Document : PRONTO.WPD
Author : M. Majoor
Subject : Pronto communication and Pronto proprietary remote control formats

Revision overview

Revision	Date	Author	Description
20010408	2001-04-08	MM	Initial draft
20020609	2002-06-09	MM	Added information on how ProntoEdit/Emulator sets and uses the serial port Added information on 'irlearn'
20020629	2002-06-29	MM	Added remote control information
20020719	2002-07-19	MM	Added '!' remark for 'irstart'/'irstop' and updated some timing information. Added some information on the learned format.
20020727	2002-07-27	MM	Added Pronto format '0100'
20020817	2002-08-17	MM	Forgot to add customer code in RC6A list

Introduction

This document describes some of the basics behind the Pronto from a technical point of view. It describes the communication between the PC and the Pronto and will discuss the proprietary remote control formats used by the Pronto.

All test/actions with the Pronto are done using the following Pronto firmware:

Sys V3.62
App V4.85
RU890 V1.0

Communication with the Pronto

Communication with the Pronto is usually handled by the ProntoEdit or ProntoEmulator program. The basic communication specifications are:

Baudrate: 115200
Databits: 8
Parity: None
Stopbits: 1/2

The Pronto (re)acts on specific commands received through its serial port. These commands are:

Command	Meaning
q ccf	Query CCF
cap ccf	Report capability of CCF
possible ccf	Report possible capability CCF
dl ccf	Download CCF (to Pronto)
ul ccf	Upload CCF (from Pronto)
irlearn 5000	Learn IR code
irstart / irstop	Start/Stop of IR code
time	Report current time
reboot	Reboot Pronto

We will go into more detail later for each of these commands. For most, if not all, of the commands a dump of the serial communication is added. In such a printout the time relation of the data can be seen. Each line has a number (either 1 or 2), which indicates which device it is. In our case device 1 is the PC and device 2 is the Pronto. A typical dump looks like this:

```
----------------hex data ----------------------*-------ASCII------
00    74 69 6D 65 0D                          1  . time.           <- sent by PC
   21               74 69 6D 65 20 3D 20 32 39  2   !     time = 29  <- sent by Pronto
                                                1                    <- sent by PC
39 37 32 37 0A                                  2  9727.            <- sent by Pronto
```

The first data from the PC ($00) is not actual data being transmitted. In fact it is the result of issuing a break signal on the communication port for about 10 ms. This break signal means that, instead of the serial line being at its usual negative level, it

will go to a positive level. If the Pronto was inactive (screen off) we could also send the data $00 itself (or any other data for that matter) instead of a break signal, but this does not work when the Pronto is active (screen on). Using a break signal works in both circumstances.
The Pronto reponds with a '!' after it detects that a request for communication is issued. Be aware that the response time of the Pronto is relatively long. Before the '!' is issued you have to wait more than half a second (typical response time is 515 ms). It should be noted that, before a command is send after receiving the '!', a small delay of about 10 ms should be taken into account. Without this delay communication is more prone to go wrong (especially when multiple 'irstart'/'irstop' sequences are used).

Note: when the serial cable is plugged in, the Pronto will issue a '!' by itself. Also, when the serial cable is plugged in while both left and right keys are being pressed the Pronto will issue a '*' instead.

q ccf

This query command has the following, typical, response:

```
-----------------hex data ----------------------*-------ASCII------
00    71 20 63 63 66 0D                         1  . q ccf.
   21                   30 20 34 35 35 32 34 20  2   !      0 45524
                                                 1
32 30 30 30 31 32 30 36 20 32 33 31 32 34 37 0D  2  20001206 231247.
                                                 1
0A                                               2  .
```

This translates into:

0	'Dirty flag' indicating the CCF has been changed in the Pronto but has not yet been uploaded. In this case it means 'not dirty'.
45524	Size of the the CCF file (in bytes).
20001206	Date (6 December, 2000) of the CCF file. This is the ISO8601 notation of the date. See further in this document for more information on the ISO8601 formats.
231247	Time (23:12:47) of the CCF file. 24-hour notation.

