Radiometrix

Hartcran House, 231 Kenton Lane, Harrow, HA3 8RP, England

Tel: +44 (0) 20 8909 9595, Fax: +44 (0) 20 8909 2233

20 April 2007

RPC2A

UHF Radio Packet Controller

Modules: RPC2A-433-64: IC+BiM2A-433-64-S

IC's:	***RPC-000-DIL:***	***18 pin DIL IC***
	RPC-000-SO:	***18 pin SO IC***
	RPC-000-SS:	***20 pin SSOP IC***

The RPC2A-433-64 is intelligent transceiver modules, which enable a radio network/link to be simply implemented between a number of digital devices. The module combines a UHF radio transceiver and a 64kbps packet controller.

The RPC2A-module

- Crystal controlled PLL FM circuitry for both Tx and Rx
- Reliable 75 meter in-building range, 300m open ground
- Built-in self-test / diagnostics / status LED's
- Complies with ETSI EN 300 220-3
- Complies with ETSI EN 301 489-3
- Single 5V supply @ < 27mA
- 64kbps half duplex
- Free format packets of 1 - 27 bytes
- Packet framing and error checking are user transparent
- Collision avoidance (listen before transmit)
- Direct interface to 5V CMOS logic
- Power save mode

INTRODUCTION

The RPC2A is an enhanced replacement for original RPC-433-40 transceiver. It is a self-contained plug-on radio port which requires only a simple antenna, 5V supply and a byte-wide I/O port on a host microcontroller (or bi-directional PC port).
The module provides all the RF circuits and processor intensive low level packet formatting and packet recovery functions required to inter-connect an number of microcontrollers in a radio network.

A data packet of 1 to 27 bytes downloaded by a Host microcontroller into the RPC2A's packet buffer is transmitted by the RPC2A's transceiver and will "appear" in the receive buffer of all the RPC2A's within radio range.

A data packet received by the RPC2A's transceiver is decoded, stored in a packet buffer and the Host microcontroller signalled that a valid packet is waiting to be uploaded.

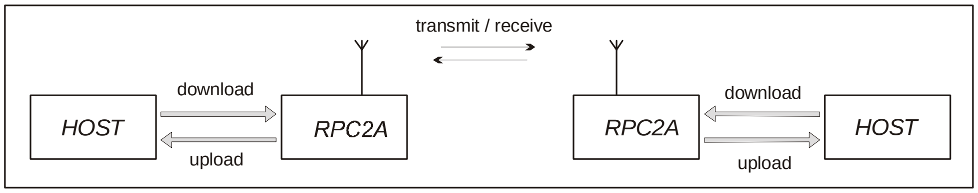


figure 1: RPC2A + Host μ-controller

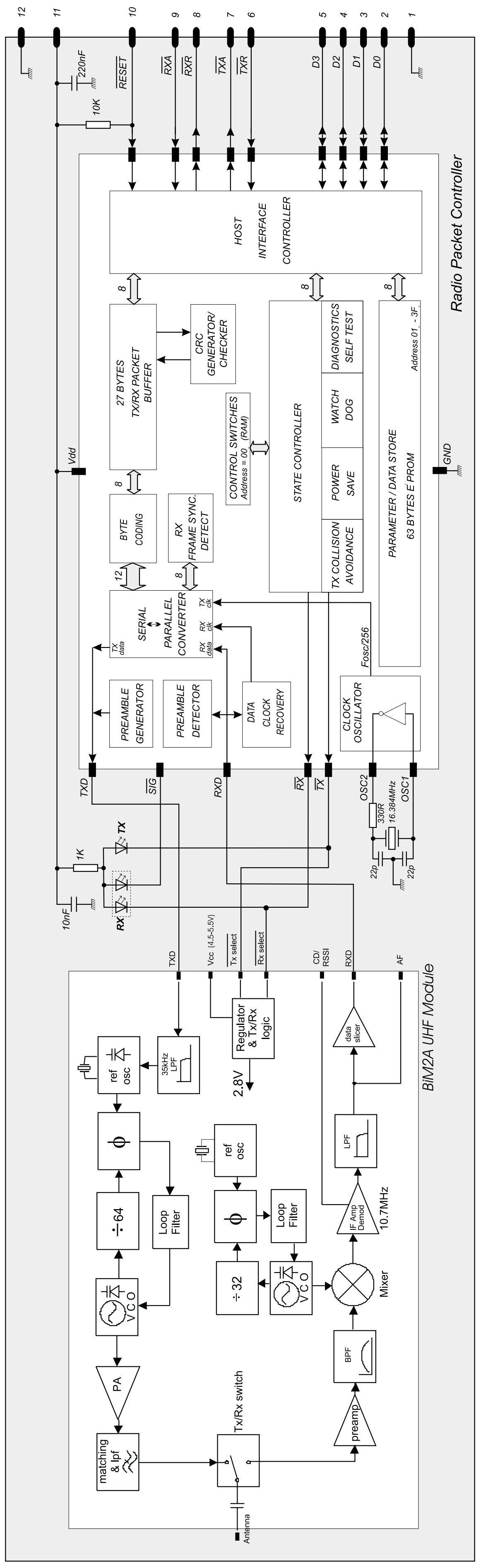


figure 2: RPC2A block diagram

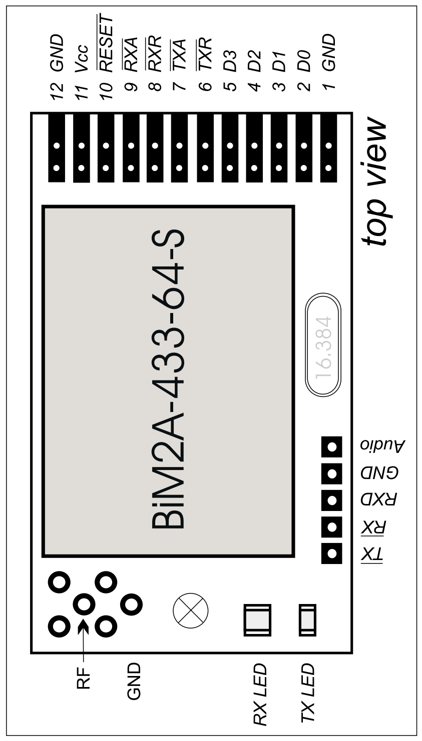
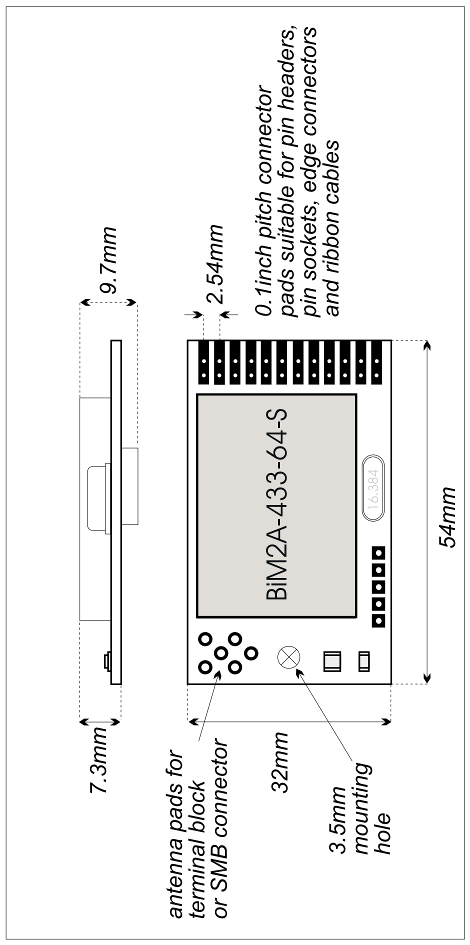


figure 4: RPC2A's pinout



9.7mm
2.54mm
0.1inch pitch connector pads suitable for pin headers, pin sockets, edge connectors and ribbon cables
BiM2A-433-64-S
16.384
54mm
7.3mm
antenna pads for terminal block or SMB connector
32mm
3.5mm mounting hole

figure 3: physical dimensions

1. FUNCTIONAL DESCRIPTION

On receipt of a packet downloaded by the Host, the RPC2A will append to the packet: Preamble, start byte and a error check code. The packet is then coded for security and mark:space balance and transmitted through the BiM2A Transceiver as a 64kbps synchronous stream. One of four methods of collision avoidance (listen before TX) may be user selected.

When not in transmit mode, the RPC2A continuously searches the radio noise for valid preamble. On detection of preamble, the RPC2A synchronises to the in-coming data stream, decodes the data and validates the check sum. The Host is then signalled that a valid packet is waiting to be unloaded. The format of the packet is entirely of the users determination except the 1st byte (the Control Byte) which must specify the packet type (control or data) and the packet size. A valid received packet is presented back to the host in exactly the same form as it was given.

To preserve versatility, the RPC2A does not generate routing information (i.e. source/ destination addresses) nor does it handshake packets. These network specific functions should be performed by the host.

Additional features of the RPC2A include extensive diagnostic/debug functions for evaluation and debugging of the radio and host driver software, a built in self test function and a sleep mode / wake-up mechanism which may be programmed to reduce the average current to less than 100µA. The operating parameters are fully programmable by the host and held in EEPROM, the host may also use the EEPROM as a general purpose non-volatile store for addresses , routing information etc.

