

Contents

| | |
|---|----|
| RAD24 Manual | 1 |
| Contents | 1 |
| Overview | 3 |
| Hardware Specifications | 3 |
| Environmental | 3 |
| Radio Specification 2.4Ghz | 3 |
| Physical | 4 |
| Communications Interface | 4 |
| Hardware Interfacing | 4 |
| Data Transfer | 4 |
| Packet Protocol | 6 |
| Packet Format: | 6 |
| Format of packets to module, from module and transmitted | 6 |
| Packet Type Byte | 7 |
| Packet Control Byte | 8 |
| Packet SubData Formats | 8 |
| Broadcast (0) | 8 |
| Broadcast Data Provider (3) | 8 |
| Routed Message (4) | 9 |
| Routed Read (5) | 10 |
| Routed Write (6) | 11 |
| Routed Response Data ACK (7) | 12 |
| Routed Response NAK (8) Timeout (9) Data Invalid (10) | 12 |
| Identify Request (17) | 12 |
| Identify Response (18) | 12 |
| Bind Request (19) | 13 |
| Bind Accept (20) | 13 |
| Module Read (21) | 14 |
| Module Write (22) | 14 |
| Module Response ACK (23) | 14 |
| Module Response NAK (24) | 14 |
| RSSI And CV | 14 |
| Bind Example Sequence | 16 |
| Module Parameters | 17 |
| Retries | 18 |
| Operation | 19 |
| Repeaters | 19 |
| Sleep & Sniff Mode (Radio module sleeping) | 19 |
| Wakeup Mode (Radio module waking another device) | 19 |
| Unslotted CSMA/CA | 20 |
| Power Saving Modes | 20 |
| Unique Addressing | 21 |
| LED | 21 |
| Diagnostics | 21 |
| Encryption | 21 |
| Channels | 21 |
| Link Quality | 21 |

| | |
|--------------------------------|----|
| Application Device..... | 22 |
| Max Data Lengths | 22 |
| Binding | 22 |
| FCC, ETSI and IC Aprovals..... | 22 |

Overview

The RAD24 module (28 X 16mm) is a general purpose radio transceiver with a serial USART interface. The module is intended to be coupled with a microcontroller. The module is available with an on-board chip antenna and external antennas.

The communication protocol has been designed for flexibility and has communication models for Request/Response and broadcast scenarios as well as built-in support for sleep/wake, binding, identification and diagnostics.

The Radio operates in the 2.4GHz band and utilises direct sequence spread spectrum. The modulation format is Offset Quadrature Phase Shift Keying (O-QPSK) with half-sine chip shaping. This is equivalent to MSK (Minimum Shift Keying) modulation.

16 channels operate over the 2.400 to 2.485 band.

Output power is 0db.

The module uses the Chipcon CC2430 radio transceiver chip and so leverages the 802.15.4 compliance of this chip but uses a non 802.15.4 compliant sync word and MAC Protocol Data Unit (MPDU) structure.

Designed to operate over 2.1 to 3.6 volts dc allowing for battery operation.

Hardware Specifications

Environmental

| Parameter | Minimum | Typical | Maximum | Units | Notes |
|-----------------------------|--------------|---------|------------|-------|-------|
| Supply voltage Range | 2.0 | 3 | 3.6 | Volts | |
| Sleep Current | - | 0.6 | | uA | |
| RX current | - | 22 | | mA | |
| TX current | - | - | 31 | mA | |
| Operating Temperature Range | -40 (-20) | - | 85 (55) | Deg C | |
| Storage Temperature Range | -40 | - | 85 | Deg C | |

Radio Specification 2.4Ghz

| Parameter | Minimum | Typical | Maximum | Units | Notes |
|-----------------------------|---------|---------------|---------|-------|--------|
| Frequency | 2.4 | - | 2.4835 | GHz | |
| Radio Bit Rate | - | 250 | - | Kbps | Note 1 |
| RX sensitivity | | -88 | | dbm | Note 2 |
| TX Output Power | -30 | | +1 | dbm | |
| Antenna matching impedance | 45 | 50 | 55 | Ohms | |
| Modulation method | | MSK (QPSK) | | | |
| Crystal Tolerance | | | +/-40 | ppm | Note 3 |
| Crystal ESR | | | 100 | Ohms | |
| Crystal Loading Capacitance | | | 16 | pF | |
| Range | - | - | 150 | M | 4 |

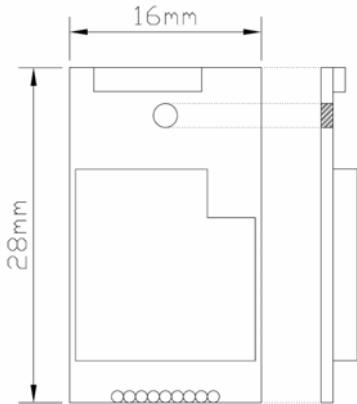
Notes

1. Device is capable 250Kbps.
2. For 250Kbps and MSK, 1% packet error rate, 20 bytes packet length, 540 kHz digital channel filter bandwidth
3. Includes Initial Tolerance + Ageing + Temperature. IE 26.000/10/10/10/16.
4. Based on Chip to PCB antenna in an open field site 3m above ground.

Physical

This module is a single sided constructed module using a 4 layer PCB.

The physical size is 16 x 28 x 3.5mm. Including integral antenna. Connection to application PCB is via surface mounting pads.



Communications Interface

The Hardware interface is 3V/3.3V logic level.

The communications UART configuration is 500000, 8 bit word, no Parity and 1 Stop bit.

Hardware Interfacing

| Pin | Direction | Radio Module |
|-----|-----------|--------------------------|
| 1 | ■ | 0V |
| 2 | ■ | +3V |
| 3 | → | RST (Reset in) |
| 4 | → | SPICK (SPI Serial Clock) |
| 5 | → | CTS (App is ready) |
| 6 | → | Srx (Serial RX) [SPI MO] |
| 7 | ← | RTS (Ready Out) |
| 8 | ← | Stx (Serial TX) [SPI MI] |
| 9 | ■ | 0V |

Data Transfer

To send radio data for transmission the main module will wait until the radio module **ready** line is active. Then the main module **ready** line will be set inactive and the data transmitted. The main module **ready** line will then become active once the last byte has gone out.

On receiving the data the radio module will drop its **ready** line and only enable it again when the data has been sent or in the case of requiring a response, activate again when the message has been sent back to the main module or the response has not arrived within the timeout period.

A no response will cause a response packet to be sent serially containing zero data and a null RSSI.

Data received with bad a checksum is just ignored by the main module.

If a response is required then the radio module will keep the **ready** line inactive until it has received the response or timed out.