Typical response time:

!	515 ms
answer	21 ms (time started after sending the command and stopped after receiving the complete answer)

cap ccf

This query command has the following, typical, response:

```
-----------------hex data ----------------------*-------ASCII------
00    63 61 70 20 63 63 66 0D                    1  . cap ccf.
   21                          63 61 70 3A 30 78  2   !        cap:0x
                                                 1
32 30 30 30 31 20 30 78 32 30 30 30 31 0D 0A     2  20001 0x20001..
```

This translates into:

0x20001 $20001 = 131073 bytes

Typical response time:

!	515 ms
answer	21 ms (time started after sending the command and stopped after receiving the complete answer)

possible ccf

This query command has the following, typical, response:

```
-----------------hex data ----------------------*-------ASCII------
00    70 6F 73 73 69 62 6C 65 20 63 63 66 0D     1  . possible ccf.
   21                                       63  2   !            c
                                                1
61 70 61 62 6C 65 3A 30 78 32 30 30 30 31 0D 0A  2  apable:0x20001..
```

This translates into:

0x20001 $20001 = 131073 bytes

Typical response time:

! 515 ms

answer 31 ms (time started after sending the command and stopped after receiving the complete answer)

dl ccf

This command will download the CCF file from the PC to the Pronto. Before this command is issued ProntoEdit investigates the current status of the Pronto. The typical start of commencing the download:

```
-----------------hex data ----------------------*-------ASCII------
00    63 61 70 20 63 63 66 0D                    1  . cap ccf.
   21                         63 61 70 3A 30 78  2   !        cap:0x
                                                 00  1                .
32 30 30 30 31 20 30 78 32 30 30 30 31 0D 0A     2  20001 0x20001..
   70 6F 73 73 69 62 6C 65 20 63 63 66 0D        1   possible ccf.
21                                         63 61  2  !             ca
                                                 00  1                .
70 61 62 6C 65 3A 30 78 32 30 30 30 31 0D 0A     2  pable:0x20001..
   71 20 63 63 66 0D                             1   q ccf.
21                   30 20 34 35 35 32 34 20 32  2  !      0 45524 2
                                                    1
30 30 30 31 32 30 36 20 32 33 31 32 34 37 0D 0A  2  0001206 231247..
00    64 6C 20 63 63 66 0D    02 01 FE 00 00 22  1  . dl ccf. .....
   21                      43                    2   !       C
.................ETC.............................................
.................ETC.............................................
```

After the 'dl ccf' command the data is send using the XModem/CRC protocol. For more information see the 'Upload/download protocol' section.

When all data is sent to the Pronto, the Pronto itself will issue the 'reboot' command. The only difference with a manually 'reboot' command is some additional text at the end ('freshly .... done'). See the 'reboot' command for the full response. The typical response from downloading ends in:

```
-----------------hex data ----------------------*-------ASCII------
0D 5F 43 43 46 2C 52 55 38 39 30 20 56 31 2E 30  2  ._CCF,RU890 V1.0
                                                 1
20 5B 30 31 2F 31 32 2F 32 30 30 31 20 31 32 3A  2   [01/12/2001 12:
                                                 1
31 30 5D 0A 0D 66 72 65 73 68 6C 79 20 6C 6F 61  2  10]..freshly loa
                                                 1
64 65 64 20 43 43 46 0A 0D 75 70 64 61 74 69 6E  2  ded CCF..updatin
                                                 1
67 20 61 6C 69 61 73 65 73 2E 2E 2E 20 64 6F 6E  2  g aliases... don
                                                 1
65 0A 0D 21                                      2  e..!
```

ul ccf

This command will upload the CCF file from the Pronto to the PC. Before this command is issued ProntoEdit investigates the current status of the Pronto. The typical start of commencing the upload:

```
-----------------hex data ----------------------*-------ASCII------
      00    71 20 63 63 66 0D                    1    . q ccf.
         21                   30 20 34 35 35 32  2     !      0 4552
                                                 1
34 20 32 30 30 30 31 32 30 36 20 32 33 31 32 34  2  4 20001206 23124
         00    75 6C 20 63 63 66 0D 43           1     . ul ccf.C
37 0D 0A    21                         02 01 FE  2  7.. !        ...
                                                 1
00 00 22 24 00 00 00 00 40 A5 5A 40 5F 43 43 46  2  ..\"$....@.Z@_CCF
                                                 1
00 00 00 00 07 D0 0C 06 00 17 0C 2F 00 00 00 00  2  ............./....
.................ETC.................................................
.................ETC.................................................
```

