

Using the RCM Serial Interface

PulsON[®] 400 RCM

TIME DOMAIN[®]

Cummings Research Park
4955 Corporate Drive Suite 101
Huntsville, AL 35805 USA

<http://www.timedomain.com>

Tel: +1 256.922.9229

+1 888.826.8378

Fax: +1.256.922.0387

320-0287A
March 2011



Copyright

All rights reserved. Time Domain® 2001-2011. All rights reserved.

Trademarks

Time Domain®, PulsON®, and “PulsON Triangle” logo are registered trademarks of Time Domain. Ethernet® is a registered trademark of Xerox Corporation. Microsoft® and Windows XP®, Windows Vista®, and Windows 7® are registered trademarks of Microsoft Corporation. Any trademarks, trade names, service marks or service names owned or registered by any other company and used in this manual are the property of its respective company.

Rights

Rights to use this documentation are set forth in the PulsON Products Terms and Conditions of Sale.

Introduction

The PulsON 400 (P400) Ranging & Communications Module (RCM) offers several different interfaces allowing users to control the module according to their specific application needs. The protocol used to communicate with the RCM is fully defined in the *P400 RCM API Specification*. That document also has additional information for customers using the Ethernet interface to control the RCM.

Although using the serial port is straightforward, we recommend users of laptops or PCs with Ethernet interfaces use the Ethernet/UDP interface. Once you have become familiar with the RCM message passing structure, the serial universal asynchronous receiver transmitter (UART) interface will be easier to move forward to embedded processor communication using the serial TTL interface.

The RCM board has a 3.3V serial UART port which customers can use to communicate with the RCM. This document describes those features unique to the use of the serial UART.

Connector Information

The connector for the serial UART on the RCM is labeled “J12”. See **Figure 1** for the location of the connector on the board.

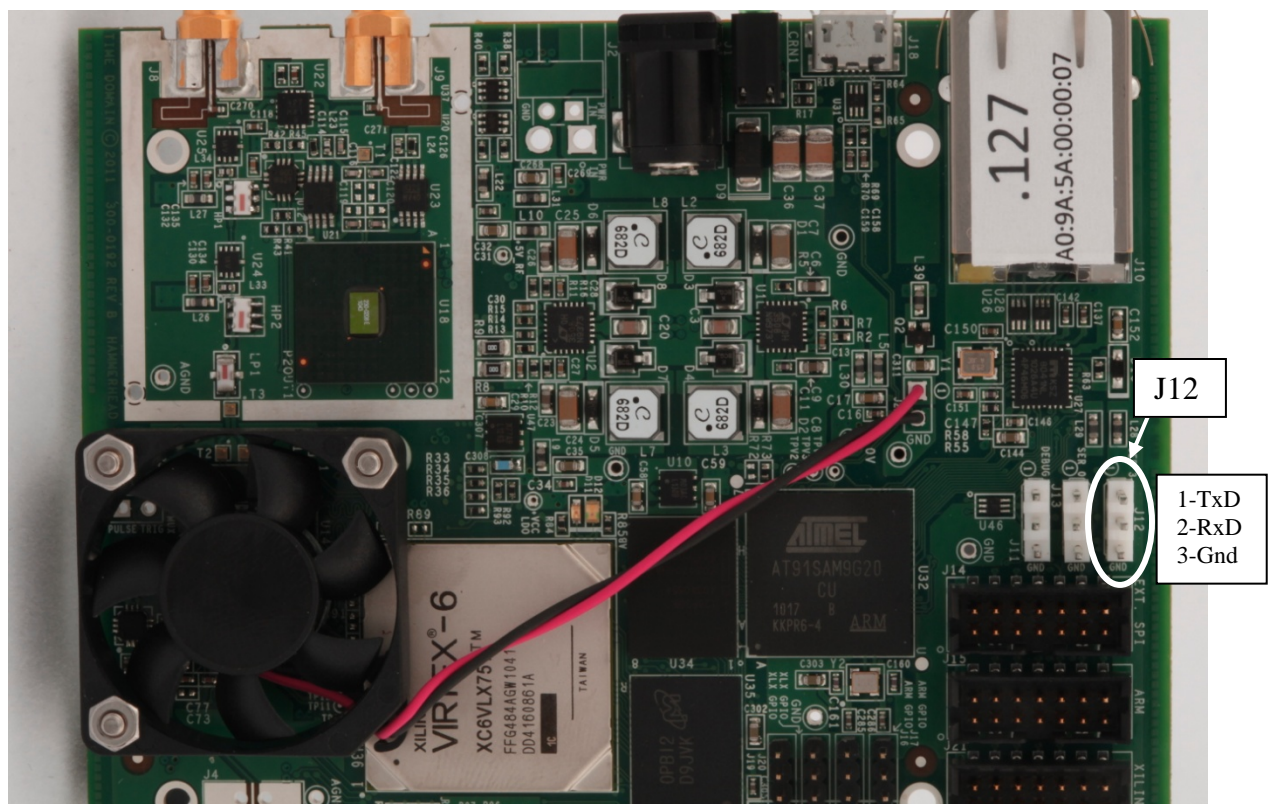


Fig. 1: RCM Serial UART connector location

The connector is a 3 pin header with the pin definitions shown in **Table 1**. Pin 1 is the pin closest to the Ethernet connector. Note that the serial UART uses 3.3V TTL levels as opposed to typical RS-232 voltages.

