0	S23
IRQ1	I/O1	PA1	S24
IRQ2	I/O10	PB0	S25
IRQ3	I/O11	PB6	S26

Register S11 determines whether each interrupt is triggered on a rising or falling edge (or both, or neither). For example, the default function code for IRQ0 is S23=0001; together with the default of

S11=0005 this makes a falling edge on I/O0 or PA0 wake up a sleepy device and return it to power mode 0.

The built-in function codes provide a very flexible set of capabilities, but there will always be the need for an operation which is not covered. If you can represent this operation as a standard AT command, then you can create your own function by using codes 2100 or 2101. Write the AT command as a text string into S3B or S3C (omitting the initial "AT"), then set up a timer or interrupt in the normal way. At the appropriate time the AT command is sent to the command line interpreter and executed. Two examples:

```
S31=0010  
S32=A101  
S3C=+UCAST:000D6F0000D5599C=Hello
```

```
S24=2100
```

```
S3B=&F
```

Every 4 seconds the device will send a unicast to node 000D6F0000D5599C. When a transition occurs on I/O1 or PA1, "AT&F" is executed so all the registers are returned to their factory default state and the device will leave the network.

12.5 Switch Application

For a classical switch application like a light switch, it is quite obvious that polling the switch is not the right way forward as this would lead to a massive increase of network traffic.

For this scenario the following methods can be considered:

- By setting register S16 S24 to 0100 0130 and enabling the appropriate interrupt in S2E S11 every falling edge on PA1 or I/O1 will cause the switch unit to transmit the status of its I/O and both any enabled A/D readings to the sink, which can control the light.
- Alternatively, the falling edge on I/O1 can wake up the switch module from power mode 3. Being a sleepy end device this could lead to many years of battery operated lifetime depending on how often the switch has been triggered and also the shelf life of the battery.
- Starting with firmware version R211, built-in function 0019 0018 allows a device to send the state of its logic inputs to another node whenever an input changes. The receiving node will set its outputs to match the incoming pattern, so that switch inputs on one node can be mirrored to load outputs on another node. The inputs could be polled at regular intervals by associating function 0018 (or more specifically 8018) with a timer, or by triggering it by an external interrupt.
- Starting with firmware version R211, built-in function 001B allows a device to read the inputs of another device and set its outputs to correspond with them. However, this is not as good as function 0019 as the local node must continually send radio messages to interrogate the remote node, and if a fast response is needed there will be a lot of radio traffic that could degrade the network. However, the corresponding one-shot function 001A could be used in conjunction with function 0019 to send switching commands to another device and receive feedback.
- Using simple digital logic blocks this application can be extended to serve more than just a single switch and also serve more than one actuator to present feedback. The switch application shows four modes of operation which are both versatile and cover many possible applications without the need for a host microcontroller. The switch application is a typical

application which shows how the module can operate autonomously without the need for a host controller.

If you are using the digital inputs as sensors, take care that they are not assigned to IRQ functions or else changes of level might cause unexpected side-effects.

ATS31=0002
ATS32=8037 Timer 4: flash LED on and off every 500ms
ATS33=0014
ATS34=A510 Timer 5: toggle Timer 4 every 5 seconds

12.6 Serial Port Replacement

Where you need to control a number of devices using RS232 or RS485 (e.g. in an industrial automation or metering scenario) and you do not want to wire up your production line all you need to do is connect the serial port of a module to your end device. If you use the AT+DMODE command the coordinator can open a channel which the end device will accept by default and send the required data. After receiving the required feedback the coordinator can close the channel again and talk to the next unit.

Any prompts on the remote side can be disabled using the extended function registers S0E and S0F, so the remote end would see nothing except a transparent link.

Mesh networking is very reliable for automation in industrial environments as reliability increases with each device added into the redundant mesh. This overcomes one of the great shortcomings of point-to-point radio in automation.

Telegesis have also assembled the ETRX2 and ETRX357 modules into units with interfaces to USB (ETRX2USB, ETRX3USB) and Ethernet (Gateway), which are very suitable for connecting to laptop computers etc as central control and data-gathering nodes without the need for the customer to design their own device.

12.7 Using a Host Microcontroller

Using an inexpensive host microcontroller with an on-chip UART you can greatly extend the possible applications of the ETRXn wireless meshing modules. The input buffer of the ETRXn module is 128 bytes, and as no command is longer than that, no buffer overflows can be expected during normal program operation. For this to work it is important to wait for feedback after the execution of a command, i.e. do not issue a new command until you have received an "OK" or "ERROR:xx" prompt.