After the 'ul ccf' command the data is send using the XModem/CRC protocol. For more information see the 'Upload/download protocol' section.

irlearn 5000

This command will put the Pronto into learning mode. The Pronto will send the learned data as soon as it has 'learned' it. In the mean time the PC queries the Pronto (with <C>) at regular intervals for its data. If the Pronto fails to 'learn' correctly, the Pronto will respond with old data after receiving the fifth query from the PC. If the PC does not receive any data after it's eleventh query then it will abort the whole and will send five times a $18.

The typical start of learning a code by the Pronto (in this case there are two queries):

```
-----------------hex data ----------------------*-------ASCII------
00    69 72 6C 65 61 72 6E 20 35 30 30 30 0D 43  1  . irlearn 5000.C
   21                                            2   !
43                                               1  C
   01 01 FE 00 00 00 6D 00 12 00 11 01 40 00 A0  2   ......m.....@..
                                                 1
00 15 00 3B 00 15 00 3B 00 15 00 13 00 15 00 13  2  ...;...;........
                                                 1
00 15 00 13 00 15 00 13 00 15 00 3B 00 15 00 13  2  ...........;....
                                                 1
00 15 00 3B 00 15 00 13 00 15 00 13 00 15 00 13  2  ...;............
                                                 1
00 15 00 13 00 15 00 3B 00 15 00 13 00 15 00 13  2  .......;........
                                                 1
00 15 03 4A 00 15 00 3B 00 15 00 3B 00 15 00 13  2  ...J...;...;....
                                                 1
00 15 00 13 00 15 00 13 00 15 00 13 00 15 00 3B  2  ...............;
                                                 1
00 15 00 13 00 15 00 3B 00 15 00 13 00 15 00 13  2  .......;........
                  06                             1        .
00 15 00 13 52 87    01 02 FD 00 15 00 13 00 15  2  ....R. .........
                                                 1
00 3B 00 15 00 13 00 15 00 13 00 15 03 78 00 15  2  .;...........x..
                                                 1
00 3B 00 15 00 3B 00 15 00 13 00 15 00 13 00 15  2  .;...;..........
                                                 1
00 13 00 15 00 13 00 15 00 3B 00 15 00 13 00 15  2  .........;......
                                                 1
00 3B 00 15 00 13 00 15 00 13 00 15 00 13 00 15  2  .;..............
                                                 1
00 13 00 15 00 3B 00 15 00 13 00 15 00 13 00 15  2  .....;..........
                                                 1
03 78 00 15 00 3B 00 15 00 3B 00 15 00 13 00 15  2  .x...;...;......
                                                 1
00 13 00 15 00 13 00 15 00 13 00 15 00 3B 00 15  2  .............;..
                                       06    15  1               . .
00 13 00 15 00 3B 00 15 00 13 D3 76    04    04  2  .....;.....v . .
06                                               1  .
                                                 2
```

The learned code transmitted here is (taken from the 'Edit IR Code' window of ProntoEdit):

```
0000 006d 0012 0011 0140 00a0 0015 003b
0015 003b 0015 0013 0015 0013 0015 0013
0015 0013 0015 003b 0015 0013 0015 003b
0015 0013 0015 0013 0015 0013 0015 0013
0015 003b 0015 0013 0015 0013 0015 034a
0015 003b 0015 003b 0015 0013 0015 0013
0015 0013 0015 0013 0015 003b 0015 0013
0015 003b 0015 0013 0015 0013 0015 0013
0015 0013 0015 003b 0015 0013 0015 0013
0015 0378
```

After the 'irlearn 5000' command the data is send using the XModem/CRC protocol. For more information see the 'Upload/download protocol' section.