1.1 OPERATING STATES

The RPC2A has four normal operating states:

- *IDLE / SLEEP*
- *HOST TRANSFER*
- *TRANSMIT*
- *RECEIVE*

IDLE/SLEEP

The *IDLE* state is the quiescent/rest state of the RPC2A. In *IDLE* the RPC2A enables the receiver and continuously searches the radio noise for message preamble. If the power saving modes have been enabled the RPC2A will pulse the receiver on, check for preamble and go back to *SLEEP* if nothing is found. The 'ON' time is 5ms, OFF time is programmable in the RPC2A's EEPROM and can vary between 22ms and 2.9s. The TX Request line from the Host is constantly monitored and will be acted upon if found active (low). A TX Request will immediately wake the RPC2A up from *SLEEP* mode.

HOST TRANSFERS

If the host sets the TX Request line low a data transfer from the Host to the RPC2A will be initiated. Similarly the RPC2A will pull RX Request low when it requires to transfer data to the Host (this may polled or used to generate a Host interrupt).

The transfer protocol is fully asynchronous, i.e. the host may service another interrupt and then continue with the RPC2A transfer. It is desirable that all transfers are completed quickly since the radio transceiver is disabled until the Host <> RPC2A transfer is completed. Typically a fast host can transfer a 27 byte packet to / from the RPC2A in under 1ms.

Transmit

On receipt of a data packet from the host, the RPC2A will append to the packet - preamble, frame sync byte and an error check sum. The packet is then coded for mark:space balance and transmitted. A full 27 byte packet is transmitted in 8.1ms of TX air time (64kb/s + 5ms preamble).

Collision avoidance (Listen Before Transmit-LBT) functions can be enabled to prevent loss of packets.

Data packets may be sent with either normal or extended preamble. Extended preamble is used if the remote RPC2A is in power save mode. Extended preamble length can be changed in the EEPROM memory.

Receive

On detection of preamble from the radio receiver, the RPC2A will phase lock, decode and error check the incoming synchronous data stream and if successful. The data is then placed in a buffer and the RX Request line is pulled low to signal to the host that a valid packet awaits to be uploaded to the Host.

An in-coming data packet is presented back to the host in the same form as it was given.

2 THE HOST INTERFACE

2.1 SIGNALS

It is recommended that the RPC2A be assigned to a byte wide bi-directional I/O port on the host processor. The port must be such that the 4 data lines can be direction controlled without affecting the 4 handshake line.

pin name	pin number	pin function	I/O	description
TXR	6	TX Request	I/P	Data transfer request from HOST to RPC2A
TXA	7	TX Accept	O/P	Data accept handshake back to HOST
RXR	8	RX Request	O/P	Data transfer request from RPC2A to HOST
RXA	9	RX Accept	I/P	Data accept handshake back to RPC2A
D0	2	Data 0 (4)	Bi-dir	4 bit bi-directional data bus. Tri-state
D1	3	Data 1 (5)	Bi-dir	between packet transfers, Driven on
D2	4	Data 2 (6)	Bi-dir	receipt for Accept signal until packet
D3	5	Data 3 (7)	Bi-dir	transfer is complete.

notes: 1. The 4 Handshake lines are active low
2. The 4 Data lines true data
3. Logic levels are 5V CMOS, see electrical specifications
4. Input pins have a weak pull-up internally

RESET
The Reset signal, may either be driven by the host (recommended) or pulled up to Vcc via a suitable resistor (10kΩ). A reset aborts any transfers in progress and restarts the Packet Controller.

HOST DRIVEN RESET
Minimum low time: 1.0 µs, after reset is released (returned high). The host should allow a delay 1ms after reset for the RPC2A to initialise itself

During this delay the host must hold TXR high (unless *DIAGNOSTIC MODES* are required) and RXR signal should be ignored.

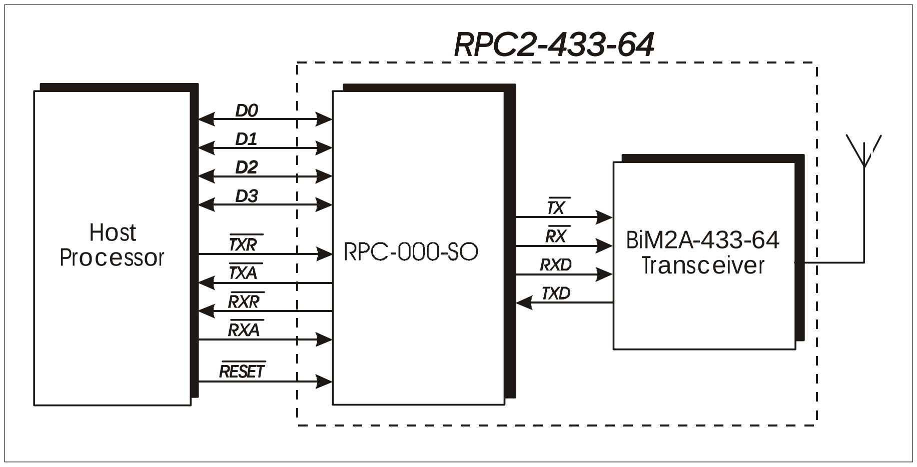


figure 5: Host to RPC2A connection

2.2 HOST TO RPC2A DATA TRANSFER

Data is transferred between the *RPC2A* and the *HOST* 4 bits (nibbles) at a time using a fully asynchronous protocol. The nibbles are always sent in pairs to form a byte, the *Least Significant Nibble* (bits 0 to 3) is transferred first, followed by the *Most Significant Nibble* (bits 4 to 7). Two pairs of handshake lines, *REQUEST* & *ACCEPT*, control the flow of data in each direction:-

TX Request & TX Accept:	control the flow from the HOST to the RPC2A (download)
RX Request & RX Accept:	control the flow from the RPC2A to the HOST (upload)

A packet transferred between host and RPC2A consists of between 1 and 28 bytes, the first byte of the packet is always the control byte.

There are two classes of HOST ↔ RPC2A transfers:

1. *Data Packets:*	To the transmitter or from the receiver
2. ***Memory Access:***	To or from the RPC2A's memory

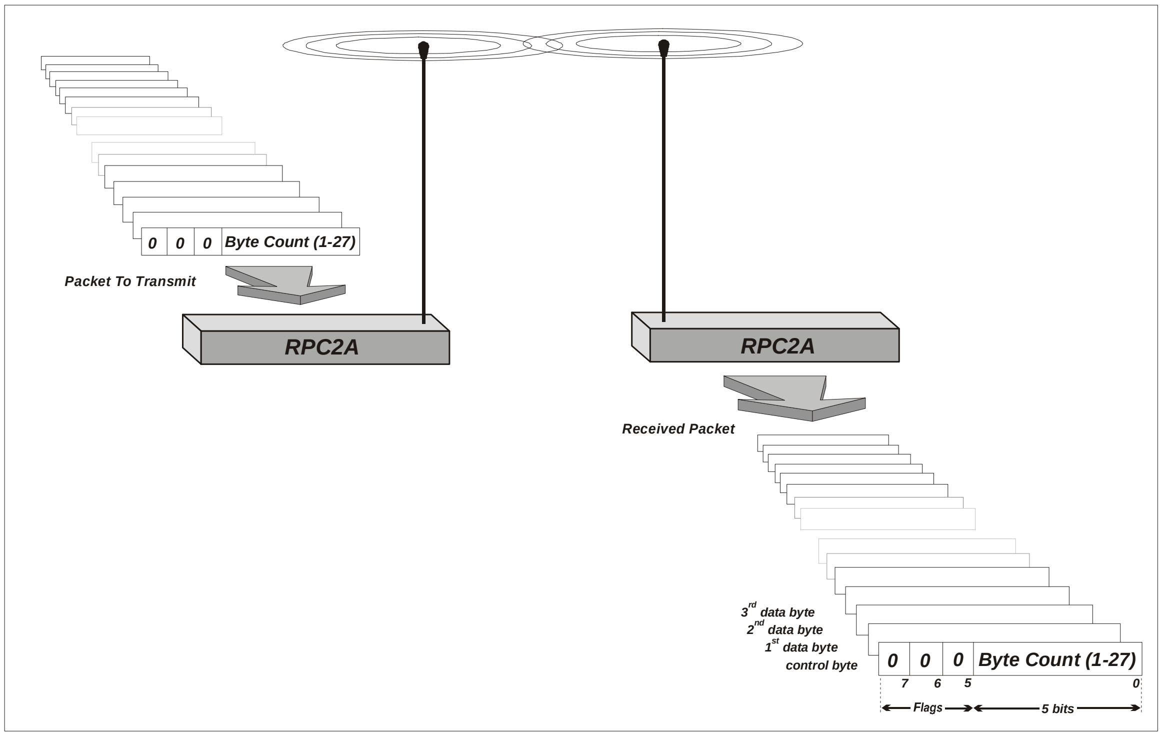


figure 6: RPC2A ↔ Host data transfer

2.1.1 *Write A Byte To RPC2A*

The sequence for a byte transfer from the Host to the RPC2A (i.e. TX download) is asynchronous and proceeds as follows:

1. HOST asserts TX Request line low to initiate transfer
2. Wait for RPC2A to pull TX Accept low (i.e. request is accepted)
3. Set data lines to output and place LS nibble on the data lines
4. Negate TX Request (set to 1) to tell RPC2A that data is present.
5. Wait for RPC2A to negate TX Accept (i.e. data has been accepted)

Repeat steps 1-5 with MS nibble.