When un-requested data arrives at the radio and if the packet is correctly addressed or broadcast then the data is sent to the main module by waiting until the **ready** line is active then dropping its own **ready** line then sending the data before raising its own **ready** line again.

Radio data arriving while the module is busy will be buffered and handled when ready again.

The radio module can be awakened by its CTS line.

Packet Protocol

Packet Format:

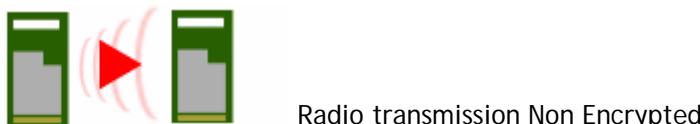
| Key | |
|--|---|
|  | = variable number of bytes |
|  | = Fixed function bytes required by module |
|  | = Generated by chip |

Format of packets to module, from module and transmitted.



| Length | Packet Type | Data |
|--------|-------------|------|
| | | |

= Length indicates number of bytes above



| PRE1 | PRE2 | PRE3 | PRE4 | SYNC1 | Length | Packet Control | Packet Type | Data | CRC1 | CRC2 |
|------|------|------|------|-------|--------|----------------|-------------|------|------|------|
| | | | | | | | | | | |

= Length indicates number of bytes above



| PRE1 | PRE2 | PRE3 | PRE4 | SYNC1 | Length | Packet Control | Len | C R C 1 | C R C 2 | Packet Type | Data | CRC1 | CRC2 |
|--------------------------|------|------|------|-------|--------|----------------|-----|------------------|------------------|-------------|------|------|------|
| 00 | 00 | 00 | 0A | E5 | | | | | | | | | |
| Encrypted Block | | | | | | | | | | | | | |
| Encryption Control Block | | | | | | | | | | | | | |
| | | | | | | | | | | Checksummed | | | |



| Length | Packet Type | Data | RSSI | CV |
|--------|-------------|------|------|----|
| | | | | |

= Length indicates number of bytes above

RSSI and CV bytes will only be appended to packets received via radio.

Packet Type Byte

The Packet control bytes indicate the type of packet and hold
The packet Type byte will be set by the instrument and passed to the module.

Packet Type

| b7 | b6 | B5 | B4 | b3 | b2 | b1 | b0 |
|-------|--------|-----------|----|----|----|----|-------------|
| Error | LoBatt | Broadcast | | | | | Packet Type |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | Function |
|-----------|--|
| Error | Bit indicated an error is present. This is set and reset by the device which will include this information in the packet sent to the module. |
| LoBatt | Bit indicated a low battery. This is set and reset by the device which will include this information in the packet sent to the module. |
| Broadcast | Used to indicate that a routed packet was broadcast so the receiver knows not to respond. |

| Value | Type | Description |
|-------|------------------------------|---|
| 0 | Broadcast | Broadcast packet |
| 3 | Broadcast Data Provider | Used to transfer data - See later Broadcast data |
| 4 | Routed Msg | Send message to specific ID - no response required. |
| 5 | Routed Read | Read data from a specific ID |
| 6 | Routed Write/Command | Write a value or execute a command to a specific ID |
| 7 | Routed Response ACK | Response - Acknowledged. May also contain data |
| 8 | Routed Response NAK | Response - Not Acknowledged |
| 9 | Routed Response Timeout | Response Timed out |
| 10 | Routed Response Data Invalid | Response - Data invalid |
| 17 | Identify Request | Broadcast to which all modules will reply with ID |
| 18 | Identify Response | Identify response |
| 19 | Bind Request | Request a link. Used for example to automate linking of devices. Module will now wait for 5 seconds and return the ID of the first device that responded or reply null. |
| 20 | Bind Accept | The response containing ID and data |
| 21 | Module Read | Read internal module parameter |
| 22 | Module Write | Write to an internal module parameter |
| 23 | Module ACK | Acknowledged internal write or return data |
| 24 | Module NAK | Command not acknowledged |

Packet Control Byte

This byte is only present in the radio packet and is generated by the module.

The sequence number increments as each device sends a packet except when retrying and repeating. This way a receiver will only react once to a packet even if received more than once. However if a second packet is received that requires a response it will be sent. But if the packet requested an action and that action has already been executed it will not execute again even though a response will be sent.

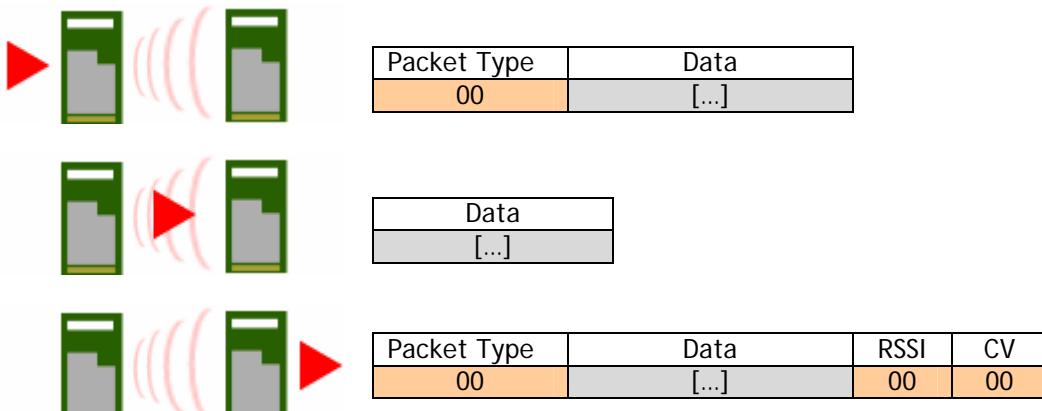
| b7 | b6 | B5 | B4 | b3 | b2 | b1 | b0 |
|----------|-----------|--------------|----|----|----|----|----|
| Repeated | Encrypted | Repeat Group | | | | | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit | Function |
|-----------------|---|
| Repeated | The byte is set when the packet is repeated so that packets are not repeated by repeaters. |
| Encrypted | Data includes extra 2 byte CRC checksum that must be checked after decryption. If a module uses an encryption key it will reject packets that are not encrypted and vice versa. |
| Repeat Group | Contains the repeat group code 0-3 which determines which packets are repeated by repeater modules. 0 is repeated by all. (See Repeater later) |
| Sequence Number | This increments for each packet sent but not when retries are made. |

Packet SubData Formats

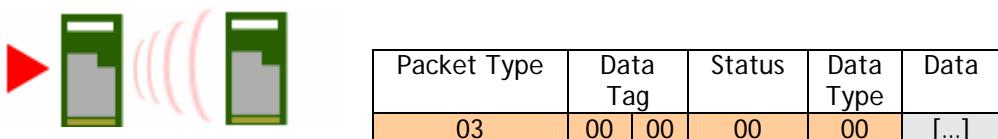
Depending on the packet type the data area will contain the following. Note that the radio data does not show the enclosing control bytes and CRC. The radio module in and out indicates the exact bytes involved.