When using a channel there are no limitations to the payload, so a buffer overflow can occur when sending too much data at once or where the recipient is temporarily out of reach. To avoid a buffer overflow XON/XOFF or hardware handshaking should be used.

12.8 Custom Functionality

If you find that you have an application which you cannot solve with the predefined functionality supplied, and you do not want to use a host microcontroller for every device, please contact us about the extension of the functionality required to fulfil your needs.

Telegesis is also willing to discuss custom firmware development for the ETRXn range of products.

12.9 Developing Your Own Firmware

Feel free to develop your own firmware to go on to the ETRXn and use the modules as hardware only. In order to do that you need to contact Silicon Labs to get access to their stack and development toolchain.

13 How to's

13.1 Check firmware version

Use the command "ATI"

13.2 Check network status

Use the command "AT+N"

13.3 Start a new network

Use the command "AT+EN"

13.4 Join a network

Use the command "AT+JN" or "AT+JPAN", or wait for a built-in timer to trigger the operation

13.5 Change device from FFD to SED

Use the command "ATS0AE=1;password" and join a network, after first leaving if necessary

13.6 Get received signal strength

Use the commands "ATS10C=1" and "AT+SN:01"

13.7 Correct a wrong baud rate setting

Use the command "ATREMS:<address>,12=0500" over the air from another device, or use the Factory Default tool that you can download from <http://www.silabs.com/telegesissoftware> to return to 19200 baud.

13.8 Set power mode

- Use the command "ATS39=0002" to go to power mode 2
- Use the command "ATREMS:<address>,39=0000" over the air from another device to wake a device up
- Use an external interrupt; IRQ0 defaults to waking a device up

13.9 Assign a different network address

Not possible

13.10 Assign a different EUI64

Not possible unless you have an InSight Adaptor and a pool of EUI64 numbers to assign

13.11 Make a secure network

See section 6 and Appendix A of the latest AT command manual [3]

13.12 Use another ZigBee profile

1. Set the endpoint register S40
2. Set the cluster ID register S42
3. Set the profile register S44
4. Display the desired endpoints by setting register S0F

Alternatively use AT+SENDUCAST or a similar command.

Refer to the Telegesis Application Note on Interoperability [1]

13.13 Use the ZigBee Smart Energy profile

Not possible. The R3xx firmware does not support the Certificate Based Key Exchange.

14 Glossary

AES	Advanced Encryption Standard
COO	Coordinator
EUI64	Unique 64 Bit serial number of a ZigBee device
FFD	Full function device
LQI	Link Quality Indication
MCU	Microcontroller unit
MED	Mobile end device
PAN	Personal area network
RFD	Reduced function device
RSSI	Received signal strength indication
SED	Sleepy end device
ZC	ZigBee coordinator
ZED	ZigBee End Device
ZR	ZigBee Router