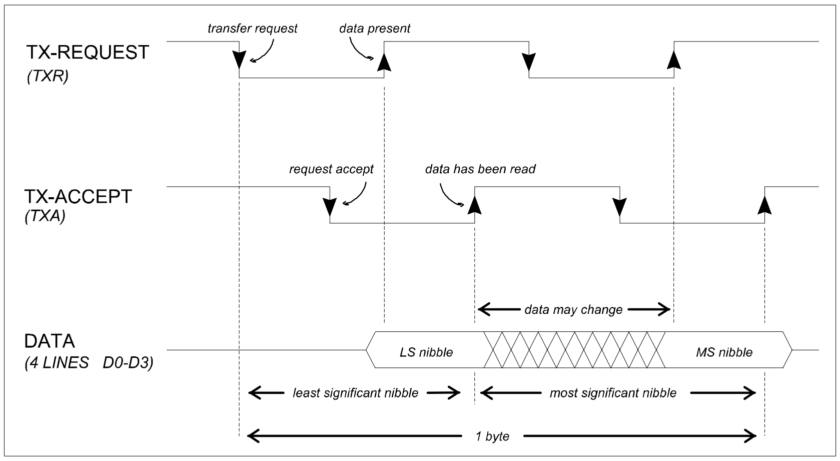


figure 7: TX download timing diagram

Notes:

- The data bus must not be set to output until step 3. i.e. after the RPC2A has accepted the request. The bus may be left as an output until the entire packet has been transferred to the RPC2A, it should then be set back to input (default state).

- The RPC2A's normal response time to the initial TX Request may be up to 1ms, thereafter, for the duration of the packet, the response will be fast.

- The RPC2A will ignore a TX Request from the Host while it is receiving a packet from the radio. If the incoming packet fails it's error check the RPC2A will respond to the TX Request as normal, i.e. the TX Accept from the RPC2A will be delayed until the incoming packet has finished. If a valid packet is received this must be uploaded to the Host before the RPC2A can respond to the Host's TX Request. Thus an RX Request will be signalled to the Host and not the expected TX Accept and the Host must upload the incoming packet before the TX packet can be downloaded. The TX Request should be left asserted (low) during the upload. The RPC2A will respond as normal after the upload is completed.

- For the above reason it is often easier to use RX Request to trigger a HOST interrupt and upload the RPC2A to the HOST under interrupt control.

- See Appendix B and C. for example RPC2A driver subroutines.

2.1.2 *Read a byte from the RPC2A*

The sequence for a byte transfer from the RPC2A to the HOST (i.e. RX upload) is asynchronous and proceeds as follows :-

1. RPC2A will assert RX Request line low to initiate transfer
2. Host pulls RX Accept low (i.e. request is accepted by the host)
3. RPC2A will turn on it's bus drivers, place LS nibble onto data lines and negate RX Request (set to 1)
4. Host reads the data and negates RX Accept (i.e. data has been accepted)

Repeat steps 1-4 with MS nibble.

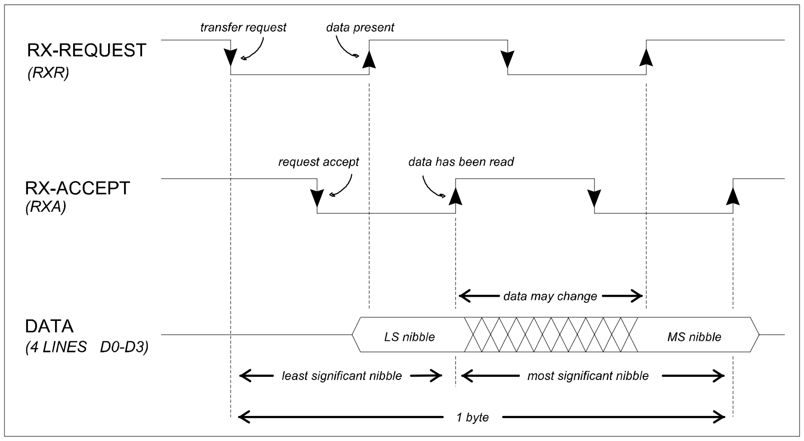


figure 8: RX upload timing diagram

Notes:
- The RPC2A will turn off it's data bus drivers after the entire packet has been uploaded to the HOST.
- See Appendix B and C. for example RPC2A driver subroutines.

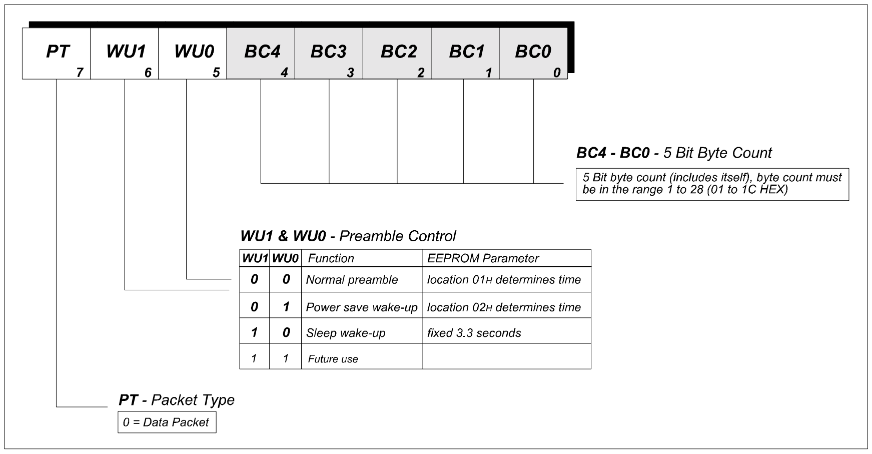
2.2 Host <> RPC2A Packet Format

2.2.1 *The Control Byte*

The first byte of a RPC2A <> HOST packet transfer is always the *Control Byte*. This byte is used to control the transfer and contains information about the type of packet, number of bytes to be transferred, memory address, read/write bit etc. Bit 7 of the control byte is the Packet Type flag, PT, it determines the class of transfer and the interpretation of the other bits in the control byte.

2.2.2 Sending and receiving Data Packets

Data packets are sent to / received from remote RPC2A's. They begin with a control byte with bit 7 cleared and may be of variable length and contain up to 27 bytes of user determined data.


PT 7
WU1 6
WU0 5
BC4 4
BC3 3
BC2 2
BC1 1
BC0 0
BC4 - BC0 - 5 Bit Byte Count
5 Bit byte count (includes itself), byte count must be in the range 1 to 28 (01 to 1C HEX)
WU1 & WU0 - Preamble Control
PT - Packet Type
0 = Data Packet

WU1	WU0	Function	EEPROM Parameter
0	0	Normal preamble	location 01H determines time
0	1	Power save wake-up	location 02H determines time
1	0	Sleep wake-up	fixed 3.3 seconds
1	1	Future use	

figure 9: Control byte for data packet

The remainder of the bytes in the data packet are of the users determination. The packet would usually be made up of a number of fields consisting of some but not necessarily all of the following :-

Source address / ID
Destination address / ID
System ID
Packet count
Encryption / Scrambler control
Additional error check codes (The RPC2A performs it's own error checks)
Routing information (for repeaters)
Link control codes (connect/disconnect/ACK/NAK etc.)
Data field

2.2.3 RPC2A Memory Access

The RPC2A's EEPROM memory can be accessed by setting bit 7 in the control byte. Bit 6 (R/W flag) defines a memory read or write. The bits left define the address.

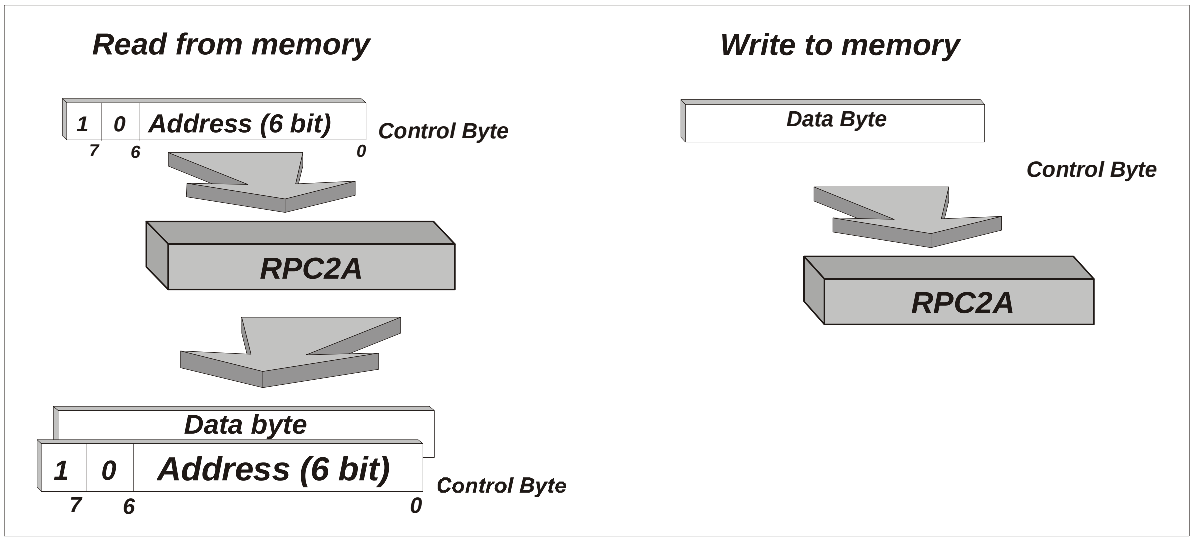


figure 10: RPC2A memory access

RPC2A Memory READS:
Host issues just the control byte, with bit 6 (W/R) cleared, bit 7 (PT) set and the memory address. The RPC2A will respond with 2 bytes, the first is a control byte which is an echo of the control byte just issued by the host, this is useful if the host is using an interrupt handler. The 2nd byte is the memory contents.