Broadcast (0)



Simple broadcast packet which every module will receive. The source of the broadcast cannot be determined by receiving this packet.

Broadcast Data Provider (3)





| Data Tag | Status | Data Type | Data |
|----------|--------|-----------|------|
| 00 | 00 | 00 | 00 |
| [...] | | | |



| Packet Type | Data Tag | Status | Data Type | Data | RSSI | CV |
|-------------|----------|--------|-----------|------|-------|----|
| 03 | 00 | 00 | 00 | 00 | [...] | 00 |
| 00 | | | | | | |

This format will be used by Mantracourt products to allow Provider/Consumer ability to a range of products.
 2 byte data Tag is set in the provider and conditionally captured by the consumer.
 The datatype will support fixed data types of signed and unsigned 8, 16 and 32 bit values, float and string.
 The status will be device specific and only used when the consumer knows where the data is coming from or can be configured to use certain bits from this byte. i.e. a Display module may set an LED to mimic bit 4 from the status of a TSC data packet which will indicate overrange.
 The data will be of a type defined by datatype.

| Function | Display As | | | | | | | | Data Type |
|----------|------------|---|---|---|---|---|---|---|-----------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Sample | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | |

| Display As | Type | Description |
|------------|----------------------|---|
| 0 | Undefined | |
| 1 | Numeric | Numeric representation based on Data Type |
| 2 | Boolean | The data may be in any format but represents a boolean result where non zero numeric is True and string length > 1 or > 0 is True |
| 3 | Text | Can display as ascii text |
| 4 | Binary (unprintable) | Unprintable characters |
| 5 | Hex | Best represented as hex |
| 6 | Bit Map (10110101) | Each bit value should be shown |
| 7 | Percent | Numeric or string value has a value 0 - 100 |

| Data Type | Description | Size | Example | Note |
|-----------|--------------------|------|-------------|------|
| 0 | No content/unknown | 0 | | |
| 1 | UINT8 | 1 | 01 | |
| 2 | UINT16 | 2 | 00 01 | |
| 3 | INT32 | 4 | 00 00 00 01 | |
| 4 | Float | 4 | 3F 80 00 00 | |
| 5 | String | 0-64 | Hello World | |
| 6 | Binary | 0-64 | "E\$%^&*(" | |

The Display As bits should be used where possible as this can help in presenting the data for display purposes.

Status will where possible have the following bit meanings to allow handhelds and displays to gather common info.

| Status Byte | | | | | | | |
|-------------|---|---|---|---|-----------|---------------|-----------|
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| | | | | | Integrity | Digital Input | Shunt Cal |

Routed Message (4)



| Packet Type | To ID | | | Data | |
|-------------|-------|----|----|-------|--|
| 04 | 00 | 00 | 00 | [...] | |



| From ID | | | To ID | | | Data | |
|---------|----|----|-------|----|----|-------|--|
| 00 | 00 | 00 | 00 | 00 | 00 | [...] | |



| Packet Type | From ID | | | Data | | RSSI | CV |
|-------------|---------|----|----|-------|--|------|----|
| 04 | 00 | 00 | 00 | [...] | | 00 | 00 |

Routed Read (5)



| Packet Type | To ID | | | Command | |
|-------------|-------|----|----|---------|--|
| 05 | 00 | 00 | 00 | [...] | |



| From ID | | | To ID | | | Command |
|---------|----|----|-------|----|----|---------|
| 00 | 00 | 00 | 00 | 00 | 00 | 00 |



| Packet Type | From ID | | | Command | RSSI | CV |
|-------------|---------|----|----|---------|------|----|
| 05 | 00 | 00 | 00 | [...] | 00 | 00 |

Although the radio does not enforce the following, Mantracourt will adopt:
Where Command consists of

| Data Type | Data |
|-----------|-------|
| 00 | [...] |

The data will be of a type defined by Data Type.

| Function | Display As | | | | | | | | Data Type |
|----------|------------|---|---|---|---|---|---|---|-----------|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | |
| Sample | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | |

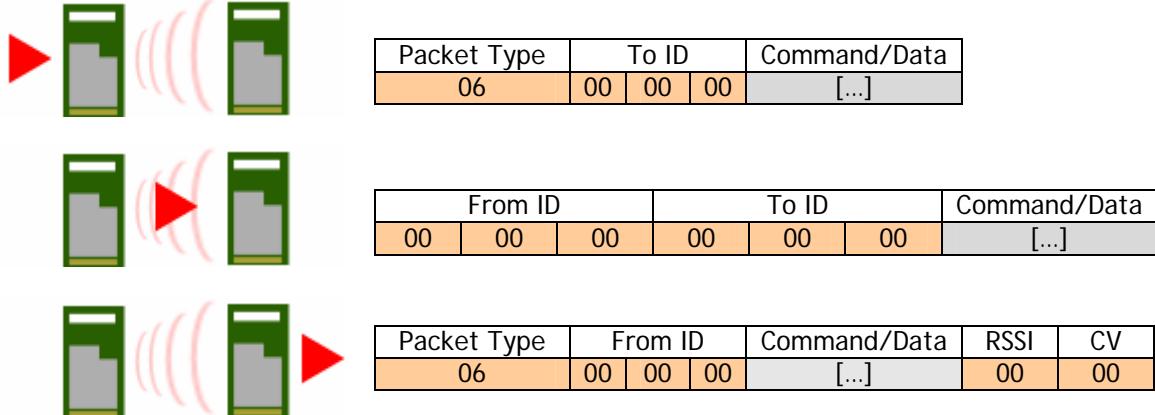
| Display As | Type | Description |
|------------|----------------------|---|
| 0 | Undefined | |
| 1 | Numeric | Numeric representation based on Data Type |
| 2 | Boolean | The data may be in any format but represents a boolean result where non zero numeric is True and string length > 1 or > 0 is True |
| 3 | Text | Can display as ascii text |
| 4 | Binary (unprintable) | Unprintable characters |
| 5 | Hex | Best represented as hex |
| 6 | Bit Map (10110101) | Each bit value should be shown |
| 7 | Percent | Numeric or string value has a value 0 - 100 |

| Data Type | Description | Size | Example | Note |
|-----------|--------------------|------|---------|------|
| 0 | No content/unknown | 0 | | |

| | | | | |
|---|--------|------|-------------|--|
| 1 | UINT8 | 1 | 01 | |
| 2 | UINT16 | 2 | 00 01 | |
| 3 | INT32 | 4 | 00 00 00 01 | |
| 4 | Float | 4 | 3F 80 00 00 | |
| 5 | String | 0-64 | Hello World | |
| 6 | Binary | 0-64 | "£\$%^&*(` | |

The Display As bits should be used where possible as this can help in presenting the data for display purposes.

