FlexiPanel

Pixie Gateway™

Serial to ZigBee Mailbox Profile Transceiver with Command Interface

Summary

Pixie Gateway provides a serial interface to the MailBox profile doe ZigBee communications protocol for low data-rate wireless mesh networks. It is ideal for OEMs who need to add generic serial communications to their ZigBee networks. It incorporates an FCC / CE certified IEEE 802.15.4 transceiver and ZigBee MailBox stack.

Pixie Gateway provides a command-oriented gateway between the ZigBee network and host devices. The MailBox COM product provides a less flexible but command-free interface for transparent cable replacement applications.

Firmware Features:

- *Incorporates ZigBee stack and MailBox profile.*
- *Up to 8 functional clusters.*
- *MailBox services include:*
  - *Network joining*
  - *Device & service discovery*
  - *Data delivery*
  - *Sleep management*
  - *Custom commands*
- *Available in the following configurations:*
  - *HC-L stack profile*
    - *ZigBee coordinator*
    - *ZigBee router*
    - *ZigBee fast end device*
    - *ZigBee sleepy end device*
  - *Other stack profiles as required*

Compatible Products

- *Fully compatible with all MailBox products, including:*
  - *MailBox API*
  - *Pixie DARC*
  - *UZBee Gateway*
  - *COMdongle (when available)*

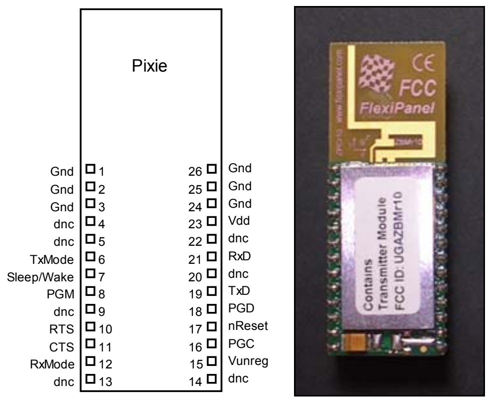


Fig 1. Pixie Gateway
(viewed from above)

Hardware Features:

- *2.4GHz IEEE 802.15.4 RF module*
- *FCC / CE / IC compliant*
- *Signature 'G' antenna, free-space range 120m, compact, low 'hand-effect' design*
- *115200 baud serial interface with flow control*
- *Sleep control input*
- *56mm x 20mm x 9mm*

Ordering Information

Table 1. Ordering information

Part No	***Description***
PIXIE-PGC*x*	Pixie Gateway (coordinator)
PIXIE-PGR*x*	Pixie Gateway (router)
PIXIE-PGF*x*	Pixie Gateway (fast end device)
PIXIE-PGS*x*	Pixie Gateway (sleepy end device)
x indicates stack profile supported: H = Home Controls	
Pixie Gateway is free for use with FlexiPanel Pixie products	

Manufactured to ISO9001:2000

Contents

Pin Connections

Pin Number	***Pin Name***	***Description***
1,2,3	*Gnd*	Power supply ground reference and ground plane connection (note 3)
4	*dnc*	Do not connect
5	*dnc*	Do not connect
6	*TxMode*	Reserved, connect low or to a spare host controller output pin for forward compatibility
7	*Wake / Sleep*	Sleep control (see text for modes of operation)
8	*PGM*	Programming input (note 1)
9	*dnc*	Do not connect (note 3)
10	*RTS*	Flow control output / device ready indicator. Data may be sent to the RxD pin if RTS is low.
11	*CTS*	Flow control input (note 4) To suspend data transmission on the TxD pin, set CTS high.
12	*RTxMode*	Reserved, leave unconnected or connect to a spare host controller input pin for forward compatibility
13	*dnc*	Do not connect
14	*dnc*	Do not connect
15	*Vunreg*	Unregulated voltage input (note 2)
16	*PGC*	Programming input (note 1)
17	*nReset*	Reset input (active low)
18	*PGD*	Programming input (note 1)
19	*TxD*	Serial output (8N1, 115200 baud)
20	*dnc*	Do not connect
21	*RxD*	Serial data input (8N1, 115200 baud)
22	*dnc*	Do not connect
23	*Vdd*	Regulated power supply input Regulated power supply output (note 2)
24,25,26	*Gnd*	Power supply ground reference and ground plane connection

Table 1. Pin descriptions for Pixie & Pixie Lite

1. Should be pulled low via a 10K resistor for normal operation.
2. Requires optional voltage regulator option to be fitted for onboard regulation to be functional.
3. In future devices, this pin may become a Vddcore power supply pin. Refer to the *Future releases of Pixie & Pixie Lite* section of this data sheet and the documentation for 18F46L10 and 18F45L10 series devices from Microchip Technology (www.microchip.com).
4. Connect low if flow control is not required