RPC2A Memory WRITES:
Host issues 2 bytes, the first is the control byte with bit 6 (W/R) set, bit 7 (PT) set and the memory address. The 2nd byte is the data to be written. The RPC2A does not give a response to memory writes.

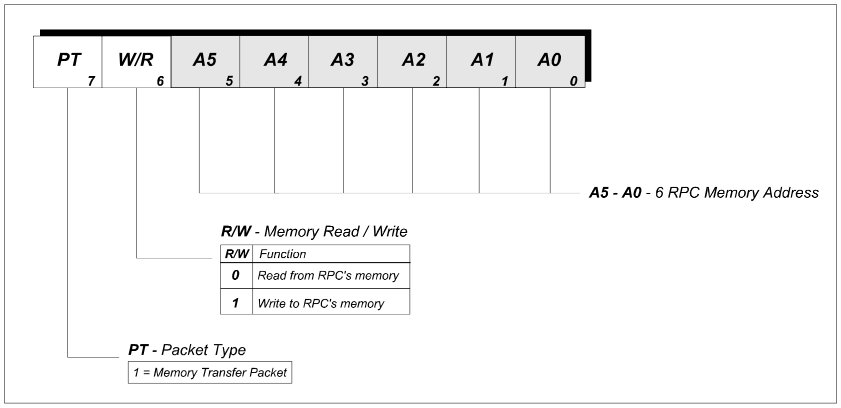


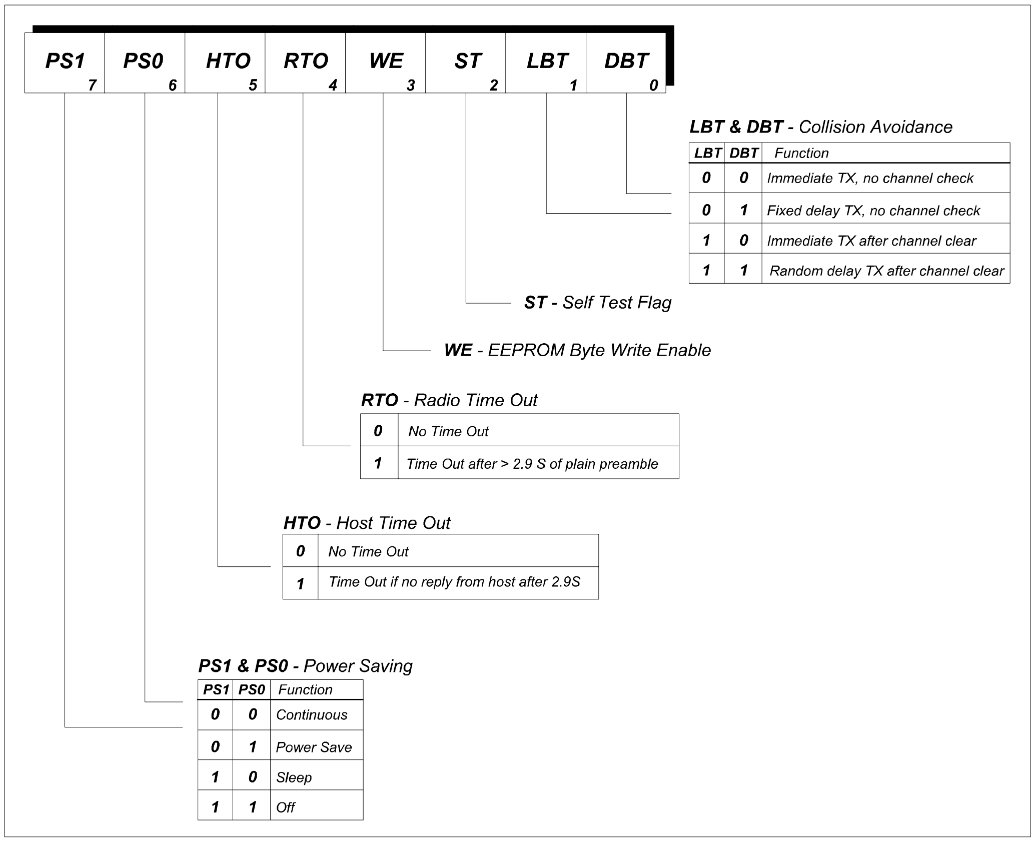
figure 11: Control byte for memory access

Notes Memory writes to locations 01 to 3F, write to the non-volatile EEPROM in the RPC2A. The EEPROM has a limit of 100,000 write cycles therefor it's use must be restricted to infrequently changed data. The RPC2A only writes to the EEPROM when instructed to by the HOST. Each byte takes 10ms to write. To prevent accidental/spurious writes to EEPROM the host must set the WE bit in SWITCHES prior to EACH byte to be written. We recommend that the host performs a read/verify after each byte write to EEPROM.

The above does not apply to any memory reads nor to writes to SWITCHES (address 00h).

3.0 RPC2A'S SWITCHES

SWITCHES is memory location 00h in RAM, it contains 8 flags which are used to determine the RPC2A's operation. On RPC2A reset, power-up or watchdog Time-Out it is loaded from location 08h (in EEPROM). The default value is 00 hex - this is all functions deselected.



LBT	DBT	Function
0	0	Immediate TX, no channel check
0	1	Fixed delay TX, no channel check
1	0	Immediate TX after channel clear
1	1	Random delay TX after channel clear

RTO	
0	No Time Out
1	Time Out after > 2.9 S of plain preamble

HTO	
0	No Time Out
1	Time Out if no reply from host after 2.9S

PS1	PS0	Function
0	0	Continuous
0	1	Power Save
1	0	Sleep
1	1	Off

figure 12: Switches

3.1 PS0 & PS1 - POWER SAVING

The RPC2A has 4 levels of power saving selected by PS0 & PS1 in SWITCHES. Power saving is achieved by shutting down the Transceiver and the RPC2A for a period of time (*OFF-TIME*) when the RPC2A is in the Idle state (i.e. nothing happening). During the *OFF* period current is reduced to the device leakage of <50 µA typ. The RPC2A will still respond immediately to a Host TX Request but cannot receive radio signals. After the programmed *OFF-TIME* the RPC2A will wake itself up, turn the receiver on and listen for valid preamble. ON time = PWR->RX (EEPROM address 05h) + 1ms = 4ms (using RPC2A Default values) If preamble is found the RPC2A will stay ON and decode the packet, if not
the RPC2A will shut down for another OFF time period.

Also see - WAKE-UP (address 02h of EEPROM) and paragraph 2.2.2 .

PS1	***PS0***		
0	***0***	CONTINUOUS	20mA (no power saving)
0	***1***	POWER SAVE	programmable sleeptime *
1	***0***	SLEEP	< 100µA (fixed off time of 2.9s)
1	***1***	OFF	< 50µA Transceiver is off (reset or TXR to wake-up)

* Sleeptime programmable in EEPROM address 03h.

value	**off -time**	**Average current**
00	22ms	2.95 mA
01	45ms	1.60 mA
02	90ms	0.85 mA
03	181ms	0.46 mA
04	362ms	0.26 mA
05	725ms	0.16 mA
06	1.45s	0.10 mA
07	2.9s	0.08 mA

The supply current's quoted above are typical for a BiM2A + RPC using the EEPROM default values.

3.2 HTO & RTO - Interface Time-Out

Both the Host and the Radio interfaces can 'hang' the RPC2A while it waits for an external event. Under error conditions the RPC2A will reset itself if the appropriate HTO or RTO switch is set.

RTO	**Radio Time Out.**
0	no time out
1	Time-Out and reset if > 2.9s of plain preamble detected. (note. valid extended preamble used for wake-ups will not cause a Time-Out to be detected)

HTO	**Host Time Out**
0	no time out
1	Time-Out and reset if Host fails to reply to any request or handshake within 2.9s

3.3 WE - EEPROM Write Enable

This bit protects the EEPROM from accidental writes, it must be set to 1 prior to each byte write to the EEPROM (addresses 01h to 3Fh). This bit will be cleared by the RPC2A after each byte write.

3.4 ST - Self Test Flag

Writing a 1 to this switch will initiate a radio self test. Both the transmitter and receiver are enabled, data is feed to the TX and checked for correct recover from the RX. If the test is good, the ST bit will set, if the test fails the ST bit will not set. The self test takes 20ms to complete.

3.5 LBT & DBT - Collision Avoidance

Listen Before Transmit, LBT, and Delay Before Transmit, DBT determine what collision avoidance the RPC2A will take before each transmission.

LBT	***DBT***	***Function***
0	***0***	***Immediate TX, no channel check***
0	***1***	***Fixed delay TX, no channel check (time slots)*** This is useful for rapid polling of up to 255 units by a master station. SLOTS is set to the units ID number, the packet size, preamble length and change over delay must be the same for all units being polled. see - EEPROM parameters
1	***0***	***Immediate TX, if channel is clear*** The receiver is turned on and the channel checked for preamble or data. The RPC2A will only go to transmit when the channel is clear.
1	***1***	***Random delay TX, if channel is clear*** This mode is useful in random access networks where there is a high statistical probability that more than 2 RPC2As could be attempting to transmit at the same time. The receiver is turned on and the channel is checked for preamble or data. If the channel is clear the RPC2A will go to transmit, if the channel is busy the RPC2A will delay by a random time (setable by TX-BACK-OFF in EEPROM) then try again for a clear channel.

4.0 User Configurable Parameters In EEPROM

The *EEPROM* has address range 01h - 3Fh (63 Bytes)
The first 15 BYTES (8 are defined) contain parameters used to control the RPC2A.