Routed Write (6)



Although the radio does not enforce the following, Mantracourt will adopt:
Where Command/Data consists of:

| Command | Data Type | Data |
|---------|-----------|-------|
| 00 | 00 | [...] |

The data will be of a type defined by Data Type.

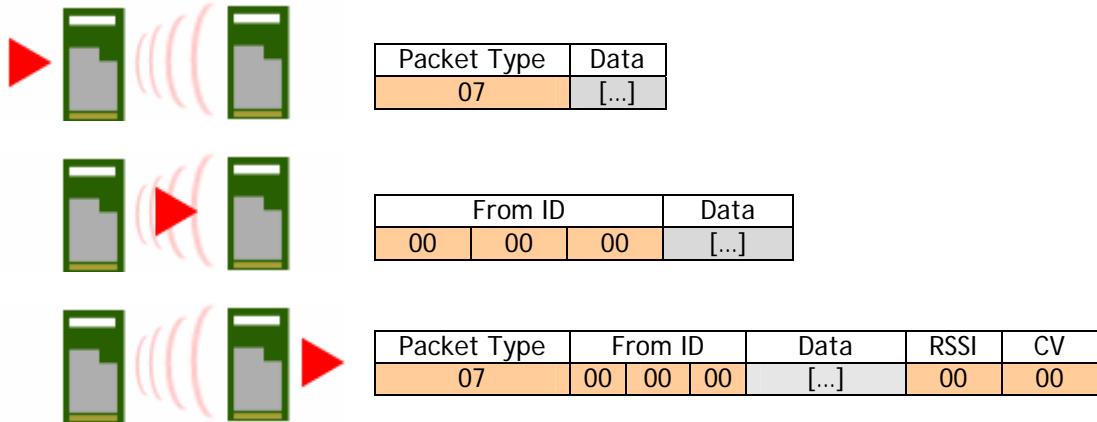
| Function | Display As | | | | | | | | Data Type | |
|----------|------------|---|---|---|---|---|---|---|-----------|--|
| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 | | |
| Sample | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | | |

| Display As | Type | Description | |
|------------|----------------------|---|--|
| 0 | Undefined | | |
| 1 | Numeric | Numeric representation based on Data Type | |
| 2 | Boolean | The data may be in any format but represents a boolean result where non zero numeric is True and string length > 1 or > 0 is True | |
| 3 | Text | Can display as ascii text | |
| 4 | Binary (unprintable) | Unprintable characters | |
| 5 | Hex | Best represented as hex | |
| 6 | Bit Map (10110101) | Each bit value should be shown | |
| 7 | Percent | Numeric or string value has a value 0 - 100 | |

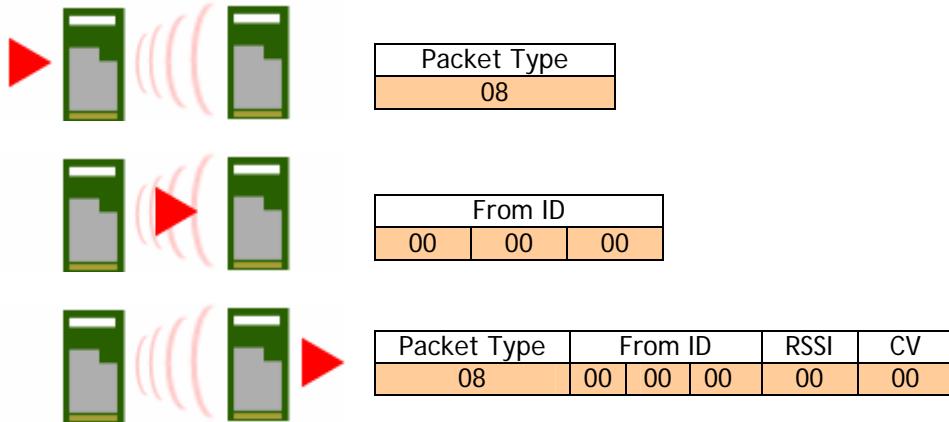
| Data Type | Description | Size | Example | Note |
|-----------|--------------------|------|-------------|------|
| 0 | No content/unknown | 0 | | |
| 1 | UINT8 | 1 | 01 | |
| 2 | UINT16 | 2 | 00 01 | |
| 3 | INT32 | 4 | 00 00 00 01 | |
| 4 | Float | 4 | 3F 80 00 00 | |
| 5 | String | 0-64 | Hello World | |
| 6 | Binary | 0-64 | "£\$%^&*(` | |

The Display As bits should be used where possible as this can help in presenting the data for display purposes.

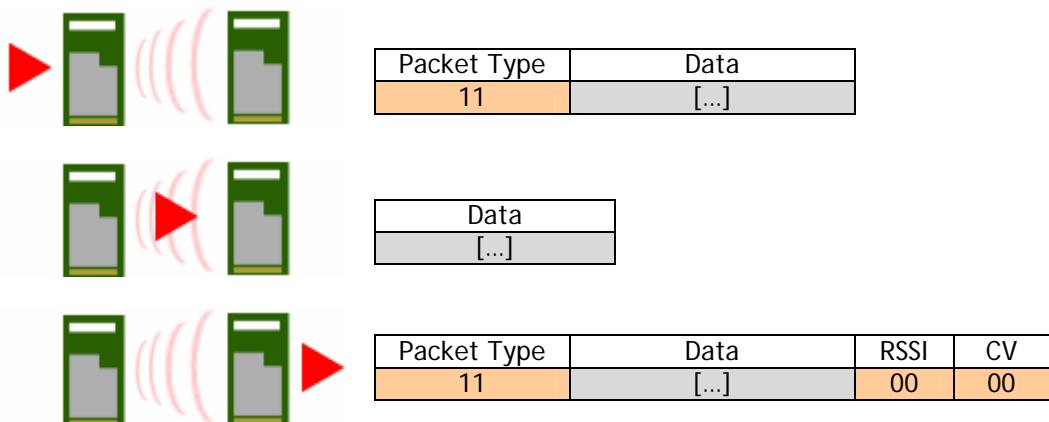
Routed Response Data ACK (7)



Routed Response NAK (8) / Timeout (9) / Data Invalid (10)