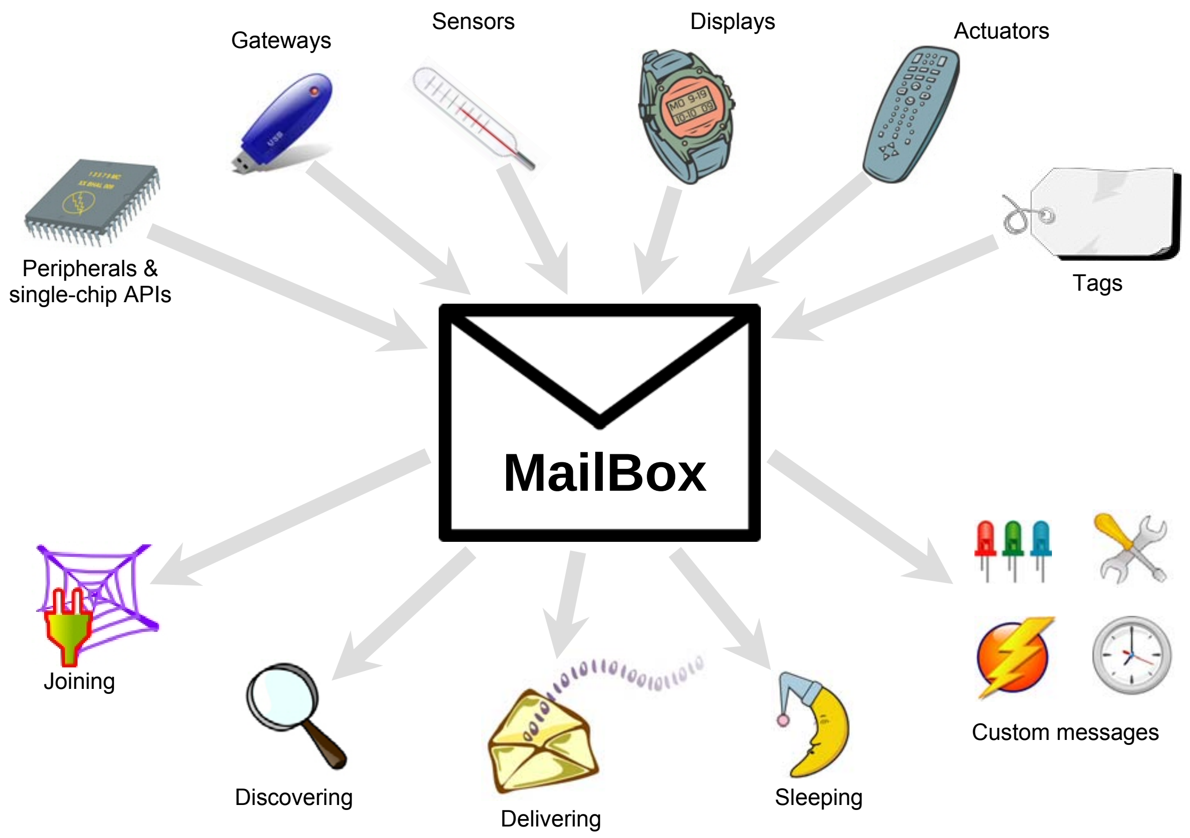
MailBox Overview

Pixie Gateway is a 2.4GHz IEEE 802.15.4 transceiver operating the MailBox profile for ZigBee / IEEE 802.15.4. It accepts commands via a 3.3 volt serial interface.

The IEEE 804.15.4 protocol provides services for transceiver devices to discover each other and then exchange packets of data in a reliable, error-free manner. Above it, the ZigBee layer allows multi-hop communications across mesh networks. At the top, the MailBox layer provides generic data communications without the need to understand how the lower layers work.

MailBox is designed for interoperability and ease-of-use. In particular, it offers compatibility with existing MailBox products and avoids the need to apply for ZigBee Alliance membership. The protocol ensures data is received at the destination error-free and in the sequence in which it was transmitted, but does not attempt to define the content of the data. It is the applications responsibility, however, to know what to do with it when it arrives. The MailBox layer communicates using the MailBox ZigBee profile which is being submitted to the ZigBee organization for registration. Until registration is complete, ‘placeholder’ profile IDs have been used which may be subject to change.

The MailBox layer adds two important features to the ZigBee communications protocol. The first is packet sequencing, which ensures that packets arrive in sequence, without loss or repetition. The second is functional clusters, which address devices according to the function they are supposed to perform, rather than their MAC or ZigBee short addresses. In addition, it provides guidelines for network-wide sleep and power outage recovery, which are not defined by the lower layers.


Gateways
Sensors
Displays
Actuators
Peripherals & single-chip APIs
Tags
MailBox
Joining
Discovering
Delivering
Sleeping
Custom messages

Gateway Overview

Pixie Gateway provides a command-oriented interface to the MailBox profile for ZigBee. Each message is composed of the **+** character followed by four characters. The last character is **R** for requests sent from the host device to Pixie Gateway. Confirmation responses from Pixie Gateway end in the letter **C**. Unprompted indication messages from Pixie Gateway end with the letter **I**.

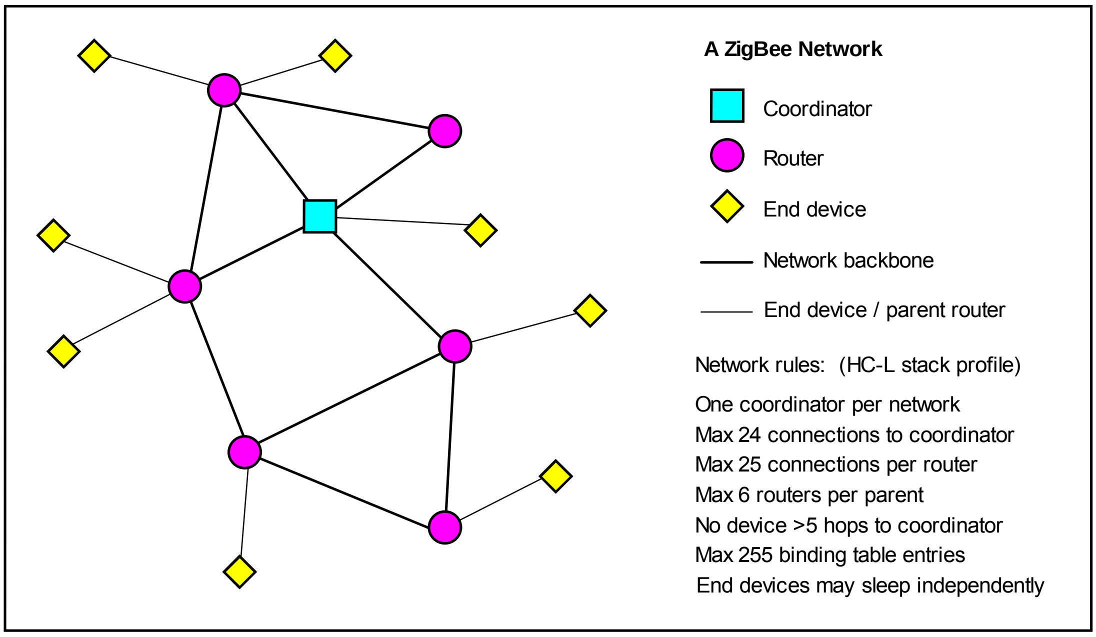
Device & Stack Profile Types

Several device types are employed in a ZigBee network. The *coordinator* is the device which dictates network-wide rules such as operating frequency. There must be one coordinator in a network and it is the first member of the network. The network is then built by joining new devices on to existing devices in the network.

Routers are devices which can forward messages on behalf of other devices. They form the basis of the multi-hop messaging system. Unless the entire network has agreed to sleep simultaneously, routers and the coordinator shall be always on and listening for messages.

End devices cannot route messages on behalf of other devices, and they cannot have admit new devices into the network. *Fast end devices* keep their radios on all the time. Sleepy end devices can spend most of their time asleep; when they wake, they must check with their parent router to see if there are any messages waiting for them.

With any ZigBee network, there are rules about how many child devices a router may have, how far a device may be from the coordinator, etc. These rules are referred to collectively as the stack profile. Devices with different stack profiles are not compatible. Since Pixie Gateway is intended to piggyback on any ZigBee network, it can be made available in any stack profile implementation. The default implementation is the Home Controls profile, whose rules are indicated in the figure below.