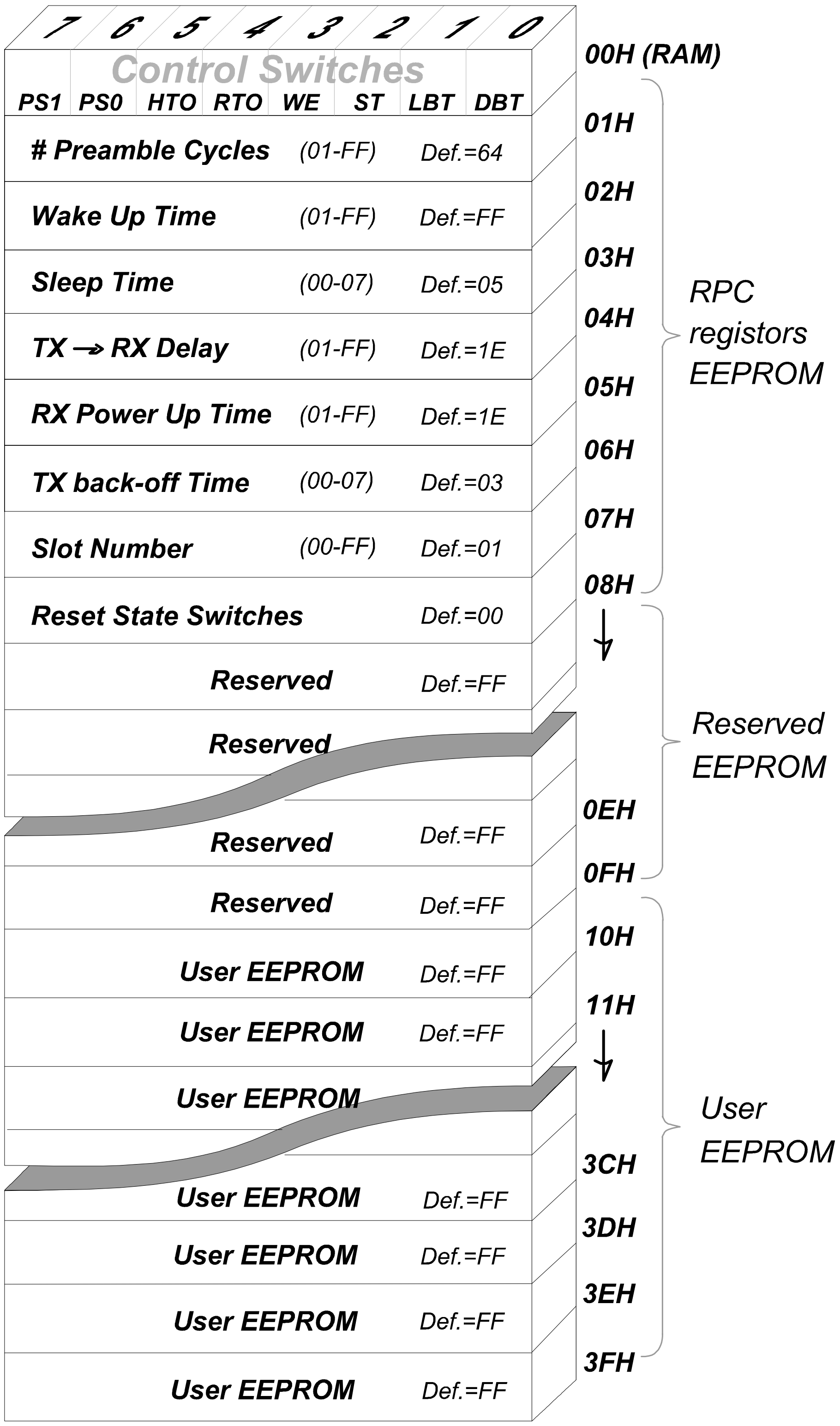


figure 13: RPC2A's EEPROM memory

PREAMBLE	***Number of "01" preamble cycles on TX packets***
	One '01' cycle takes 31.2µs @ 64kbit/s
address	01
default	A0
formula	Preamble time = *PREAMBLE* * 0.0312 ms
valid range	01 to FF

WAKE-UP	***Number of units of 'WAKE-UP PREAMBLE + PLEASE HOLD LINE'***
	To be sent as extended preamble to wake-up a remote RPC2A in power save mode. *WAKE-UP* should be set to approx. 1.5 times the remote units OFF Time
address	02
default	FF
formula	Wake-up message = *WAKE-UP* * 13.1 ms
valid range	01 to FF

SLEEP-TIME	***Power Save 'Off' Time (RC controlled)***
	The OFF time is controlled by an RC oscillator in the RPC2A which has a wide tolerance of +/- 30%
address	03
default	05
formula	Off-time = 22 * $2^{\textit{SLEEP-TIME}}$ ms
valid range	00 to 07

TX ↔ RX	***TX ↔ RX change over delay in units of 100µs***
address	04
default	1E
formula	Delay = TX ↔ RX * 0.1 ms
valid range	01 to FF

PWR → RX	***RX stabilisation delay in units of 100µs***
address:	05
default:	1E
formula:	Delay = PWR → RX * 0.1 ms
valid range:	01 to FF

TX-BACK-OFF	***Maximum TX Back-off delay in units of 1ms***
	Used when LBT=1 & DBT=1
address	06
default	03
formula	maximum delay = ($2^{\text{TX-BACK-OFF}}$ - 1) ms
valid range	00 to 07

00 = 0 - 1 ms	*04*= 0 - 31 ms
01 = 0 - 3 ms	*05* = 0 - 63 ms
02 = 0 - 7 ms	*06* = 0 - 127 ms
03 = 0 - 15 ms	*07* = 0 - 255 ms

TX-SLOT	***0 - 255 slot number for delayed (polled) TX***
	Delayed TX in packet units, used when LBT=0 & DBT=1
address	07
default	01
formula	delay = TX-Slot * (Preamble*0.05 + Tpacket + 3*TX ↔ RX + 0.5) ms
	where Tpacket = Number of bytes in packet * 0.30 ms
valid range	00 to FF

RESET STATE	***RESET STATE OF SWITCHES***
	The contents of this address are copied into *SWITCHES* on RPC2A reset, power-up or watchdog Time-Out
address	08
default	00

Address 09 to 0F are reserved for future and should not be used by the HOST

EEPROM Addresses 10 TO 3F (48 BYTES) are free for HOST use as general storage.

5.0 DIAGNOSTIC / DEBUG TEST MODES

These special test modes are useful for system testing and debugging

To select these modes the RPC2A should be released from reset with the TXR line held low, normal RPC2A operation will resume when the TXR is set high, i.e. TXR should be held while in these test modes.

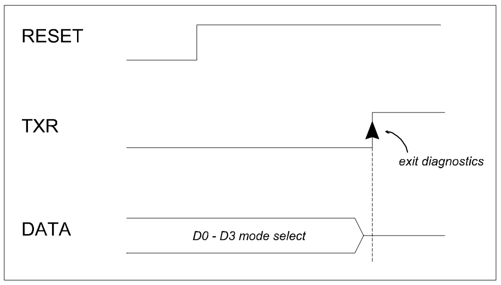


figure 14: diagnostic mode selection timing diagram

note: For normal operation of the RPC2A the TXR line must be held high for either 1ms after a reset pulse or 100ms after a power up.

There are 9 test modes which are selected by a binary code applied to the RPC2A's data bus. A 4 bit DIL switch or rotary HEX switch connected between the data bus and 0V will select the modes (the RPC2A has weak internal pull-up's). Alternatively the HOST may select the test modes by holding TXR low, resetting the RPC2A and driving the required test mode code onto the data bus.

note: The RPC2A continuously monitors the mode selected i.e.. a reset is not required on mode changes.

In some modes the RXR output from the RPC2A is driven low to indicate 'pass' or 'OK'. An LED + 1kΩ from RXR to 5V is recommended.

Mode	**Name**	**Function**
0	*RX-ON*	*PREAMBLE DETECTOR ON* (RXR RED LED = preamble detected)
1	*RX-PULSE*	10ms ON : 10ms OFF, *PREAMBLE DETECTOR ON* RXR LED
2	*TX-ON-PRE*	Preamble Modulation - send continuous preamble
3	*TX-ON-SQ*	100Hz *SQUARE WAVE MOD* - TX testing on spec. Analyser
4	*TX-ON-255*	random 64kbit/s data for *EYE DIAGRAM* tests, sync's on RXR
5	*TX-PULSE*	10ms ON : 10ms OFF, *PREAMBLE BURSTS*, RX lock in tests
6	*ECHO*	*TRANSPONDER MODE*, re-transmit any valid packets received
7	*RADAR*	Send ASCII *TEST PACKET* "RADIOMETRIX" and listen for echo
8	*SELF-TEST*	Loop test, TX > RX (OK on RXR)

Modes 6 & 7 are particularly useful for software debugging and range testing.