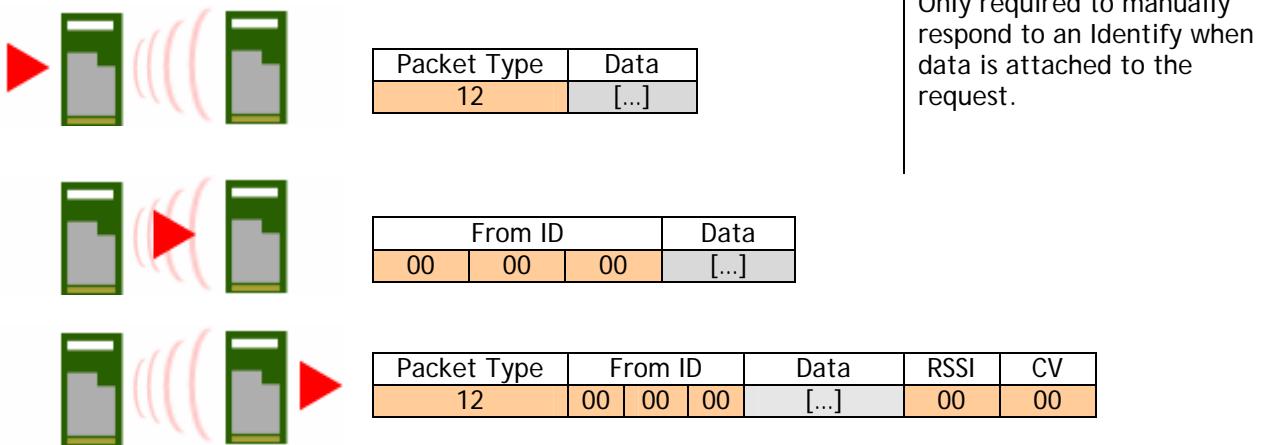


Identify Request (17)



The identify request may include optional data with which the responder may decide whether to respond or not. All devices will respond when there is no data.

Identify Response (18)



The responder may optionally provide data which may be dependant on the data supplied during the request.

Bind Request (19)

| Packet Type | Data Tag | Direction | Config | Duration (optional) |
|-------------|----------|-----------|--------|---------------------|
| 13 | 00 00 | 00 | 00 | 00 |



Data Tag: Use zeros if not required.

Direction: 0=remote to local

1=local to remote

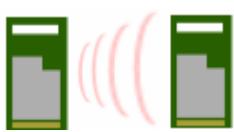
Config: When set will indicate the remote should be able to be configured. i.e. exit any low power modes.

If Direction is one the Channel and Key will **not** be sent in the Bind Set packet and the local module will bind to the remote channel and key.



| Packet Type | To ID | Data Tag | Config | Channel | Key | RSSI | CV |
|-------------|----------|----------|--------|---------|----------|------|----|
| 13 | 00 00 00 | 00 00 | 00 | 00 | [..16..] | 00 | 00 |

Sent only if Bind Accept received. Channel and Key are only sent when binding remote to local settings.



| Packet Type | Data Tag | Config | RSSI | CV |
|-------------|----------|--------|------|----|
| 13 | 00 00 | 00 | 00 | 00 |

Sending the serial packet to the module puts it into listen mode on channel 16 with secret fixed key for 2 seconds or the optional duration in seconds.

If a PT_BIND_ACCEPT is received within this time the PT_BIND_REQUEST response is sent over radio which optionally includes the channel and key to change to which are the current key and channel of this radio module. Once transmitted or the time period has elapsed the module reverts to normal channel and operation. No response is expected or waited for. A PT_BIND_ACCEPT serial packet is sent so the requester can see the ID and data tag.

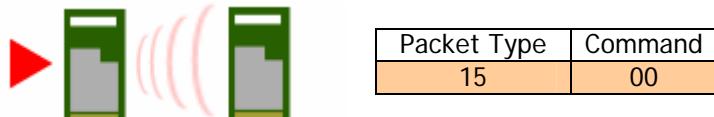
Bind Accept (20)

| | |
|-------------|----------|
| Packet Type | Data Tag |
| 14 | 00 00 |

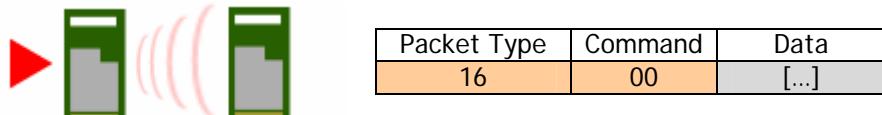
| | | | | | | | | | | | | | | | |
|-------------|--|-------------|----------|----------|---------|----------|-------|----------|----------|----------|-------|----|----------|----|----|
| | <table border="1"> <tr> <td>From ID</td><td>Data Tag</td><td>Channel</td><td>Key</td></tr> <tr> <td>00 00 00</td><td>00 00</td><td>00</td><td>[..16..]</td></tr> </table> | From ID | Data Tag | Channel | Key | 00 00 00 | 00 00 | 00 | [..16..] | | | | | | |
| From ID | Data Tag | Channel | Key | | | | | | | | | | | | |
| 00 00 00 | 00 00 | 00 | [..16..] | | | | | | | | | | | | |
| | <table border="1"> <tr> <td>Packet Type</td> <td>From ID</td> <td>Data Tag</td> <td>Channel</td> <td>Key</td> <td>RSSI</td> <td>CV</td> </tr> <tr> <td>14</td> <td>00 00 00</td> <td>00 00</td> <td>00</td> <td>[..16..]</td> <td>00</td> <td>00</td> </tr> </table> | Packet Type | From ID | Data Tag | Channel | Key | RSSI | CV | 14 | 00 00 00 | 00 00 | 00 | [..16..] | 00 | 00 |
| Packet Type | From ID | Data Tag | Channel | Key | RSSI | CV | | | | | | | | | |
| 14 | 00 00 00 | 00 00 | 00 | [..16..] | 00 | 00 | | | | | | | | | |
| | <table border="1"> <tr> <td>Packet Type</td> <td>From ID</td> <td>Data Tag</td> <td>RSSI</td> <td>CV</td> </tr> <tr> <td>14</td> <td>00 00 00</td> <td>00 00</td> <td>00</td> <td>00</td> </tr> </table> | Packet Type | From ID | Data Tag | RSSI | CV | 14 | 00 00 00 | 00 00 | 00 | 00 | | | | |
| Packet Type | From ID | Data Tag | RSSI | CV | | | | | | | | | | | |
| 14 | 00 00 00 | 00 00 | 00 | 00 | | | | | | | | | | | |

This is sent by a device wishing to be bound so can be triggered from a button or on power cycle. The radio switches to channel 16 with a secret fixed key and transmits the above packet. It then waits 50mS for a PT_BIND_REQUEST packet. If it receives no response then it will revert to normal channel and key. If it does receive PT_BIND_REQUEST then the channel and key will be set to that contained in the packet (if it is present) and the settings saved to flash. The PT_BIND_REQUEST packet will then be sent serially. Operation will then continue as normal.