| Pin | Signal Name | Voltages |
|-----|---------------------------|-----------------------------|
| 1 | TxD (connect to host RxD) | +3.3V when 1, Ground when 0 |
| 2 | RxD (connect to host TxD) | +3.3V when 1, Ground when 0 |
| 3 | Ground | Reference |

Table 1: RCM Serial UART connector pinouts

Serial Protocol

The RCM serial UART uses the following communications parameters: 115,200 baud, 8 data bits, no parity, and 1 stop bit (115200, 8, N, 1). No flow control is used.

Commands sent to the RCM over the serial port are identical to the commands sent over the Ethernet interface, except each command must have 4 bytes prefixed to it and 2 bytes appended to it. The first two bytes of the prefix are a synchronization pattern (0xA5A5) and the next 2 bytes are the length of the command packet (i.e., the length of the command packet itself, not counting the 4 prefix bytes or 2 suffix bytes). The 2 suffix bytes are a 16-bit cyclic redundancy check (CRC) to ensure validity of the packet data. The length bytes and CRC bytes must be sent in network, or big-endian, byte order (all multi-byte fields in the RCM API are sent in network byte order).

The CRC used is the CCITT CRC-16. The CRC is computed over the command packet only; i.e., the CRC does not include the 4 prefix bytes. CRC sample code is included at the end of this document.

For example, the sequence of bytes in the RCM_GET_CONFIG_REQUEST message sent over the serial interface is shown in **Table 2**. All numbers are in hexadecimal format. For multi-byte fields, the most significant byte is designated by the abbreviation MSB, and the least significant byte is LSB.

| Byte Offset | Value | Description |
|-------------|-------|---------------------------|
| 0 | A5 | Sync header – first byte |
| 1 | A5 | Sync header – second byte |
| 2 | 00 | Length – MSB |
| 3 | 04 | Length – LSB |
| 4 | 00 | Command byte – MSB |
| 5 | 02 | Command byte – LSB |
| 6 | 00 | Message ID – MSB |
| 7 | 01 | Message ID – LSB |
| 8 | 7E | CRC – MSB |
| 9 | 41 | CRC - LSB |

Table 2: RCM_GET_CONFIG_REQUEST message

The resulting RCM_GET_CONFIG_CONFIRM packet sent from RCM is shown in **Table 3**.

| Byte Offset | Value | Description |
|-------------|-------|----------------------------------------|
| 00 | A5 | Sync header – first byte |
| 01 | A5 | Sync header – second byte |
| 02 | 00 | Length – MSB |
| 03 | 20 | Length – LSB |
| 04 | 00 | Command byte – MSB |
| 05 | 42 | Command byte – LSB |
| 06 | 00 | Message ID – MSB |
| 07 | 01 | Message ID – LSB |
| 08 | 00 | Node ID – MSB |
| 09 | 00 | Node ID – 2 nd byte |
| 0A | 00 | Node ID – 3 rd byte |
| 0B | 12 | Node ID – LSB |
| 0C | 00 | Pulse integration index – MSB |
| 0D | 07 | Pulse integration index – LSB |
| 0E | 00 | Antenna mode |
| 0F | 00 | Code channel |
| 10 | 00 | Antenna delay A – MSB |
| 11 | 00 | Antenna delay A – 2 nd byte |
| 12 | 00 | Antenna delay A – 3 rd byte |
| 13 | 00 | Antenna delay A – LSB |
| 14 | 00 | Antenna delay B – MSB |
| 15 | 00 | Antenna delay B – 2 nd byte |
| 16 | 00 | Antenna delay B – 3 rd byte |
| 17 | 00 | Antenna delay B – LSB |
| 18 | 00 | Flags – MSB |
| 19 | 00 | Flags – LSB |
| 1A | 00 | TX power |
| 1B | 00 | Unused |
| 1C | 00 | Timestamp – MSB |
| 1D | 08 | Timestamp – 2 nd byte |
| 1E | 93 | Timestamp – 3 rd byte |
| 1F | CC | Timestamp – LSB |
| 20 | 00 | Status – MSB |
| 21 | 00 | Status – 2 nd byte |
| 22 | 00 | Status – 3 rd byte |
| 23 | 00 | Status – LSB |
| 24 | F2 | CRC – MSB |
| 25 | 12 | CRC – LSB |

Table 3: RCM_GET_CONFIG_CONFIRM Message

See the P400 RCM *API Specification* for additional information about the messages supported by the RCM.