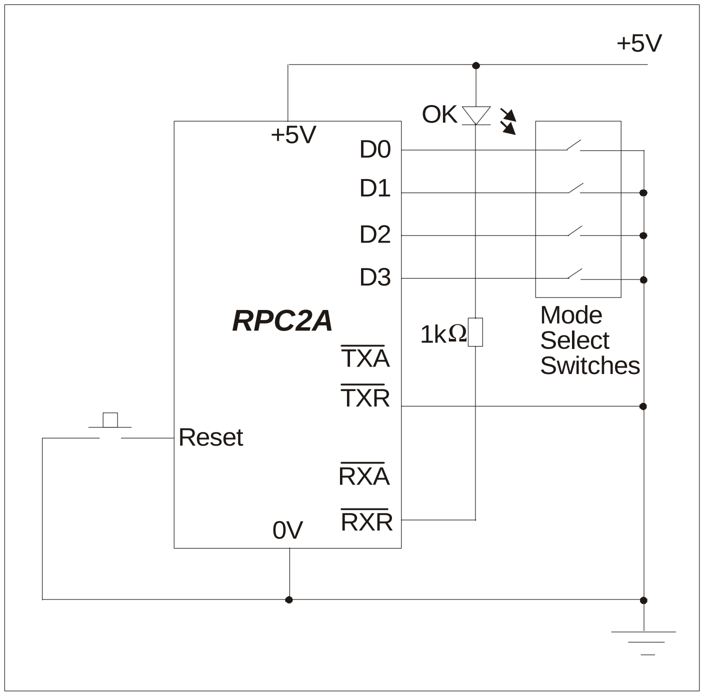


figure 15: stand-alone diagnostic mode

D3	**D2**	**D1**	**D0**	**Mode**
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8

Appendix - A

A Detailed look at the RPC2A's transceiver interface

The RPC2A interfaces to the transceiver using 4 lines :-

TX	*output*	Active low enable for the transmitter.
TXD	*output*	Serial data to be sent.
RX	*output*	Active low enable for the receiver.
RXD	*input*	Received serial data.

Note: ***1*** All lines are 5V CMOS levels
2 There is no requirement for a carrier/signal detect signal from the transceiver nor for the RXD output to be muted when no signal is present.

The enable lines - TX & RX
These normally high, active low lines are used to control the transceiver. The RPC2A is a half-duplex controller thus in normal operation the transceiver is either transmitting or receiving or off. The RPC2A only enables the TX and the RX at the same time during self test (local loop back).

Transmit Data - TXD
TXD is the serial data to the transmitter, it is held low when the transmitter is not enabled. When the TX is enabled a synchronous 64kbit/s (15.6µs/bit) serial code is present to modulate the transmitter.

Receive Data - RXD
RXD is a hi-impedance input which is fed with a 'squared-up' (5V logic level) signal from the receivers' data slicer. The RPC2A contains a very selective, noise immune signal detector and therefor does not require that the RXD signal be muted in the absence of signal, i.e.. squared-up channel noise may be fed to the RPC2A when no signal is present.

The RPC2A's Packet Encoder
The packet is made-up of 4 parts:

Preamble
This is a simple 32kHz square wave, the number of cycles being programmed by address 01h of the EEPROM. The preamble has two functions, the initial portion it is used to allow the data slicer in a remote receiver to establish the correct slicing point (for the A-433-64-S this takes a minimum of 3ms), after the receiver has settled, the remaining portion is used by the receiving RPC2A to positively identify and phase lock onto the incoming the signal (this requires 15 cycles of preamble). The preamble may extended to wake-up a remote RPC2A in power saving mode.

Frame sync
A 7 bit Barker sequence is used to identify the start of the data. Alternatively if the transmitter is sending extended preamble (to wake a power saving remote RPC2A) a complimented 7 bit Barker sequence is sent every 256 preamble cycles as a 'Please Hold The Line' code. An 8th balancing bit is added after the Barker sequence.

Data
Each byte in the RPC2A's buffer is expanded into a 12 bit symbol prior to sending. The symbol coding has the following properties :-
- Perfect 50:50 balance, i.e.. always 6 one's & 6 zero's
- There are never more than 4 consecutive one's or zero's. This minimises the low frequency components in the code and allows fast settling times to be used for the receivers' data slicer.
- Minimum Hamming distance = 2, i.e.. each code is different from any other code by a minimum of 2 bits, thus all odd number of bit errors will always be detected.
- In general only 256 of 4096 (6.25%) possible codes are valid, i.e.. a 93.75 % probability of trapping a byte error.
- Preamble and the Frame sync codes are not part of the symbol alphabet, a 'clash' signal will cause immediate termination of the current decode followed by an attempt to lock to the new signal.

Check Sum
Since the receiver checks each symbol for integrity, a simple 8 bit check sum is used to test for overall packet integrity. This is also coded into a 12 bit symbol prior to transmission.

The RPC2A's Packet Decoder

Signal Decoding is in 4 stages :-

Search
Initially the RPC2A's decoder searches the radio noise on the RXD line for the 32kHz preamble signal. The search is performed by a 16 times over-sampling detector which computes the spectral level of 32kHz in 240 samples of the RXD signal (750µs window).If the level exceeds a pre-set threshold the decoder will attempt to decode a packet.

Lock-in
The same set of 240 samples are used to compute the phase of the incoming preamble and synchronise the internal recovery clock to an accuracy of +/- 2µs. The recovery clock samples the mid point of each incoming data bit and shifts the samples trough an 8 bit serial comparator. The comparator searches the data on a bit by bit basis for the frame sync byte. While the search is in progress, the decode will abort if the preamble fails to maintain a certain level of integrity. If the comparator finds the 'please hold the line' code used during extended wake-up preamble a phase re-lock is triggered to ensure accurate phase tracking until the actual packet arrives. When the frame sync is detected the decoder attains full synchronisation and will move to the Decode state.

Decode
Data is now taken in 12 bits at a time (one symbol), decoded into the original byte and placed in the receive buffer. The symbol decoder verifies each received symbol as valid (only 256 out of a possible 4096 are valid) and will immediately abort the decode on a symbol failure. The first byte contains the byte count and is used to determine the end of message.

Check Sum
The last byte is the received check sum, this is verified against a locally generated sum of all the received bytes in the packet. If it matches the packet is valid and RXR line will be pulled low to inform the Host that a packet awaits uploading.

Notes on error handling

The RPC2A's' decoder is deliberately non bit error tolerant, i.e.. no attempt is made to repair corrupt data bits. All of the redundancy in the code is directed towards error checking. For an FM radio link using short packet lengths, e.g. RPC2A + BiM , packets are either 100% or so grossly corrupt as to be unrecoverable. By the same reasoning, the Host is not informed when the RPC2A decoder aborts a packet decode since corrupt information is of little value. A packet acknowledge Time-Out and re-transmission is the preferred strategy for error handling.

APPENDIX - B

Example RPC2A driver subroutines for Arizona PIC16C73

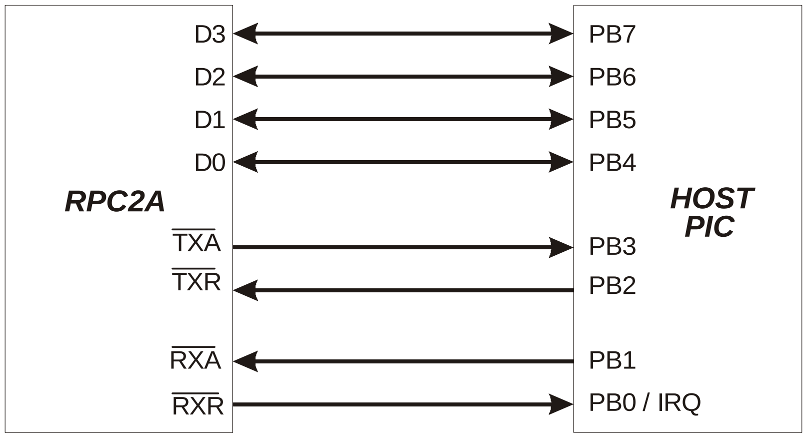


figure 16: RPC2A to PIC -μC interface

Packet transfers to / from the RPC2A are best handled in the host by two subroutines :- OUT_BYTE & IN_BYTE