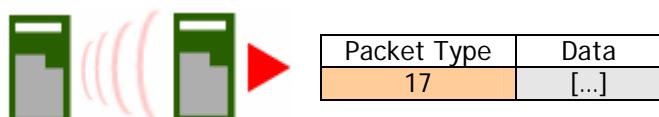
Module Read (21)



Module Write (22)



Module Response ACK (23)



Module Response NAK (24)



RSSI And CV

RSSI can be calculated from the RSSI byte which is stored in 2's compliment format. To convert the binary value to RSSI use the following algorithm.

```
If RSSI > 127 Then RSSI = (RSSI - 1 Xor 0xFF) * -1
RSSI = RSSI - 45
```

The binary value of CV (0-255) can be directly used where a poor CV is around 55 and a good CV is 110.

Mantracourt may also refer to Link Quality which is derived from the RSSI and CV values:

```
LQI = (((94 + RSSI) + (CV - 55)) / 2) * 3.9
```

Which should give a total usable range of 0 to 255.

If a link quality is to be represented to an end user then from this range only portion 50 to 128 should represent 0-100% quality.

See Module Parameters for command table and structures.

Bind Example Sequence

|  | <table border="1"> <thead> <tr> <th>Packet Type</th><th>Data Tag</th><th>Direction</th><th>Config</th><th>Duration (optional)</th></tr> </thead> <tbody> <tr> <td>13</td><td>00 00</td><td>00</td><td>00</td><td>00</td></tr> </tbody> </table> <p>Handheld enters Bind Request mode via serial packet. Handheld does nothing until timeout or receipt of a Bind Accept packet via radio.</p> | Packet Type | Data Tag | Direction | Config | Duration (optional) | 13 | 00 00 | 00 | 00 | 00 | | | | | | |
|---|--|-------------|----------|---------------------|----------|---------------------|-------|----------|----------|----------|----------|-------|----------|----|----------|----|----|
| Packet Type | Data Tag | Direction | Config | Duration (optional) | | | | | | | | | | | | | |
| 13 | 00 00 | 00 | 00 | 00 | | | | | | | | | | | | | |
|  | <table border="1"> <thead> <tr> <th>Packet Type</th><th>Data Tag</th></tr> </thead> <tbody> <tr> <td>14</td><td>00 00</td></tr> </tbody> </table> <p>Remote device issues a Bind Accept packet via serial</p> | Packet Type | Data Tag | 14 | 00 00 | | | | | | | | | | | | |
| Packet Type | Data Tag | | | | | | | | | | | | | | | | |
| 14 | 00 00 | | | | | | | | | | | | | | | | |
|  | <table border="1"> <thead> <tr> <th>From ID</th><th>Data Tag</th><th>Channel</th><th>Key</th></tr> </thead> <tbody> <tr> <td>00 00 00</td><td>00 00</td><td>00</td><td>[..16..]</td></tr> </tbody> </table> <p>The Bind Accept packet is sent via radio.</p> | From ID | Data Tag | Channel | Key | 00 00 00 | 00 00 | 00 | [..16..] | | | | | | | | |
| From ID | Data Tag | Channel | Key | | | | | | | | | | | | | | |
| 00 00 00 | 00 00 | 00 | [..16..] | | | | | | | | | | | | | | |
|  | <table border="1"> <thead> <tr> <th>Packet Type</th><th>From ID</th><th>Data Tag</th><th>Channel</th><th>Key</th><th>RSSI</th><th>CV</th></tr> </thead> <tbody> <tr> <td>14</td><td>00 00 00</td><td>00 00</td><td>00</td><td>[..16..]</td><td>00</td><td>00</td></tr> </tbody> </table> <p>The handheld receives this packet</p> | Packet Type | From ID | Data Tag | Channel | Key | RSSI | CV | 14 | 00 00 00 | 00 00 | 00 | [..16..] | 00 | 00 | | |
| Packet Type | From ID | Data Tag | Channel | Key | RSSI | CV | | | | | | | | | | | |
| 14 | 00 00 00 | 00 00 | 00 | [..16..] | 00 | 00 | | | | | | | | | | | |
|  | <table border="1"> <thead> <tr> <th>Packet Type</th><th>To ID</th><th>Data Tag</th><th>Config</th><th>Channel</th><th>Key</th><th>RSSI</th><th>CV</th></tr> </thead> <tbody> <tr> <td>13</td><td>00 00 00</td><td>00 00</td><td>00</td><td>00</td><td>[..16..]</td><td>00</td><td>00</td></tr> </tbody> </table> <p>The handheld transmits a Bind Request packet.</p> | Packet Type | To ID | Data Tag | Config | Channel | Key | RSSI | CV | 13 | 00 00 00 | 00 00 | 00 | 00 | [..16..] | 00 | 00 |
| Packet Type | To ID | Data Tag | Config | Channel | Key | RSSI | CV | | | | | | | | | | |
| 13 | 00 00 00 | 00 00 | 00 | 00 | [..16..] | 00 | 00 | | | | | | | | | | |
|  | <table border="1"> <thead> <tr> <th>Packet Type</th><th>From ID</th><th>Data Tag</th><th>RSSI</th><th>CV</th></tr> </thead> <tbody> <tr> <td>14</td><td>00 00 00</td><td>00 00</td><td>00</td><td>00</td></tr> </tbody> </table> <p>The handheld constructs a Bind Accept serial packet to inform the application of the remote device that accepted.</p> | Packet Type | From ID | Data Tag | RSSI | CV | 14 | 00 00 00 | 00 00 | 00 | 00 | | | | | | |
| Packet Type | From ID | Data Tag | RSSI | CV | | | | | | | | | | | | | |
| 14 | 00 00 00 | 00 00 | 00 | 00 | | | | | | | | | | | | | |
|  | <table border="1"> <thead> <tr> <th>Packet Type</th><th>To ID</th><th>Data Tag</th><th>Config</th><th>Channel</th><th>Key</th><th>RSSI</th><th>CV</th></tr> </thead> <tbody> <tr> <td>13</td><td>00 00 00</td><td>00 00</td><td>00</td><td>00</td><td>[..16..]</td><td>00</td><td>00</td></tr> </tbody> </table> <p>The remote device receives the Bind Request packet and because the channel and key are present it changes radio settings to match.</p> | Packet Type | To ID | Data Tag | Config | Channel | Key | RSSI | CV | 13 | 00 00 00 | 00 00 | 00 | 00 | [..16..] | 00 | 00 |
| Packet Type | To ID | Data Tag | Config | Channel | Key | RSSI | CV | | | | | | | | | | |
| 13 | 00 00 00 | 00 00 | 00 | 00 | [..16..] | 00 | 00 | | | | | | | | | | |
|  | <table border="1"> <thead> <tr> <th>Packet Type</th><th>Data Tag</th><th>Config</th><th>RSSI</th><th>CV</th></tr> </thead> <tbody> <tr> <td>13</td><td>00 00</td><td>00</td><td>00</td><td>00</td></tr> </tbody> </table> <p>The remote device now formats a serial Bind Request packet for the application which may use the Data Tag. It should also attempt to enter config mode if the config byte is set.</p> | Packet Type | Data Tag | Config | RSSI | CV | 13 | 00 00 | 00 | 00 | 00 | | | | | | |
| Packet Type | Data Tag | Config | RSSI | CV | | | | | | | | | | | | | |
| 13 | 00 00 | 00 | 00 | 00 | | | | | | | | | | | | | |