CRC Sample Code

The CRC used to protect the serial data is the CCITT CRC-16. The CRC is computed over the command packet only (not the first 4 prefix bytes). Also, the 2 length bytes should not include the 2 bytes of the CRC. Like all multi-byte fields, the CRC should be sent in network byte order. Sample C code to calculate the CRC is included below.

```
// Table of CRC constants - implements  $x^{16}+x^{12}+x^5+1$ 

static unsigned short crc16_tab[] = {
    0x0000, 0x1021, 0x2042, 0x3063, 0x4084, 0x50a5, 0x60c6, 0x70e7,
    0x8108, 0x9129, 0xa14a, 0xb16b, 0xc18c, 0xd1ad, 0xe1ce, 0xf1ef,
    0x1231, 0x0210, 0x3273, 0x2252, 0x52b5, 0x4294, 0x72f7, 0x62d6,
    0x9339, 0x8318, 0xb37b, 0xa35a, 0xd3bd, 0xc39c, 0xf3ff, 0xe3de,
    0x2462, 0x3443, 0x0420, 0x1401, 0x64e6, 0x74c7, 0x44a4, 0x5485,
    0xa56a, 0xb54b, 0x8528, 0x9509, 0xe5ee, 0xf5cf, 0xc5ac, 0xd58d,
    0x3653, 0x2672, 0x1611, 0x0630, 0x76d7, 0x66f6, 0x5695, 0x46b4,
    0xb75b, 0xa77a, 0x9719, 0x8738, 0xf7df, 0xe7fe, 0xd79d, 0xc7bc,
    0x48c4, 0x58e5, 0x6886, 0x78a7, 0x0840, 0x1861, 0x2802, 0x3823,
    0xc9cc, 0xd9ed, 0xe98e, 0xf9af, 0x8948, 0x9969, 0xa90a, 0xb92b,
    0x5af5, 0x4ad4, 0x7ab7, 0x6a96, 0x1a71, 0x0a50, 0x3a33, 0x2a12,
    0xdbfd, 0xcbbc, 0xfbbf, 0xeb9e, 0x9b79, 0x8b58, 0xbb3b, 0xab1a,
    0x6ca6, 0x7c87, 0x4ce4, 0x5cc5, 0x2c22, 0x3c03, 0x0c60, 0x1c41,
    0xedae, 0xfd8f, 0xcdec, 0xddcd, 0xad2a, 0xbd0b, 0x8d68, 0x9d49,
    0x7e97, 0x6eb6, 0x5ed5, 0x4ef4, 0x3e13, 0x2e32, 0x1e51, 0x0e70,
    0xff9f, 0xefbe, 0xdfdd, 0xcffc, 0xbf1b, 0xaf3a, 0x9f59, 0x8f78,
    0x9188, 0x81a9, 0xb1ca, 0xaleb, 0xd10c, 0xc12d, 0xf14e, 0xe16f,
    0x1080, 0x00a1, 0x30c2, 0x20e3, 0x5004, 0x4025, 0x7046, 0x6067,
    0x83b9, 0x9398, 0xa3fb, 0xb3da, 0xc33d, 0xd31c, 0xe37f, 0xf35e,
    0x02b1, 0x1290, 0x22f3, 0x32d2, 0x4235, 0x5214, 0x6277, 0x7256,
    0xb5ea, 0xa5cb, 0x95a8, 0x8589, 0xf56e, 0xe54f, 0xd52c, 0xc50d,
    0x34e2, 0x24c3, 0x14a0, 0x0481, 0x7466, 0x6447, 0x5424, 0x4405,
    0xa7db, 0xb7fa, 0x8799, 0x97b8, 0xe75f, 0xf77e, 0xc71d, 0xd73c,
    0x26d3, 0x36f2, 0x0691, 0x16b0, 0x6657, 0x7676, 0x4615, 0x5634,
    0xd94c, 0xc96d, 0xf90e, 0xe92f, 0x99c8, 0x89e9, 0xb98a, 0xa9ab,
    0x5844, 0x4865, 0x7806, 0x6827, 0x18c0, 0x08e1, 0x3882, 0x28a3,
    0xcb7d, 0xdb5c, 0xeb3f, 0xfb1e, 0x8bf9, 0x9bd8, 0xabbb, 0xbb9a,
    0x4a75, 0x5a54, 0x6a37, 0x7a16, 0x0af1, 0x1ad0, 0x2ab3, 0x3a92,
    0xfd2e, 0xed0f, 0xdd6c, 0xcd4d, 0xbdaa, 0xad8b, 0x9de8, 0x8dc9,
    0x7c26, 0x6c07, 0x5c64, 0x4c45, 0x3ca2, 0x2c83, 0x1ce0, 0x0cc1,
    0xef1f, 0xff3e, 0xcf5d, 0xdf7c, 0xaf9b, 0xbfba, 0x8fd9, 0x9ff8,
    0x6e17, 0x7e36, 0x4e55, 0x5e74, 0x2e93, 0x3eb2, 0x0ed1, 0x1ef0
};

unsigned short crc16(unsigned char *buf, int len)
{
    int i;
    unsigned short cksum = 0;

    for (i = 0; i < len; i++) {
        cksum = crc16_tab[((cksum>>8) ^ *buf++) & 0xFF] ^ (cksum << 8);
    }
    return cksum;
}
```