Additionally LISTEN_BUS is called on completion of a packet transfer to the RPC2A to return the data bus to inputs (default state).

```
;--------------------------------------------------------------------;
;
;                             RPC2A DRIVERS
;
;
;--------------------------------------------------------------------
;
;                 HOST PROCESSOR PIC16C73 or similar
;
RPC2A EQU   06    ;USE PORT B ON PIC
;
;     ** Bit assignments for RPC2A PORT **
;
D7    EQU   7     ;Bi-Dir
D6    EQU   6     ;Bi-Dir
D5    EQU   5     ;Bi-Dir
D4    EQU   4     ;Bi-Dir
TXA   EQU   3     ;INPUT
TXR   EQU   2     ;OUTPUT
RXA   EQU   1     ;OUTPUT
RXR   EQU   0     ;INPUT  ON RB0, CAN BE CONFIGURED AS AN INTERRUPT
;
RPC2A_DDR 86            ;Data direction register for port B (RPC2A)
;                 ;This register is in BANK 1 of the register file
;
;--------------------------------------------------------------------
;
W     EQU   0     ;Accumulator as Destination
F     EQU   1     ;Register File as Destination
INDF  EQU   00    ;INDirect File register
```

```
;SUBROUTINE IN_BYTE
;
;IN_BYTE    READ A BYTE FROM THE RPC2A INTO FILE POINTED TO BY FSR
;           W IS DESTROYED
;
;           NOTE  THIS ROUTINE WILL HANG THE HOST UNTIL THE HOST
;                 COMPLETES THE TRANSFER OF TWO NIBBLES
;
;                 THIS SUBROUTINE CAN BE CONFIGURES TO RUN AS PART OF
                  ANINTERRUPT HANDLER IF THE  :RXR LINE FROM THE RPC2A
:                 IS USED TO TRIGGER A HOST INTERRUPT
;
;
IN_BYTE     BTFSC RPC2A,RXR   ;WE GOT A RX REQUEST YET?
            GOTO  IN-BYTE     ;NO , SO LOOP BACK AND WAIT
;
;                             READ THE LS NIBBLE FROM THE RPC2A
;
            BCF   RPC2A,RXA   ;ACCEPT THE REQUEST (SET ACCEPT LOW)
;
AWAITDATA   BTFSS RPC2A,RXR   ;HAS REQUEST GONE UP? data is present
            GOTO  AWAITDATA   ;LOOP BACK TILL IT DOES
;
            NOP               ;TIME DELAY TO ENSURE DATA STABLE
;                             ;BEFORE READ
;
            MOVF  RPC2A,W           ;READ THE LS NIBBLE FROM THE BUS
            BSF   RPC2A,RXA   ;TELL RPC2A WE GOT NIBBLE (ACCEPT = 1)
            ANDLW B'11110000' ;JUST THE DATA
;
            MOVWF INDF        ;SAVE LS NIBBLE IN TARGET FILE (VIA
;                             ;FSR)
            SWAPF INDF        ;RIGHT JUSTIFY LS NIBBLE
;
;                 NOW GET MS NIBBLE FROM THE RPC2A
;
;
INNIBBLE    BTFSC RPC2A,RXR   ;WE GOT NEXT RX REQUEST YET ?
            GOTO  INNIBBLE    ;NO , SO LOOP BACK AND WAIT
;
            BCF   RPC2A,RXA   ;ACCEPT REQUEST (SET ACCEPT LOW)
;
AWAITD1     BTFSS RPC2A,RXR   ;HAS REQUEST GONE UP? data is present
            GOTO  AWAITD1     ;LOOP BACK TILL IT DOES
;
            NOP               ;TIME DELAY TO ENSURE DATA STABLE
;                             ;BEFORE READ
;
            MOVF  RPC2A,W           ;READ THE MS NIBBLE FROM THE BUS
            BSF   RPC2A,RXA   ;TELL RPC2A  WE GOT NIBBLE (ACCEPT=1)
            ANDLW B'11110000' ;JUST THE DATA
;
            IORWF INDF        ;COMBINE MS NIBBLE WITH LS NIBBLE
                              ;ALREADY
;                             ;IN THE FILE (VIA FSR)RETURN
;
;  A BYTE HAS BEEN READ FROM THE RPC2A INTO ADDRESS POINTED AT BY FSR
;
;--------------------------------------------------------------------
;
```

```
;SUBROUTINE OUT_BYTE
;OUT_BYTE   WRITE A BYTE FROM FILE POINTED TO BY FSR TO RPC2A
;           W IS DESTROYED
;
;           NOTE  THIS ROUTINE WILL HANG THE HOST UNTIL THE RPC2A
;                 ACCEPTS THE TRANSFER OF TWO NIBBLES
;
;           WARNING     OUT_BYTE WILL SET THE DATA BUS TO DRIVE AFTER ;
            DETECTING A TXA FROM THE RPC2A.
;                       THE CALLING ROUTINE MUST SET 4 DATA LINES
;                       BACK TO I/P ON COMPLETION OF PACKET TRANSFER ;
            (i.e. call LISTENBUS)
;
OUT_BYTE    SWAPF INDF,W      ;GET LS NIBBLE FROM FILE (VIA FSR) INTO
;                             ;BITS 4 to 7 of W
            ANDLW B'11110000' ;JUST THE NIBBLE
            IORLW B'00000010' ;SET TXR LOW, LEAVE RXA HIGH
            MOVWF RPC2A       ;SET TXR LOW, OUTPUT NIBBLE
;
WACCEPT     BTFSC RPC2A,TXA   ;WE GOT A TX ACCEPT BACK YET?
            GOTO  WACCEPT     ;NO, SO LOOP BACK AND WAIT
;
;WE GOT ACCEPTANCE SO IT'S OK TO DRIVE BUS
;
            BSF   STATUS,RP0  ;SELECT PAGE 1
            MOVLW B'00001001' ;DRIVE BUS
            MOVWF RPC2A_DDR
            BCF   STATUS,RP0  ;SELECT PAGE 0 BUS IS NOW DRIVING
;
            BSF   RPC2A,TXR   ;REMOVE REQUEST, DATA IS ON BUS
WDUN        BTFSS RPC2A,TXA   ;HAS DATA BEEN READ?
            GOTO  WDUN        ;WAIT TILL RPC2A REMOVES ACCEPT
;
;LS NIBBLE OF (FSR) IS SENT , NOW DO MS NIBBLE
;
            MOVF  INDF,W      ;GET MS NIBBLE FROM FILE (VIA FSR)
;
            ANDLW B'11110000' ;JUST THE MS NIBBLE
            IORLW B'00000010' ;SET TXR LOW (BIT 2), RXA STAYS HIGH
            MOVWF RPC2A       ;OUTPUT NIBBLE + TXR LOW
;
WACCEPT1    BTFSC RPC2A,TXA   ;WE GOT A TX ACCEPT BACK YET?
            GOTO  WACCEPT1    ;NO, SO LOOP BACK AND WAIT
;
            BSF   RPC2A,TXR   ;REMOVE REQUEST, DATA IS ON BUS
;
WDUN1       BTFSS RPC2A,TXA   ;HAS DATA BEEN READ?
            GOTO  WDUN1       ;WAIT TILL RPC2A REMOVES ACCEPT
;
            RETURN
;
;            BYTE IS SENT TO RPC2A
;---------------------------------------------------------------------
; SUBROUTINE - LISTEN_BUS , SET DATA BUS TO INPUT
;
LISTEN_BUS  BSF STATUS,RP0    ;SELECT PAGE 1
            MOVLW B'11111001' ;BUS TO INPUT
            MOVWF RPC2A_DDR
            BCF   STATUS,RP0  ;SELECT PAGE 0
            RETURN
;               BUS IS LISTENING TO RPC2A
;---------------------------------------------------------------------
```

Example RPC2A driver subroutines for Motorola 68HC11

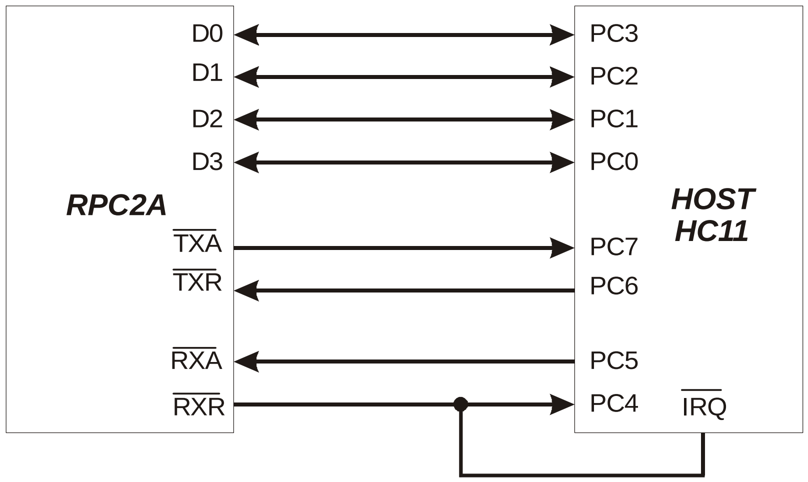


figure 17: RPC2A to HC11 μ-C interface

Packet transfers to / from the RPC2A are best handled in the host by two subroutines :- OUT_BYTE & IN_BYTE