Module Parameters

* = Factory set / only available under factory control

| Parameter / Command | Number | Bytes / Format | Data Example |
|-------------------------|--------|---|----------------------------|
| Factory Level | | | |
| CMD_VERMAJOR* | 1 | 1 byte unsigned char | 01 |
| CMD_VERMINOR* | 2 | 1 byte unsigned char | 01 |
| CMD_ID* | 3 | 3 bytes | 00 00 01 |
| CMD_PRODUCT_ID* | 4 | 1 byte unsigned char | 00 |
| CMD_FUNCTION_ID* | 5 | 1 byte unsigned char | 00 |
| User Level | | | |
| CMD_NAME | 10 | 12 bytes Must include null terminator | Station1<00> |
| CMD_CHANNEL | 11 | 1 byte unsigned char | 00 |
| CMD_POWER | 12 | 1 byte unsigned char | 00 |
| CMD_REPEATERS | 13 | 1 byte unsigned char | 00 |
| CMD_REPEATERS_GROUP | 14 | 1 byte unsigned char | 00 |
| CMD_ENCRYPTION_KEY | 15 | 16 bytes | |
| CMD_WAKE_CHECK_INTERVAL | 16 | 2 bytes unsigned int | 00 00 |
| CMD_WAKER_DURATION | 17 | 2 bytes unsigned int | 00 00 |
| CMD_USE_CSMA | 18 | 1 byte unsigned char | 00 |
| CMD_ALWAYS_WAKE | 19 | 1 byte unsigned char | 00 |
| CMD_QUALITY | 20 | 2 bytes RSSI Level and CV (chipcon format) | 00 00 |
| CMD_RSSI | 21 | 1 bytes unsigned int -70 db represented by 70 | 00 |
| Commands | | | |
| CMD_WAKER_MODE | 22 | 4 bytes | 00 00 01 FF |
| CMD_SLEEP | 23 | 1 bytes (5 bytes) 0=Sleep only 1=Sleep with wake check at WakeCheckInterval 2=Sleep for SleepPeriod then wake where sleep period is specified by following 4 bytes | 01 Or 02 00 00 00 FF |
| CMD_SAVE_FLASH | 24 | 0 bytes | |
| CMD_RESET | 25 | 0 bytes | |
| CMD_SPECTRUM | 26 | 1 byte 0=Turn off spectrum mode 1=Turn on spectrum mode When activated the radio will no longer communicate except serially and will supply Extended Read packets containing spectrum analyser data. | 01 |
| Test Modes | | | |
| CMD_TX_TEST_MODE* | 27 | 1 byte 0=1000 packets 1=Un-modulated carrier 2=Modulated carrier | |
| CMD_START_RX_TEST_MODE* | 28 | 0 bytes | |
| CMD_STOP_RX_TEST_MODE* | 29 | 0 bytes | |
| Diagnostics | | | |
| CMD_LED_MODE | 34 | 1 byte 0=Normal Flash 1=On when awake 2=Radio TX 3=Radio RX | 02 |

| | | | |
|--|--|----------|--|
| | | 4=Serial | |
|--|--|----------|--|

Retries

Retries are made on packet types 5 and 6. After a timeout period where no response is forthcoming the packet will be resent without changing the sequence number but setting the Retry bit. Three attempts will be made. The same sequence number stops a device from reacting twice to a retried packet.

Operation

Repeaters

Each Radio module will have the ability to act as a repeater. The module will support the following parameters to enable routing.

| | |
|--------------------|---|
| Repeater | Set whether the module will act as a repeater. |
| RepeatGroup | Set the group code for packets to repeat. Zero will be repeated by all repeaters but packets with non zero will only be repeated when the repeater is set to the matching code. |

If a packet is received at the module that is not specifically addressed to that module and the repeated bit indicator is not set then if the packet matches **RepeaterGroup** code or is zero then the repeated bit will be set and the packet will be repeated without altering anything except the CRC values.

Broadcast packets and all packets that are not routed will always be repeated whenever **Repeater** is non enabled.

There will be no hold off period on repeated packets. As soon as clear channel assessment reports clear the packet will be resent.

Note that only one repeater may be involved with a packets journey. If we allow more repeater hops then the traffic will increase and we will have to configure devices to let them know they are in a multi repeater environment so as to increase timeouts. Also packet structures would have to include extra routing info and routers will have to be identified with a unique code.

With one repeater we can work transparently and remember that we can increase range or avoid obstacles and also install multiple repeaters to increase coverage but cannot use more than one repeater to carry a packet over multiple hops.

Sleep & Sniff Mode (Radio module sleeping)

The radio module can enter sleep/sniff mode via an external or internal command and will wake at intervals set in its flash variables (radioint).

Board will wake at this interval and power up radio and send a wake request as broadcast using the packet Type **WakeReq**.

The module will then wait for a set period to see if it receives a packet marked as **WakeUp** type. The **WakeUp** packet must be addressed to this ID.

If the radio module detects a **WakeUp** command the **SCIk** line will activate which will wake the main device.

NOTE: The time to wait for a response will be length of 100 (or max) byte packet + 320uS. So wait for 5mS.

NOTE: In repeater environments this may double the time spent transmitting the wakeup command but should still fit in to 5mS.

680uS to transmit. Repeater picks up and retransmits with no CSMA delay $450\mu\text{s} \times 2 = 900\ \mu\text{s}$. Now a 100 byte packet appears after $320\mu\text{s} + 7900\mu\text{s}$ now response $680\mu\text{s} + 900\mu\text{s}$.