15 Module Pinouts

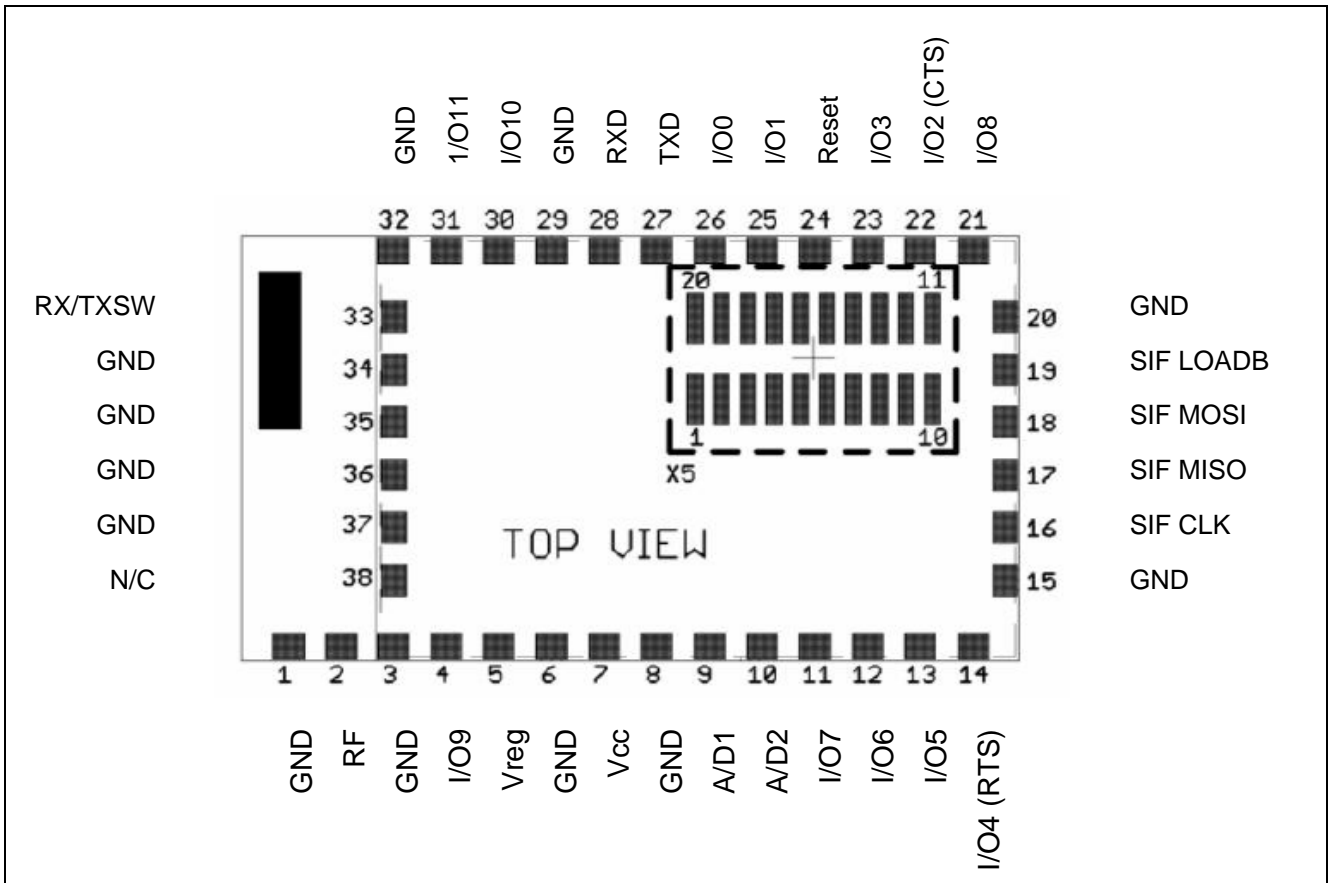


Table 19. ETRX2 pinout

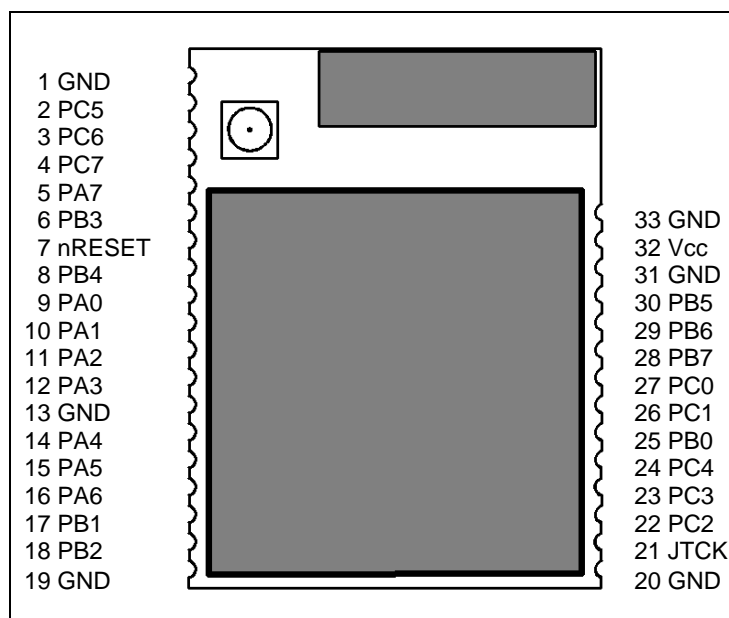


Table 20. ETRX357 pinout

16 References

- [1] The Institute of Electrical and Electronics Engineers, Inc, IEEE Std. 802.15.4 Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs), 2003.
- [2] Telegesis, "R309 AT Command Manual (or version for firmware in use)".
- [3] Telegesis, "Application Note on Interoperability".

17 Appendix - Starting a network with Telegesis Terminal

This is quick guide to starting a network using a few Telegesis ETRX2 or ETRX357 modules connected to the serial port of a PC or laptop and our Telegesis Terminal application program (downloadable from our website). For the sake of brevity we do not describe all the possible situations that can arise, but we have included a few redundant steps in this guide to smooth your path; for example, the AT+N and AT+PANSCAN commands are not always necessary, but they help you to discover how your devices are set up.