A ZigBee Network
Coordinator
Router
End device
Network backbone
End device / parent router
Network rules: (HC-L stack profile)
One coordinator per network
Max 24 connections to coordinator
Max 25 connections per router
Max 6 routers per parent
No device >5 hops to coordinator
Max 255 binding table entries
End devices may sleep independently

Functional Clusters

ZigBee devices use an 8-byte globally unique address when joining a network. Once joined, they are assigned a 2-byte address which is unique within the network. Since neither address is under the control of the application, this leaves unresolved the problem of which address a device should transmit its messages to. Functional clusters help perform this task.

MailBox devices use two-byte functional clusters to identify themselves according to the functions they perform. All devices support cluster 0x0000 and they may support any number of other clusters. In addition to sending a message to a specific device short address, a message may be broadcast to all devices which support a particular cluster. Function-specific device discovery allows network devices to bind together without user intervention.

The interpretation of cluster values 0x0001-0x00FD is application specific and not defined in the protocol. For example, in a sensor network, all sensors might support clusters 0x0000 and 0x0001. Data gathering gateways might support clusters 0x0000 and 0x0002. A gateway could send a message to all sensors by broadcasting to cluster 0x0001. A sensor can search for a data gateway using device discovery of devices supporting functional cluster 0x0002 and then direct MailBox packets directly to them.

It is anticipated that 2-byte clusters will be supported in ZigBee 1.1. At that time, the range of application specific clusters will be extended to 0x0001-0xFFFD.

Cluster 0xFFFE shall be the Redirect Address signifier. If 0xFFFE is specified as a destination address, the actual destination shall be that specified by the most recently received Redirect Address message. This allows reduced function devices such as sensors to be instructed remotely as to where to send their data.

Cluster 0xFFFF signifies a null cluster; messages addressed to the null cluster will be discarded without transmission.

Network Joining

A device can’t do much until it has established communication with the network. Therefore after initializing, a device should attempt to join the network using the +MJNR message. If this is the first time it has joined the network, the device which will be its parent should be place in the “permit join” state. It it is a Gateway device, this is achieved with the +MPJR command.

Device and Service Discovery

A variety of mechanisms are provided for working out what other devices are on the network and who to send messages to. Device discovery (+MDDR) is used to search for devices supporting specific functional clusters. Service discovery (+MSDR) is used to interrogate a specific device. Present messages (+MPRR) announce to other devices that this device is present on the network. Redirect requests (+MRDR) instruct other devices where to send their messages.

Data Delivery

Payloads of up to 64 bytes of data can be sent at a time in a packet with the +MDAR request. Packets can be broadcast to all devices supporting a functional cluster, or unicast to a specific short address. They can be unacknowledged, or acknowledged.

- *Unacknowledged*. Frame sequence numbers are used to determine if a frame has arrived out of sequence or has been lost. Loss of sequence is reported to the receiving application but no other action is taken. No acknowledge is sent to the source. Broadcast communication is permitted.

- *Acknowledged*. The destination will formally accept a frame before the source transmits the next frame. Broadcast transmissions are not permitted. If a frame does not reach the destination, it can be repeated, thus guaranteeing uninterrupted, sequenced data.

Sleep Management

Two sleep modes are provided. *Device sleep* permits sleepy end devices to sleep at any time. *Network sleep* allows the entire network to sleep simultaneously.

Sleep is entirely under the control of the host. On receipt of a network sleep (+MNSI) message, the host instructs the module when and whether to sleep. Other than on receipt of an +MNSI message, a device should only be put in the sleep state if it is a sleepy end device. A sleepy end device may sleep indefinitely, but if it needs to poll its parent for messages, it must be woken intermittently.

Three device settings affect sleep behavior:

The *PersistenceTime* is the length of time that a router will hold a message for a sleeping child.

The *PollRate* is the frequency with which a sleepy end device will poll its parent for messages while it is awake. (While asleep, it cannot poll at all.) If *PollRate* is set to zero, the parent will only be polled on wakeup.

The *SleepWakeMode* device setting governs the operation of the Sleep/Wake pin. In the default mode (+DSWR=00 command), the device is placed in sleep mode using the +MSMR command and woken by a change of state of the Sleep/Wake pin. In this mode, the Sleep/Wake pin is tied to the RxD input so that sending a character will wake the device. (The character will otherwise be ignored. A null character is recommended.)

In the alternate Sleep/Wake pin mode (+DSWR=01 command), the device will sleep when the Sleep/Wake pin is high and wake when the pin is low.

In both modes, the RTS pin will be high during sleep and during waking. When waking is complete, RTS will go low and a +DRYI message will be generated.

Custom Messages

128 custom messages are available for application-specific use.

Application ID

The use of functional clusters alone does not allow for the fact that two MailBox applications may need to coexist on a network. Specifically, one manufacturer may associate different meanings to different clusters.

To allow for this, a 4-byte application ID is associated with each manufacturer's interpretation of clusters. This application ID is included with all data transactions and is also available for direct querying to verify a device prior to unicast transmission. It also allows manufacturer specific messages to be transmitted.

In order for application IDs to be unique, FlexiPanel will allocate a 3-byte MailBox Unique Identifier (MUI) on request. This is free of charge.

Application IDs shall be allocated as follows:

Bytes 0-2:	Manufacturer's MUI number
Byte 3:	Assigned by manufacturer

The 'free-for-all' MUI 00:00:00 may be used by any application provided it can ensure that it is the only MailBox application on the network.

Endpoints

ZigBee endpoints allow several applications to share one ZigBee radio. The MailBox protocol uses one ZigBee endpoint. By default it is 0x10, but this may be changed for the convenience of other applications. It may be any endpoint in the allowed range 0x01-0xF0. The same endpoint must be used for all devices employing the same Application ID.

Notation, Byte & Bit order

All numbers in this documentation are in decimal unless prefixed with 0x, in which case they are hexadecimal. Index counting starts at zero, so the first byte of a message is byte zero.

Multi-byte data is transmitted least-significant byte first ('little-endian'), as is standard in the ZigBee specification.

Symbol Periods

Several time periods are expressed in units of Symbol Periods. A symbol period is 1/62,500 second. 0x10000 symbol periods equates to just over one second.

Copy Protection

To protect against copying, if the Pixie Gateway firmware is run on any hardware except FlexiPanel Pixie and Pixie Lite products, it will cease to function after approximately two minutes.

Evaluation Kit

The easiest way to get to know Pixie Gateway is with the ZigBee Evaluation Kit available from FlexiPanel. This will also require a Microchip ICD2 In-Circuit Debugger to program the firmware into the Pixies supplied.

In the evaluation boards, the I/O pins are connected as follows:

Pin Number	Pin Name	Description
6	*TxMode*	LED labeled "A4 / EP4"
7	*Wake / Sleep*	Switch labeled "EP2 A2"
8	*PGM*	Programming input (note 1)
10	*RTS*	LED labeled "A4 / EP5 / RTS"
11	*CTS*	Switch labeled "Config SW" *Ensure jumper A8 – A9 is fitted.*
12	*RxMode*	Switch labeled "ModeA"
17	*nReset*	Reset push switch
19	*TxD*	Serial output (8N1, 115200 baud)
21	*RxD*	Serial data input (8N1, 115200 baud)

Release notes, version 0B400115103521200906pt

In this release, security is not supported. ***pt*** refers to the stack profile / device type. Refer to the **DVRC** message for details.