Total = 9.8mS. We should wait 10 mS in repeater environment.

Wakeup Mode (Radio module waking another device)

To wake a device the wake 'command' is now executed on the waker. In the main module a parameter called **WAKE** will accept a 6 char ID. On writing an ID to this parameter the device will go into wake mode. I.e. **Wake='FFE123'**

The radio module will spend a duration, set in flash, to listen for a WakeRequest type packet from either the specified device or any device if the Wake target is broadcast ID.

If the WakeRequest is received a response packet is sent of type WakeUp.

The radio module ready line will return to ready after the data has been received from the main module. I.e. this mode is asynchronous and the module is ready to handle other tasks. Being in wake mode really means that for the duration any packet received marked as WakeReq will be tested to see if it should be responded to. At the end of the duration or when a response to wake has been set the internal WakeState variable will be updated.

A WakeState parameter will also be supported that will query the radio module for the state of the last Wake procedure. This will return 0=Failed 1=OK 2=Busy

A WakeCancel command will cancel an in-progress wake mode.

Unslotted CSMA/CA

Algorithm is based on backoff periods where one backoff period is equal to 20 symbols (10 bytes).

NB = Number of backoffs

BE = Backoff Exponent

NB is the number of times the algorithm was required to back off whilst attempting the current transmission, this is initialised to zero. BE is the backoff exponent and is related to how many backoff periods a device shall wait before attempting to assess the channel.

BE is either 0 when CSMA is disabled or 2 when enabled. unslotted CSMA/CA algorithm works as follows:

1. NB = 0, BE = 2
2. Delay for random period in range 1 to $(2^{BE} - 1)$ backoff periods
3. Perform CCA
4. if channel is idle then transmit data otherwise go to step 5
5. NB = NB + 1, BE = min(BE + 1, 3)
6. if NB > 1000 transmission has failed (Channel Access Error) otherwise goto step 2.

The time between requesting that a data frame is transmitted and it actually being transmitted is therefore dependant upon the backoff period and the result of the CCA. The time taken to complete a single backoff period can be calculated as follows:

$$\text{BackoffPeriod (ms)} = [(2^{BE} - 1) * 20] / 62.5$$

One backoff period = 320uS

When CSMA is enabled the random backoff time will vary from 320uS to 960uS for a first try then 320uS to 2.24mS for subsequent tries so there will always be at least a 320uS gap between CSMA packets. This will allow for sleep/wake or response packets, which do not use CSMA, to slot in as soon as CCA indicates clear.

So possible backoff periods for each BE is as follows:

| BE/RND | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|--------|---|-------|-------|-------|--------|---------|---------|---------|
| 0 | 0 | 320uS | | | | | | |
| 1 | 0 | 320uS | 640uS | | | | | |
| 2 | 0 | 320uS | 640uS | 960uS | 1.28mS | | | |
| 3 | 0 | 320uS | 640uS | 960uS | 1.28mS | 1.600mS | 1.920mS | 2.240mS |

Power Saving Modes

Via comms it must be possible to set the radio into various levels of sleep and to be able to wake via interrupt using one of the handshaking lines.

| Mode | Description |
|---------------------|--|
| Deep Sleep | Radio module is effectively off. Can be woken with CTS line or comms. |
| Deep Sleep + Period | Asleep for preset period. Will fully awake after period elapses. |
| Deep Sleep Wakable | Radio is asleep but wakes periodically to request whether it should fully awake. |
| Awake | Fully awake. |

Unique Addressing

Each radio module will be allocated a unique 3 byte address allocated at manufacture.

LED

The module will have a low current LED fitted.

In normal operation (non sleep modes) this LED will flash once a second. This will enable visual indication of operation.

Where the module has other operational modes such as packet monitor or repeater the LED may flash differently to indicate this.

Also the module may be put into diagnostic modes where the LED may flash on packet receive or transmit for example.

Diagnostics

The modules could enter diagnostic modes whereby they perform a lot of checks to minimise requirements externally at test time.

Encryption

The module will support optional 128 bit AES encryption. This will be used to isolate groups of devices if required.

The encryption is either disabled or enabled with a 128 bit key. When enabled the Packet Control byte indicates this with a special bit.

Encryption acts on the Packet Type Byte, 3 byte encryption header and the data part of the packet

The Encryption Header data consists of 3 bytes at the start of the data . The first byte is the length of the unencrypted data followed by the 2 byte CRC value of that length of data.

When decoding encrypted data the Packet Type Byte, Encryption Header and the data (including any additional padding added by the encryption process) are decrypted using the module key.

The data is used to generate a 2 byte CRC value using the length of data (original length) as specified by the Length Byte in the Encryption Header and if this matches the CRC bytes in the Encryption Header then the packet is accepted.

Channels

Channels can be programmed from 2400 to 2483.5 MHz. We will use 5 MHz channels giving 16 possible.

Channels 1 to 16.

Frequency = 2405 + 5 (Channel-1) MHz, k=1 to 16

Link Quality

RSSI and Correlation Value data will be present in each packet from the radio. These values are useful in their own right.

Further link quality can be calculated by correlation Value and RSSI level.

This value could be passed via radio or serial and gives an LQI for the last received packet.

LQI is calculated as follows:

$$\text{LQI} = (((95 + \text{RSSI}) + (\text{CV} - 80) * 2)) * 2$$

Where RSSI has already been converted to dB i.e. 2s compliment taken into account and -45 added.

This will be calculated when LQI is asked for and results in a value from 0 to 255.

Application Device

When the module is used as part of another device and that device supports parameters that affect the module, such as power level and channel etc, then the application device would need to read these parameters from the radio module at startup and write to it when its own parameters are updated.

Max Data Lengths

The max Data Length byte value is 127 which includes the CRC bytes giving a total of 125 bytes. The Packet Control byte reduces this to 124 bytes.

So with no encryption the max size of packet in via serial is 124 bytes which will include the Packet Type byte. Encryption requires a further 3 bytes so packet size would be 121 bytes.

When using encryption this will be reduced further because of the 16 byte encryption block minimum so the max length would be 105. Probably best to limit to 101 bytes. Note that over serial this will allow 100 data bytes (in broadcast) and 1 length byte.

Binding

Using the built in binding process we have removed the need for a reset to defaults mechanism for comms. Binding can be used to change the channel and encryption key of any device to match another. The bind also gives details of ID and default Data Tag so coordinating comms settings a more intimate data bind may take place.

Some applications will bind using switches but inaccessible items such as T24-SA will send a Bind Request on power cycle.

FCC, ETSI and IC Aprovals

Family: RAD24

Models: i and e for internal and external antenna variants.

IC:7224A-RAD24

FCC ID:VHARAD24

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