1. Have the appropriate manuals available for reference, for example the ETRX2 or ETRX357 Development Kit Product Manual and the R3xx AT Command Manual [3]. The precise manuals that you will need may vary according to your devices and firmware.
2. Install Telegesis Terminal on your computer following the advice in the Development Kit Product Manual
3. Start Telegesis Terminal and establish serial connection with one or more modules. You can have more than one Telegesis Terminal open at the same time. Verify that you are correctly connected with the AT! command
4. Select the device you want to become the network coordinator. Check whether it is already in a PAN with the command AT+N. If it is in a PAN, leave the network with the command AT+DASSL
5. Establish a new PAN with the command AT+EN
6. Select another node to join the PAN. If you want it to be a Sleepy End Device configure it with the command ATS0AE=1;password (for R3xx firmware)
7. Check whether it is already in a PAN with the command AT+N. If it is in a PAN, leave the network with the command AT+DASSL
8. (Optionally) find all the available networks with the command AT+PANSCAN
9. Join the PAN with either the command AT+JN or AT+JPAN:<CC>,<PPPP> where <CC> is the radio channel and <PPPP> the PAN ID as reported by AT+PANSCAN
10. Verify that the device has joined the network with the command AT+N
11. Scan the network and create new Telegesis Terminal buttons by clicking on "Configure" Find the buttons labelled "Ident Node1" etc and check your network connections by clicking on each one.
12. Now you can try experimenting with broadcast and unicast messages, and reading and writing the S-registers to test the buttons and LEDs on your development kit boards.