Bibliography

IEEE 802.15.4 specification, downloadable from *www.ieee.org*.

The MailBox Profile for ZigBee, downloadable from *www.flexipanel.org*.

ZigBee for Applications Developers, white paper downloadable from *www.flexipanel.com*.

ZigBee Specification, downloadable from *www.zigbee.org*.

Usage examples

The following examples show how Pixie Gateway can be used. To test these functions, use the HyperTerminal terminal emulator available in Microsoft Windows to communicate with Pixie Gateway. The ZigBee Evaluation Kit available from FlexiPanel Ltd contains two evaluation boards, which is ideal for experimentation and demonstrates communications with the Pixie DARC sensor device.

Application software interfacing with Pixie Gateway should treat it as a serial COM port device. Since data flow is asynchronous and unpredictable, data transmission should not block data reception and vice versa. For Microsoft Windows programming, for example, it is vital that overlapped file I/O is used.

Remember, if you are using the evaluation boards, that the "Config switch" *is the flow control for the TxD output. It must be set to low and must be is connected to the pin by fitting jumper A8-A9, otherwise you will not get any output!*

Device Present

The presence of Pixie Gateway can be checked by issuing a version number request:

+DVRR	(request version)
+DVRC=0B40011510352120090611	*(confirmation – version number may vary)*

Setting the MAC Address

The MAC address should be set prior to sending any MailBox-layer commands (i.e. commands beginning with M). If you bought the device with the firmware pre-loaded, the MAC address will have probably have been set already for you. Otherwise, to obtain a MAC address, contact FlexiPanel Ltd.

+DSMR=0500004138C81500	(request to set MAC address)
+DSMC	*(confirmation – confirms MAC address set)*

Note that the MAC address is specified Little-Endian, i.e. the MAC address specified in the example would normally be written as ***0015C83841000005***. MAC addresses must be unique. The Gateway will automatically reset after confirming this command.

Starting a network

If a gateway device is a coordinator, it starts the network as follows:

+MJNR=00	(request start network)
+MJNC=000000	*(confirmation– device address is 0x0000)*

Note that you can only issue MJNR once. If you need to repeat it, you must reset the gateway using DRSR first.

Joining a network

For other devices to join the network, a device which is already a member of the network must be put into the permit-join state. For Gateway devices, this may be done as follows:

+MPJR=FF	(request permit joining indefinitely)
+MPFC=00	*(confirmation – permitting joining indefinitely)*

The new device may then be joined onto the network. If it is a Gateway device, joining is initiated as follows:

+MJNR=00	(request join)
+MJNC=006F79	*(confirmation on new device– joined with address 0x796F)*

Note that you can only issue MJNR once. If you need to repeat it, you must reset the gateway using DRSR first.

Device Discovery

To scan for other MailBox devices on the network, use the MDDR command. You can filter the responses by the functional clusters that other devices support. For example

+MDDR=0000	(Discover devices supporting cluster 0x0000, i.e. all devices)
+MDDI=6F79	*(indication - Device 0x796F responds)*
+MDDC=00	*(confirmation – device discovery complete)*

Service Discovery

To ask a specific device for further information, issue a service discovery request. For example

+MSDR=6F79800000	(Ask device *0x796F* what its application ID is)
+MDDC=00800000000000	*(confirmation – application ID is 00000000)*

A particularly useful service discovery request is to enquire the short addresses of a device's children. Working down from the coordinator (address 0x0000), you can get a map of the entire network. The format is:

+MSDR=xxxx010004xxxx0100	(where **xxxx** is the address of the device to ask)
+MDDC=[30 digits]nnnnyyyy…	*(where **nnnn** is the number of children and **yyyy** is a list of their addresses)*

For example:

+MSDR=000001000400000100

+MDDC=00010013000400000000C81500000003006F7970797179

The response from the coordinator indicates it has 3 children with addresses 0x796F, 0x7970, 0x7971.

Sending Data

Data is sent using the MDAR command. The format is:

`+MDAR=zzxxxxnnww…` where

`zz`	is 00 for unacknowledged, 01 for broadcast and 02 for acknowledged.
`xxxx`	is the address of the destination (or the functional cluster in the case of broadcast)
`nn`	is the number of bytes of data
`ww…`	is the data, in hex

For example:

`+MDAR=006F790412345678`	(send **`12345678`** to device 0x796F)
`+MDAC=00FF0101`	*(first 2 digits 00 indicate success)*

Receiving Data

Data is received as an MDAI indication. The format is:

`+MDAI=vvyyyyssnnww…` where

`vv`	is 00 for success,
`yyyy`	is the address of the source
`ss`	is a sequence number (ignore)
`nn`	is the number of bytes of data
`ww…`	is the data, in hex

For example:

`+MDAI=000000FF0412345678`	(received **`12345678`** from device 0x0000)

Message Reference

Format

Messages sent to and received from Pixie Gateway take the form of ASCII message strings with the following general format:

```
+XXXX=hhhhhhhh<CR><LF>
```

All messages begin with the **`+`** character followed by the four-letter command or response code ***`XXXX`***, followed by the **`=`** character. If any additional data accompanies the message, it usually follows as a series of bytes each represented two hexadecimal digits ***`hh`***. Multi-byte integers are parsed *little-Endian,* i.e. least significant byte first. If no additional data accompanies the message, the **`=`** character may be omitted. Finally, the string is terminated with a carriage return character **`<CR>`** and optionally a linefeed **`<LF>`** character. Extra **`<CR>`** and/or **`<LF>`** characters are permitted between messages.

Inline editing (*e.g.* pressing backspace) is not supported. If typing a message from HyperTerminal, abandon it by entering a **`Z`** character and then press *Enter*. (You can ignore the ***`+DERI=40`*** syntax error generated.)

Do not send a message until processing of the previous message is complete. While processing a command, Pixie Gateway ignores any characters prior to the first **`+`** character encountered.

Device Messages

Messages starting with a ***`D`*** character relate to device settings and values.