Additionally LISTEN_BUS is called on completion of a packet transfer to the RPC2A to return the data bus to inputs (default state).

```
*********************************************************************
*
* CPU REGISTER EQUATIONS                                                 *
*********************************************************************
*
*This section contains a few of the necessary register equations used
*in the example subroutines.

PORTC       EQU   $1003       ;ADDRESS OF RPC2A PORT

DDRC        EQU   $1007       ;DATA DIRECTION REGISTER PORT-C

* Port-C7 = RX-accept  OUTPUT
* Port-C6 = RX-request INPUT
* Port-C5 = TX-accept  INPUT
* Port-C4 = TX-request OUTPUT
* Port-C3 = RPC2A data bit-3
* Port-C2 = RPC2A data bit-2
* Port-C1 = RPC2A data bit-1
* Port-C0 = RPC2A data bit-0
```

```
**********************************************************************
* RANDOM ACCESS MEMORY                                                *
*********************************************************************
ORG   RAM         ;RAM AREA DEFINITION

SAVE_1      RMB   1           ;TEMPORARILY SAVE LOCATION 1
SAVE_X      RMB   2           ;HOLDS FILES POINTER FOR IN_BYTE

**********************************************************************
* SUBROUTINE: IN_BYTE                                                 *
*********************************************************************

*This subroutine is designed to be called by an interrupt handler to
*read a byte from the RPC2A into a file pointed at by X
*
*Note: The interrupt handler should load the X register with the file address
before calling this subroutine.

IN_BYTE     CLR   SAVE_1      ;CLEAR TEMPORARILY MEMORY LOCATION
            LDAB  #%10010000  ;SET CORRECT DATA DIRECTION i/p
            STAB  DDRC
WAIT_RQ     LDAB  PORTC       ;WAIT FOR RX-REQUEST TO GO LOW
            BITB  #%01000000  ;
            BNE   WAIT_RQ
IN_LP       LDAB  PORTC
            ANDB  #%01111111  ;FORCE RX-ACCEPT TO GO LOW
            STAB  PORTC
WAIT_RQ1    LDAB  PORTC       ;WAIT FOR RX-REQUEST TO GO HIGH
            BITB  #%01000000
            BEQ   WAIT_RQ1
DAT_IN      LDAA  PORTC       ;READ IN DATA
            ANDA  #%00001111
            LDAB  PORTC       ;FORCE ACCEPT HIGH
            ORAB  #%10000000
            STAB  PORTC
            STAA  SAVE_1      ;SAVE NIBBLE TO TEMP LOCATION
WAIT_RQ2    LDAB  PORTC       ;WAIT FOR RX-REQUEST TO GO LOW
            BITB  #%01000000
            BNE   WAIT_RQ2
IN_LP2      LDAB  PORTC
            ANDB  #%01111111  ;FORCE RX-ACCEPT TO GO LOW
            STAB  PORTC
WAIT_RQ3    LDAB  PORTC       ;WAIT FOR RX-REQUEST TO GO HIGH
            BITB  #%01000000
            BEQ   WAIT_RQ3
DAT_IN2     LDAA  PORTC       ;READ IN DATA
            ANDA  #%00001111
            ASLA
            ASLA
            ASLA
            ASLA
            LDAB  PORTC       ;FORCE ACCEPT HIGH
            ORAB  #%10000000
            STAB  PORTC
            ORAA  SAVE_1      ;PUT NIBBLES TOGETHER IN TEMP LOCATION
            STAA  SAVE_1
READ_END    STAA  0,X         ;SAVE DATA TO POINTER ADDRESS
```

```
*********************************************************************
* SUBROUTINE: OUT_BYTE
*********************************************************************
*This subroutine will output of one byte to the RPC2A. Register X
*should contain the address of the memory location of the byte to be *send.
*Note: that register X has to be pre-loaded before entering this *
subroutine.

OUT_BYTE    LDAA  0,X          ;GET THE BYTE TO SEND TO RPC2A
            ANDA  #%00001111   ;PREPARE LEAST SIGNIFICANT NIBBLE
            LDAB  PORTC
            ANDB  #%11101111   ;FORCE TX-REQUEST LOW
            STAB  PORTC
WAIT_ACC    LDAB  PORTC
            BITB  #%00100000   ;WAIT FOR TX ACCEPT TO GO LOW
            BNE   WAIT_ACC
            LDAB  #%10011111   ;CHANGE DATA DDRC TO OUTPUT
            STAB  DDRC         ;TURN BUS DRIVE ON
            ORAA  #%10000000   ;MAKE SURE RXA IS HIGH
            STAA  PORTC        ;OUTPUT DATA
            LDAB  PORTC
            ORAB  #%00010000   ;FORCE TX-REQUEST HIGH
            STAB  PORTC
WAIT_REQ    LDAB  PORTC        ;WAIT FOR TX_ACCEPT TO GO HIGH
            BITB  #%00100000
            BEQ   WAIT_REQ
            LDAA  0,X          ;PREPARE MOST SIGNIFICANT NIBBLE
            LSRA               ;BY SWAPPING THE LS- & MS-NIBBLE
            LSRA
            LSRA
            LSRA
            LDAB  PORTC
            ANDB  #%11101111   ;FORCE TX-REQUEST LOW
            STAB  PORTC
WAIT_TXA1   LDAB  PORTC        ;WAIT FOR TX-ACCEPT TO GO LOW
            BITB  #%00100000
            BNE   WAIT_TXA1
            ORAA  #%10000000
            STAA  PORTC        ;OUTPUT DATA
            LDAB  PORTC
            ORAB  #%00010000   ;FORCE TX-REQUEST HIGH
            STAB  PORTC
WAIT_TXR1   LDAB  PORTC        ;WAIT FOR TX_ACCEPT TO GO HIGH
            BITB  #%00100000
            BEQ   WAIT_TXR1
            RTS

*********************************************************************
* SUBROUTINE: LISTEN TO BUS
********************************************************************
*
*This will turn the RPC2A host to listen mode again and should
*be called when the whole packet has been sent to the RPC2A
*
LISTEN_BUS  LDAA  #%10010000   ;PUT PORT BACK TO LISTEN
            STAA  DDRC
            RTS
********************************************************************
```

APPENDIX - D

The RPC2A as a control IC

Clock frequency

All timings within the RPC2A (except sleep) are determined by the clock frequency. The standard frequency is 16.38MHz and all timings unless explicitly stated otherwise, assume this clock frequency.

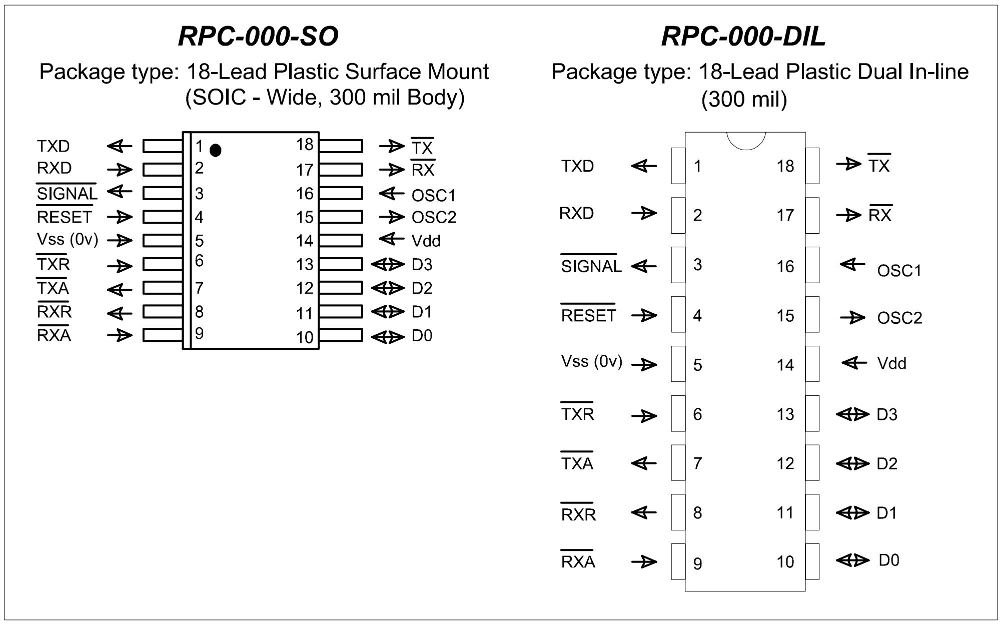
$$\text{The data rate} = \frac{f_{clk}}{256}\ \text{bit / s}$$ (i.e. 64kbit/s for Fclk=16.38MHz)

Clock accuracy

The RPC2A uses synchronous data transmission and requires an accurate reference clock. In the worst case , max. preamble and packet length, the allowable bit rate timing error between transmitter and receiver is 0.2 bits in 1000 bits, i.e. +/-200ppm total or +/-100ppm at each end.

$$\text{BIT TIME} = \frac{256}{f_{xtal}}\ \text{Hz}$$ i.e. 16.38MHz crystal = 31.2µs PER BIT

Accuracy, temp drifts MUST KEEP X-TAL +/- 100ppm of nominal



RPC-000-SO
Package type: 18-Lead Plastic Surface Mount (SOIC - Wide, 300 mil Body)
TXD 1, RXD 2, SIGNAL 3, RESET 4, Vss (0v) 5, TXR 6, TXA 7, RXR 8, RXA 9
TX 18, RX 17, OSC1 16, OSC2 15, Vdd 14, D3 13, D2 12, D1 11, D0 10
RPC-000-DIL
Package type: 18-Lead Plastic Dual In-line (300 mil)
TXD 1, RXD 2, SIGNAL 3, RESET 4, Vss (0v) 5, TXR 6, TXA 7, RXR 8, RXA 9
TX 18, RX 17, OSC1 16, OSC2 15, Vdd 14, D3 13, D2 12, D1 11, D0 10

figure 18: RPC2A-000-SO & RPC2A-000-DIL outlines

Radiometrix Ltd
Hartcran House
231 Kenton Lane
Harrow, Middlesex
HA3 8RP
ENGLAND
Tel: +44 (0) 20 8909 9595
Fax: +44 (0) 20 8909 2233
sales@radiometrix.com
www.radiometrix.com

Copyright notice

Limitation of liability

The information furnished by Radiometrix Ltd is believed to be accurate and reliable. Radiometrix Ltd reserves the right to make changes or improvements in the design, specification or manufacture of its subassembly products without notice. Radiometrix Ltd does not assume any liability arising from the application or use of any product or circuit described herein, nor for any infringements of patents or other rights of third parties which may result from the use of its products. This data sheet neither states nor implies warranty of any kind, including fitness for any particular application. These radio devices may be subject to radio interference and may not function as intended if interference is present. We do NOT recommend their use for life critical applications.
The Intrastat commodity code for all our modules is: 8542 6000.

R&TTE Directive

After 7 April 2001 the manufacturer can only place finished product on the market under the provisions of the R&TTE Directive. Equipment within the scope of the R&TTE Directive may demonstrate compliance to the essential requirements specified in Article 3 of the Directive, as appropriate to the particular equipment.
Further details are available on The Office of Communications (Ofcom) web site:
http://www.ofcom.org.uk/radiocomms/ifi/licensing/licensing_policy_manual/

Information Requests
Ofcom
Riverside House
2a Southwark Bridge Road
London SE1 9HA
Tel: +44 (0)*845 456 3000 or 020 7981 3040*
Fax: +44 (0)*20 7783 4033*
information.requests@ofcom.org.uk

European Radiocommunications Office (ERO)
Peblingehus
Nansensgade 19
DK 1366 Copenhagen
Tel. +45 33896300
Fax +45 33896330
ero@ero.dk
www.ero.dk