Info

NWK Info

Disas Local

Establish PAN

NWK Info

Disas Local

Scan for PANs

Join any PAN

Join PAN

NWK Info

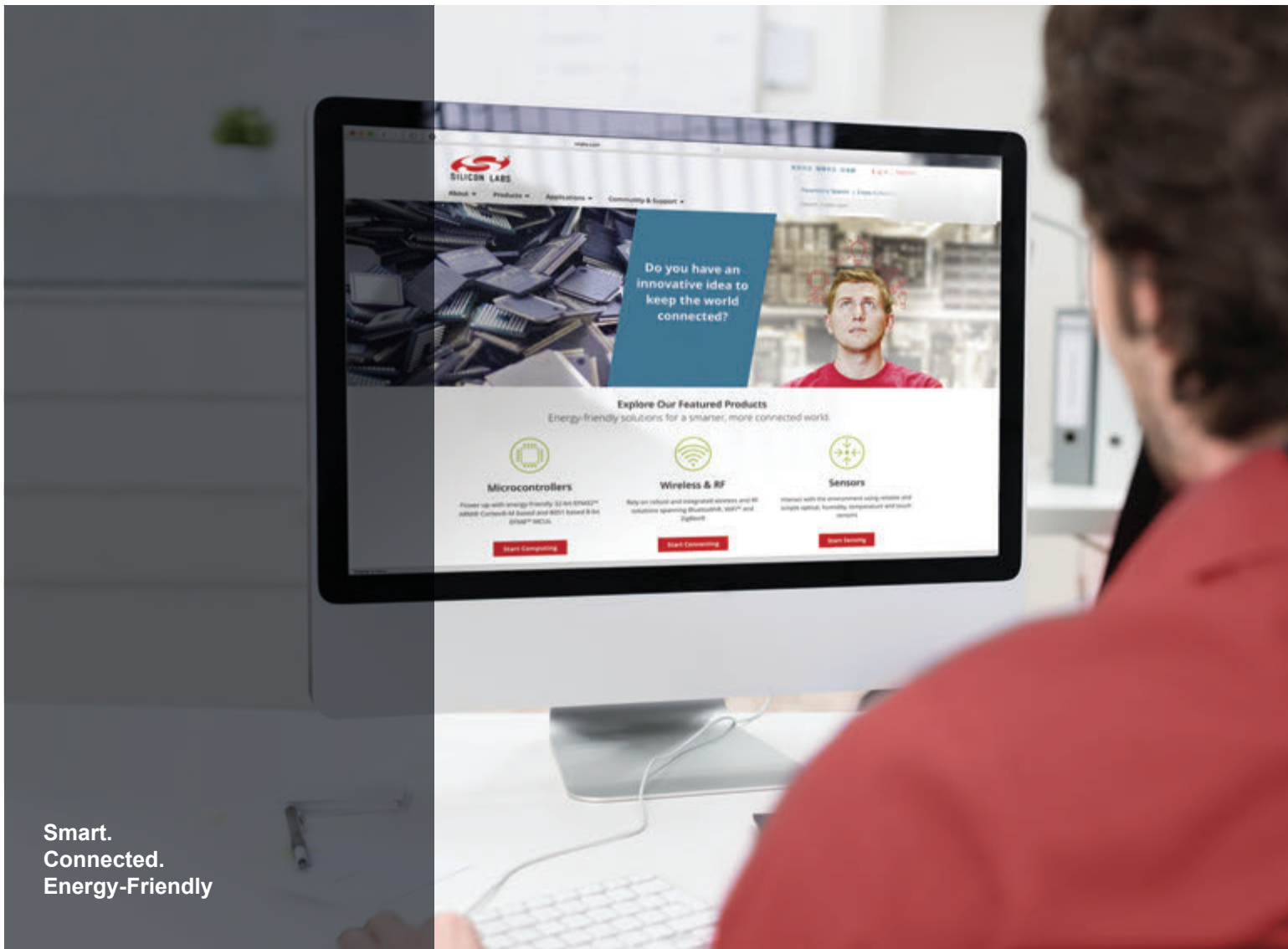
Configure

Ident Node1

Some messages you may see

(the actual data will depend on your local network, of course)

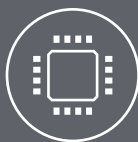
+PANSCAN:20,4567,E8D173419CA155A9,02,01	a network has been found
JPAN:22,2395,18BDC2597DB2F128	the device has just joined a PAN
+N=COO,22,03,2395,18BDC2597DB2F128	the device is the coordinator of a PAN
+N=FFD,22,03,2395,18BDC2597DB2F128	the device is a member of a PAN as a Full Function Device (ie a router)
NEUNODE:A52C,000D6F000059435B,FD3E FFD:000D6F000059435B,A52C	a new device has just joined the PAN



Smart.
Connected.
Energy-Friendly



Products
www.silabs.com/products



Quality
www.silabs.com/quality



Support and Community
community.silabs.com

Disclaimer

Silicon Laboratories intends to provide customers with the latest, accurate, and in-depth documentation of all peripherals and modules available for system and software implementers using or intending to use the Silicon Laboratories products. Characterization data, available modules and peripherals, memory sizes and memory addresses refer to each specific device, and "Typical" parameters provided can and do vary in different applications. Application examples described herein are for illustrative purposes only. Silicon Laboratories reserves the right to make changes without further notice and limitation to product information, specifications, and descriptions herein, and does not give warranties as to the accuracy or completeness of the included information. Silicon Laboratories shall have no liability for the consequences of use of the information supplied herein. This document does not imply or express copyright licenses granted hereunder to design or fabricate any integrated circuits. The products are not designed or authorized to be used within any Life Support System without the specific written consent of Silicon Laboratories. A "Life Support System" is any product or system intended to support or sustain life and/or health, which, if it fails, can be reasonably expected to result in significant personal injury or death. Silicon Laboratories products are not designed or authorized for military applications. Silicon Laboratories products shall under no circumstances be used in weapons of mass destruction including (but not limited to) nuclear, biological or chemical weapons, or missiles capable of delivering such weapons.

Trademark Information

Silicon Laboratories Inc.®, Silicon Laboratories®, Silicon Labs®, SiLabs® and the Silicon Labs logo®, Bluegiga®, Bluegiga Logo®, Clockbuilder®, CMEMS®, DSPLL®, EFM®, EFM32®, EFR®, Ember®, Energy Micro, Energy Micro logo and combinations thereof, "the world's most energy friendly microcontrollers", Ember®, EZLink®, EZRadio®, EZRadioPRO®, Gecko®, ISOModem®, Precision32®, ProSLIC®, Simplicity Studio®, SiPHY®, Telegesis, the Telegesis Logo®, USBXpress® and others are trademarks or registered trademarks of Silicon Laboratories Inc. ARM, CORTEX, Cortex-M3 and THUMB are trademarks or registered trademarks of ARM Holdings. Keil is a registered trademark of ARM Limited. All other products or brand names mentioned herein are trademarks of their respective holders.



Silicon Laboratories Inc.
400 West Cesar Chavez
Austin, TX 78701
USA

<http://www.silabs.com>