Ready indication (DRYI)

DRYI indicates the device has completed initialization. Example:

```
+DRYI
```

Device Wake-up indication (DDWI)

DDWI indicates the device has completed wake-up. Example:

```
+DDWI
```

Error indication (DERI)

DERI indicates a device-level error occurred. Example:

```
+DERI=01
```

The following error values may be reported:

Value	Interpretation
01	MailBox was not ready when command was sent
02	Ran out of memory
03	Operation not permitted prior to +MJNR command
40	Command syntax in error
41	MAC address must be set with +DSMR command prior to sending any mailbox messages
42	MAC address may not be changed once set. Note this error is fully recovering, i.e. there is no harm in repeatedly sending the +DSMR command, it is simply ignored after the first time.
43	+MJNR command can be issued once only; reset before trying again.
80+	Internal error, contact FlexiPanel Ltd quoting value

Device Get request (DG*x*R)

DG***x***R requests device-level attribute data. Refer to DS***x***R request for a list of attributes.

Command format	***+DG*x*R***

Device Get confirm (DG*x*C)

DG***x***C confirms device-level attribute data.

Command format		***+DG*x*C=***{varies according to **x**}*
AttributeValue	*(see* **DS*x*R***)*	Attribute value

Device Set request (DS*x*R)

DS***x***R requests to set device-level attribute data.

Command format		***+DS*x*R=***{length varies with **x**}*
AttributeValue	*(see below)*	Attribute value

The following attributes are implemented. They are non-volatile and should only be set prior to joining the network. Repeatedly setting them to the same values will not exhaust the Flash memory. Setting values may not necessarily have any effect until the ZigBee stack is next initialized.

All values are little-endian, i.e. least significant byte first. For example, to set the MAC address 1234567890ABCDEF, the command is +DSMR=EFCDAB9078563412.

Attribute	**Bytes**	***x***
ApplicationID	4 bytes	A
ApplicationEndpoint	1 byte	E
ClusterList	14 bytes	C
MailBoxTimeout	4 bytes	T

Attribute	**Bytes**	***x***
PersistTime	4 bytes	P
PollRate	4 bytes	L
SleepWakeMode	1 byte	W
TransmitVolume	1 byte	V
SequenceBufferSize	1 byte	S
RetryDelay	2 bytes	R
RedirectAddr	3 bytes	D
MACaddress	8 bytes	M

ApplicationID Four-byte Application ID. If the application can guarantee that it is the only operator of MailBox applications on the network, the three most significant bytes may be 00:00:00; the least significant byte may then be application specific. To use a unique allocation of the upper three bytes, refer to the MailBox profile definition. Default value is 0x00000000**.**

ApplicationEndpoint Application Endpoint used by this Application ID. The default value is 0x10**.**

ClusterList A list of seven 2-byte functional clusters which are to be supported in addition to the mandatory cluster, 0x0000. If fewer than seven clusters are to be specified, the remainder should be set to the null cluster value, 0xFFFF. Default value is all null clusters**.** With ZigBee 1.0, the maximum non-null cluster value is 0x00FE.

MailBoxTimeout Number of symbol periods expected for a message to propagate to another node in a network and for a reply to be received. If a reply is not received within this time, (or a multiple of it where appropriate,) an operation will be abandoned. Default value is 0x00010000 – one second.

PersistTime Symbol periods that a message should reside on a router pending collection by a sleepy end device. Default value is 0x00400000 – approx one minute.

PollRate Symbol periods between polls to a parent. Sleepy devices are required to do this while awake in order to pick up messages. Other devices (except the coordinator) may choose to do it simply to confirm that the parent is still there. Default value is 0x0080 for sleepy devices (approx half a second) and 0x0000 for other devices.

SleepWakeMode Defines operation of the Sleep/Wake pin. Refer to the sleep management section for details.

TransmitVolume Transmit power as device in CC2420 documentation. Default value is 0xFF, maximum power.

SequenceBufferSize Number of records to buffer in order to monitor packet sequence / loss / repetition. Refer to MailBox profile definition for details. Default value is 0x10 for Pixie, 0x04 for Pixie Lite. Maximum value 0x2A.

RetryDelay The period, in multiples of 256 symbol periods, which a remote device should pause before attempting to retransmit if incoming data is rejected for flow control reasons. Flow control might reject the data if the CTS input is high or its receive buffer is full. Refer to MailBox profile definition. Default value is 0x0100, i.e. 0x010000 symbol periods (approx one second).

RedirectAddr The destination to send messages to when the Redirect Cluster (0xFFFE) is specified in a MDAR data request. If the first byte is zero, the second and third bytes shall be interpreted as a short address. If the first byte is nonzero, the second and third bytes shall be interpreted as a broadcast cluster. The value 0xFFFFFF will be interpreted as no address being specified. No transmission shall be made and the Data.confirm will report success. Unlike other MIB values, this value is volatile and is reset to 0xFFFFFF on startup.

MACaddress The 8-byte MAC address of the device. The Gateway will automatically reset after confirming this command.

Device Set confirm (DS*x*C)

DS*x*C confirms device-level attribute data setting.

Command format		***+DSxC=****{1 byte}*
status	1 byte	Zero if successful

Version request (DVRR)

DVRR requests firmware version information from Pixie Gateway. Pixie Gateway will generate a DVRC confirmation in response. Example:

+DVRR

Version confirm (DVRC)

The DVRC confirmation is generated by Pixie Gateway in response to a DVRR request. Example:

+DVRC=0B40011510352120090611

The response takes the form ***+DMCC=PPPPPPPPVVVVVVDDMMYYPT***, where the digits are as follows:

PPPPPPPP	Product ID (0B400115 indicates Pixie Gateway firmware)
VVVVVV	Pixie Gateway version number
DDMMYY	Firmware release date
P	Stack profile
T	Device type

P = 1 Home Controls stack profile

T = 1 FFD Coordinator
T = 2 FFD Router
T = 3 RFD Fast End Device (Rx On when idle)
T = 4 RFD Sleepy End Device (Rx Off when idle)

MailBox Messages

Messages starting with an ***M*** character relate to the MailBox layer. Many responses begin with a status byte which will be zero if the operation was successful. Refer to the ZigBee specification for the meaning of non-zero status bytes.

Join.request (MJNR)

In the case of a coordinator, MJNR requests to initialize the ZigBee stack and form a network.

In the case of other devices, MJNR requests to initialize the ZigBee stack and join a network.

Command format		***+MJNR=****{1 byte}*
Erase	1 byte	If non-zero, network information is erased

On a coordinator, this command will start a network. If it has never started a network before, or *Erase* was true, it will be a new network. Otherwise it will restart an existing network.

On routers and end devices, if it has never before joined a network or *Erase* was true, it will then attempt to join any network. A router or coordinator on the network it is supposed to join must be in range and in the *PermitJoin* state to allow the new device to join.

Otherwise, if it was previously a member of a network, it will rejoin the network and no special join permission will be required.

MJNR should only be sent once. If the first attempt is not successful, the device should be first reset using DRSR.

Join.confirm (MJNC)

In the case of a coordinator, MJNC reports on an attempt to initialize the ZigBee stack and form a network.

In the case of other devices, MJNC reports to an attempt to initialize the ZigBee stack and join a network.

Command format		***+MJNC=****{3 bytes}*
Status	1 byte	Zero if successful
ShortAddress	2 bytes	Short address allocated to this device

Permit-Join.request (MPJR)

MPJR requests that the permit join state is changed. This command does not apply to end devices.

Command format		***+MPJR=****{1 byte}*
PermitDuration	1 byte	`0x00` = Do not permit joining `0x00` = Permit joining for *PermitDuration* seconds `0x00` = Permit joining indefinitely

Permit-Join.indication (MPJI)

MPJI indicates a new device has been admitted to the network by this node. It may or may not have previously been a member.

Command format		***+MPJI=****{24 bytes}*
Filler1	8 bytes	*(ignore)*
ShortAddress	2 bytes	Short address of device joining
Filler2	6 bytes	*(ignore)*
ExtendedAddress	8 bytes	Long address of device joining

Permit-Join.confirm (MPJC)

MPJC responds to a request to change the permit join state.

Command format		***+MPJC=****{1 byte}*
Status	1 byte	Zero if successful

Leave.indication (MLVI)

MLVI indicates this device was instructed to leave the network. This is a rare message, only generated by non-MailBox devices.

Command format	***+MLVI***

Sync-Loss.indication (MLSI)

MLSI indicates this device did not receive a reply from its parent during a poll operation. Polling will have been tried four times; the application should assume that the parent is no longer operating, on this frequency at least. The recommended course of action is to sleep a while to conserve power, then reset and then attempting to rejoin the network.

Command format	***+MSLI***

Device-Discovery.request (MDDR)

MDDR initiates a scan for other MailBox devices with the Functional Cluster specified by *FuncID*.

Command format		***+MDDR=****{2 byte}*
FuncID	2 byte	Functional cluster to scan for

Device-Discovery.indication (MDDI)

MDDI indicates a device responded to the device discovery request. It will be a MailBox device supporting the Functional Cluster specified. However, its Application ID has not been verified; it is possible that it is a response from another MailBox application running on the same network. The Application ID can be verified using an MSDR service discovery request.

Command format		***+MDDI=**{2 bytes}*
ShortAddress	2 bytes	Short address of device responding

Device-Discovery.confirm (MDDC)

MDDC confirms that a device discovery scan has completed.

Command format		***+MDDC=**{1 byte}*
Status	1 byte	Zero if successful

Service-Discovery.request (MSDR)

MSDR initiates a request for information from the ZigBee device with address *Destination*.

Command format		***+MSDR=**{variable length}*
Destination	2 byte	Device from which to request service discovery
InfoType	2 byte	Service to request
DataPayloadLen	1 byte	Length of ZDO payload
DataPayload	*DataPayloadLen* bytes	ZDO Payload

InfoType indicates the information required. If equal to 0x0080, the information requested is the device's application ID. *DataPayloadLen* should be zero and *DataPayload* shall be empty.

If *InfoType* is anything other than 0x0080, the value shall be interpreted as a ZigBee Device Object (ZDO) cluster and the data will be sent to the ZDO on the remote device. For further information refer to the ZigBee specification. Note that for ZDO requests, *Destination* does not have to be a MailBox device.

If supplementary information is required for a ZDO request, *DataPayload* shall contain the supplementary information and *DataPayloadLen* shall indicate its length. If supplementary information is not required, *DataPayloadLen* shall be zero and *DataPayload* shall be empty.

The ZDO services currently supported by MailBox devices are shown below. Refer to the ZigBee specification for full details of their actual function and request / confirm payloads.

Name	***Description***	***InfoType***
IEEE_ADDR_req	Get MAC address & child device information	0x0001
NODE_DESC_req	Get node descriptor	0x0002
POWER_DESC_req	Get power descriptor	0x0003
SIMPLE_DESC_req	Get simple descriptor	0x0004
ACTIVE_EP_req	Get active endpoint list	0x0005

Service-Discovery.confirm (MSDC)

MSDC confirms that service discovery has completed. If the *InfoType* service was ApplicationID (0x0080), the response is as follows:

Command format		***+MSDC=***{*variable length*}
Status	1 byte	Zero if successful
InfoType	2 bytes	Service requested (0x0080)
DataPayload	2 bytes	Application ID

If the *InfoType* service was a ZDO request, the response is as follows.

Command format		***+MSDC=***{*variable length*}
Status	1 byte	Zero if successful
InfoType	2 bytes	Service requested
DataPayloadLen	1 byte	Length of ZDO response payload
DataPayload	*DataPayloadLen* bytes	ZDO response payload, if any

Data.request (MDAR)

MDAR sends data to the device(s) specified by *Destination*. If *IsBroadcast* is true, *Destination* is interpreted to be a Functional Cluster and the message is broadcast. If *IsBroadcast* is false, *Destination* is interpreted to be a short address and the message is unicast.

If *IsAcknowledge* is true, the MDAC confirmation will only be returned when the destination has acknowledged receipt of the message. Acknowledge is only permitted for unicast messages. It is possible that acknowledged messages may be returned with confirm status *TIMED_OUT*, *RETRY_LATER*, or *CHECKSUM_FAIL*. In such cases, the sender may elect to attempt to resent the data at a later time by repeating the MDAR request. If it chooses to do so, it should set the *IsRetry* bit.

Command format		***+MDAR=***{*variable length*}
Flags	1 byte	Bit 0: IsBroadcast Bit 1: IsAcknowledge Bit 3: Is Retry
Destination	2 byte	Short address or Functional cluster
DataPayloadLen	1 byte	Length of Data payload
DataPayload	*DataPayloadLen* bytes	Data payload

Data.indication (MDAI)

MDAI indicates a data frame was received from another device.

Command format		***+MDAI=****{length varies}*
Status	1 byte	Transmission status report
ShortAddress	2 bytes	Short address of source
Seq	1 byte	Sequence number used
DataPayloadLen	1 byte	Length of Data payload
DataPayload	*DataPayloadLen* bytes	Data payload

Seq will indicate the sequence number used to transmit the data. This is provided for informational purposes, should a higher layer wish to attempt data recovery. *SourceAddr* will contain the short address of the sender.

Possible status return codes are gives below.

Name	***Description***	***Value***
SUCCESS	Operation completed successfully	0x00
UNKNOWN	This packet received OK but insufficient information to determine if it is in sequence	0x01
SEQUENCE_ERROR	An error occurred (unicast)	0x02
FRAMES_LOST	This packet received OK but earlier packet(s) are missing	0x07
LATE_FRAME	This packet received OK but a later packet has already been the subject of a Data.indication	0x08
RESET_MISMATCH	This packet received OK but a device has been reset	0x09

Data.confirm (MDAC)

MDAC confirms that a data request has completed.

Command format		***+MDAC=****{4 bytes}*
Status	1 byte	Zero if successful; see below for other codes
Seq	1 byte	Sequence number used
RetryDelay	2 bytes	Retry wait period

Seq will indicate the sequence number used to transmit the data. This is provided for informational purposes, should a higher layer wish to attempt data recovery.

Status will indicate the result of the transmission as follows:

Name	***Description***	***Value***
SUCCESS	Operation completed successfully	0x00
UNKNOWN	This packet received OK but insufficient information to determine is in sequence	0x01
SEQUENCE_ERROR	An error occurred (unicast)	0x02
NOT_PERMITTED	Operation not permitted	0x03
TIMED_OUT	Acknowledge not received	0x04
RETRY_LATER	Destination could not accept packet; try again later no sooner than	0x05
STACK_FAIL	A ZigBee stack failure occurred	0x06
RESET_MISMATCH	This packet received OK but a device has been reset	0x09
CHECKSUM_FAIL	Packet not received due to checksum failure	0x0A

SUCCESS shall only confirm that the destination(s) received the packets if the transmission was acknowledged. Otherwise, it shall only confirm successful transmission to immediate neighbors.

If the status is *TIMED_OUT*, *RETRY_LATER*, or *CHECKSUM_FAIL*, the application may retry the data request as many times as desired, provided it does so prior to transmitting any other packets. If retry, the MDAR *IsRetry* flag should be set to true. If the status is *RETRY_LATER*, the retry should occur no sooner than *256×RetryDelay* symbol periods later.

Seq is provided for informational purposes and indicates the sequence number used to transmit the packet. If the status is *STACK_FAIL*, *Seq* will contain the ZigBee stack error status value.

Present.request (MPRR)

MPRR sends a message to the device(s) specified by *Destination*, announcing its presence. If *Broadcast* is 0x01, *Destination* is interpreted to be a Functional Cluster and the message is broadcast. If *Broadcast* is false, *Destination* is interpreted to be a short address and the message is unicast.

Command format		***+MPRR=****{3 bytes}*
IsBroadcast	1 byte	0x01 if broadcast
Destination	2 byte	Short address or Functional cluster

Present.indication (MPRI)

MPRI indicates a presence message was received. *SourceAddr* will contain the short address of the transmitting device. It will have the correct Application ID. *ClusterList* will list the functional clusters that the device supports. (The mandatory cluster 0x0000 may or may not be in the list.) *NumCluster* will equal the number of clusters in *ClusterList*. Any null clusters (0xFFFF) in the list shall be ignored.

Command format		***+MPRI=****{length varies}*
ShortAddress	2 bytes	Short address of device responding
NumCluster	1 byte	Number of clusters listed
ClusterList	2 × *NumCluster* bytes	List of clusters supported

Present.confirm (MPRC)

MPRC confirms that a presence request has completed.

Command format		***+MPRC=****{1 byte}*
Status	1 byte	Zero if successful

Redirect.request (MRDR)

MRDR sends a message to the device(s) specified by *Destination*, requesting that they change their redirect address. If *IsBroadcast* is true, *Destination* is interpreted to be a Functional Cluster and the message is broadcast. If *IsBroadcast* is false, *Destination* is interpreted to be a short address and the message is unicast.

The message instructs the destination mailbox layer to set its redirect address to *AddrOrClust*, which shall be interpreted as a short address if *RedirFlags* bit 0 is 0, or as a broadcast cluster otherwise. Transmissions using the redirect address shall be acknowledged if *RedirFlags* bit 1 is 1.

Command format		***+MRDR=****{6 bytes}*
IsBroadcast	1 byte	0x01 if broadcast
Destination	2 bytes	Short address or Functional cluster
RedirFlags	1 byte	Bit 0: true if redirect address is Broadcast Bit 1: true if redirect address is Acknowledged
AddrOrClust	2 byte	Redirect Address Short address or Functional cluster

Redirect.indication (MRDI)

MRDI indicates its MIB_RedirectAddr was changed by another device.

Command format	***+MRDI***

Redirect.confirm (MRDC)

MRDC confirms that a redirect request has completed.

Command format		***+MRDC=****{1 byte}*
Status	1 byte	Zero if successful

Network-Sleep.request (MNSR)

MNSR sends a message to the device(s) specified by *Destination*, indicating that they may sleep. If *Broadcast* is true, *Destination* is interpreted to be a Functional Cluster and the message is broadcast. If *Broadcast* is false, *Destination* is interpreted to be a short address and the message is unicast.

Refer to *Network-Sleep.indication* for definitions of *ClosedownPeriod*, *SleepPeriod* and *WakeupPeriod*.

Command format		***+MNSR=****{xx byte}*
IsBroadcast	1 byte	0x01 if broadcast
Destination	2 bytes	Short address or Functional cluster to transmit to
ClosedownPeriod	2 bytes	Closedown Period
SleepPeriod	3 bytes	Sleep Period
WakeupPeriod	2 bytes	Wakeup Period

Network-Sleep.indication (MNSI)

MNSI indicates network sleep message was received. *SourceAddress* will contain the short address of the new device. It will have the correct Application ID.

ClosedownPeriod is the duration, in seconds, from the moment *Network-Sleep.indication* was generated, that the application should continue to keep the Gateway. The application may not initiate any new MailBox-layer requests within the *Closedown* period.

The *SleepPeriod* is the duration, in seconds, from the moment the *Network-Sleep.indication* message was received, that the Gateway may enter a sleep state. This is not automatic, it must be invoked using a Device-Sleep.request.

The *WakeupPeriod* is the duration, in seconds, immediately following the end of the *Sleep period*. During this period, the application may not initiate any transmissions unless it first receives a transmission from another device.

Sleep is optional on receipt of a *Network-Sleep.indication*. Devices which must remain awake in order to continue to provide services for non-MailBox nodes in the ZigBee network should stay awake.

Command format		**`+MNSI=`***{xx bytes}*
Source	2 bytes	Short address of originator
ClosedownPeriod	2 bytes	Closedown Period
SleepPeriod	3 bytes	Sleep Period
WakeupPeriod	2 bytes	Wakeup Period

Network-Sleep.confirm (MNSC)

MNSC responds to a request to change the permit join state.

Command format		**`+MNSC=`***{1 byte}*
Status	1 byte	Zero if successful

Custom.request (MCSR)

MCSR sends an application-specific message to the device(s) specified by *Destination*. If *IsBroadcast* is true, *Destination* is interpreted to be a Functional Cluster and the message is broadcast. If *IsBroadcast* is false, *Destination* is interpreted to be a short address and the message is unicast. Custom messages are not acknowledged.

Command format		**`+MCSR=`***{variable length}*
CustomFID	1 byte	Custom message number (0x80 – 0xFF)
Flags	1 byte	Bit 0: IsBroadcast
Destination	2 byte	Short address or Functional cluster
DataPayloadLen	1 byte	Length of Data payload
DataPayload	*DataPayloadLen* bytes	Data payload

Custom.indication (MCSI)

MCSI indicates a custom frame was received from another device.

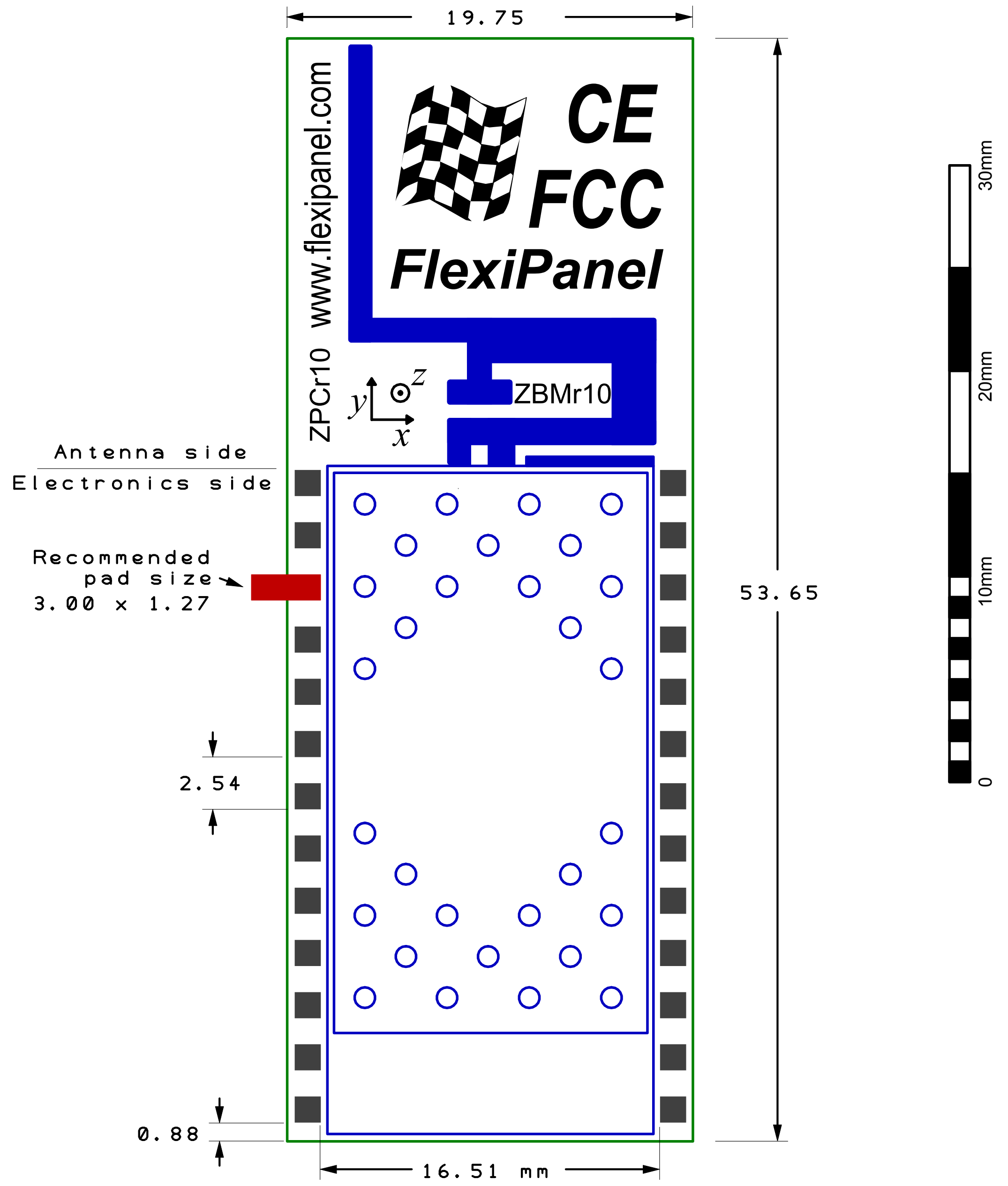
Command format		***+MCSI=****{length varies}*
CustomFID	1 byte	Custom message number (0x80 – 0xFF)
ShortAddress	2 bytes	Short address of source
DataPayloadLen	1 byte	Length of Data payload
DataPayload	*DataPayloadLen* bytes	Data payload

SourceAddress will contain the short address of the sender.

Custom.confirm (MCSC)

MCSC confirms that a data request has completed.

Command format		***+MCSC=****{4 bytes}*
Status	1 byte	Zero if successful



FlexiPanel
www.flexipanel.com

Drawing	DRWG-ZPCr10
Date	11 June 06
Drawn by	R G Hoptroff
Description	Pixie ZigBee module (rev 10)

Notes

1. Dimensions in mm
2. Module height 3.6mm
3. Keep antenna side as free of components as possible, preferably overhanging the edge of the main board
4. Pour as much grounded copper as possible on the main board, but none on top layer below Pixie module
5. If pins fitted, pin pitch is 20.32mm
6. Pixie Lite same size & footprint as Pixie
6. Incorporates FCC / CE / IC certified EasyBee transceiver ZBMr10

Reference

Radio Frequency

Max RF output power	1mW = 0dBm
RF frequency range	2400MHz to 2485MHz
Communications protocol	IEEE 802.15.4 (DSSS O-QPSK chip encoding)
Raw data rate	250kbit/s
RF channels	16
Free space range with integral antenna	Approx 120m

Electrical

Current consumption, excluding I/O pins	≤30mA
Current consumption, sleep mode	2µA
Supply Voltage (regulated) Vcc	2.1V to 3.6V

Mechanical

Max operating/storage temperature	-40ºC to +85 ºC
Dimensions L×W×H mm	55.7 × 20.3 x 10.5 (note 1)

Regulatory

FCC compliance	G-antenna version compliant, awaiting certificate
CE compliance	G-antenna version compliant, awaiting certificate
IC (Industry Canada) compliance	G-antenna version compliant, awaiting certificate

Contact Information

FlexiPanel

Developed by:

FlexiPanel Ltd
2 Marshall St, 3rd Floor,
London W1F 9BB, United Kingdom
email: support@flexipanel.com
www.flexipanel.com

Manufactured and distributed by:

R F Solutions Ltd
Unit 21, Cliffe Industrial Estate,
Lewes, BN8 6JL, United Kingdom
email : sales@rfsolutions.co.uk
http://www.rfsolutions.co.uk
Tel: +44 (0)1273 898 000