A typical 'failed' learning by the Pronto:

```
-----------------hex data ----------------------*-------ASCII------
00    69 72 6C 65 61 72 6E 20 35 30 30 30 0D 43  1  . irlearn 5000.C
   21                                            2   !
43 43 43 43 43                                   1  CCCCC
               01 01 FE 00 00 00 6D 00 00 00 00  2       ......m....
                                                 1
01 40 00 A0 00 15 00 3B 00 15 00 3B 00 15 00 13  2  .@.....;...;....
                                                 1
00 15 00 13 00 15 00 13 00 15 00 13 00 15 00 3B  2  ...............;
                                                 1
00 15 00 13 00 15 00 13 00 15 00 3B 00 15 00 13  2  ...........;....
                                                 1
00 15 00 13 00 15 00 13 00 15 00 3B 00 15 00 13  2  ...........;....
                                                 1
00 15 00 13 00 15 03 4A 00 15 00 3B 00 15 00 3B  2  .......J...;...;
                                                 1
00 15 00 13 00 15 00 13 00 15 00 13 00 15 00 13  2  ................
                                                 1
00 15 00 3B 00 15 00 13 00 15 00 13 00 15 00 3B  2  ...;...........;
                                  06    15    06     1            . . .
00 15 00 13 00 15 00 13 2A D1    04    04        2  ........*. . .
```

Note: it seems that the Pronto does not allow for a negative acknowledgment from the PC. This means that when a $15 is send instead of the $06 the Pronto does not retransmits the block.

irstart / irstop

This command will send remote control data to the Pronto which it will transmit on its infrared output. This command is the same as used by ProntoEmulator for emulating the ProntoEdit CCF file.
The typical sequence:

```
-----------------hex data ----------------------*-------ASCII------
00    69 72 73 74 61 72 74 0D    01 01 FE 60 00  1  . irstart. ...`.
   21                          43                2   !        C
00 76 00 00 00 01 00 00 00 01 00 00 00 00 00 00  1  .v..............
                                                 2
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1  ................
                                                 2
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1  ................
                                                 2
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1  ................
                                                 2
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1  ................
                                                 2
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1  ................
                                                 2
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  1  ................
                                                 2
00 00 00 00 00 00 00 00 00 00 00 00 00 00 0D D7  1  ................
                                                 2
   04    04    69 72 73 74 6F 70 0D              1   . . irstop.
06    15    06                    21            2  . . .       !
```

There are some delays to be taken into account before receiving data from the Pronto and replying to it:
. 'irstop' wait about 30 ms before sending this.
The last '!' response from the Pronto is generated when it has send the infrared code. It is ONLY generated when the screen of the Pronto is active!
You can not send multiple 'irstart' commands without any delays to the Pronto, you should take a delay into account between commands (about 100 ms).
Instead of the '!' response as a result of the break condition, the Pronto can also respond with 'C' when itself detects a timeout (use a timeout of > 700 ms).

The data transmitted is the same data as in the 'Edit IR Code' window of ProntoEdit.
After the 'irstart' command the data is send using the XModem/CRC protocol. For more information see the 'Upload/download protocol' section.

Instead of responding with <$06> the Pronto might also respond with <C>.

Typical response time:

!	515 ms
<C>	28 ms
<$06>	27 ms (response time after data has been send)
<$15>	15 ms
<$06>	2 ms
!	515 ms

Important note: because a command send using 'irstart' is only transmitted once some devices which depend on repeated codes will not respond. When the same data is programmed into the Pronto the device will usually respond because the code is repeated while the key is being pressed.

Only a single proprietary format code is allowed to be send to the Pronto. All but the first one are ignored by the Pronto. Thus when the following code is required to be sent twice ("5000 0073 0000 0001 0000 0001" which is the equivalent of RC5 0 1) then sending "5000 0073 0000 0001 0000 0001 5000 0073 0000 0001 0000 0001" is not possible. This has to be divided into two separate 'irstart'/'irstop' commands. If you explicitly want to send such a code twice in one go, then you have to convert it into a learned code sequence.

time

When the "time" command is issued the Pronto will respond with its current time. The returned number is the number of seconds passed, so we must convert these into individual hours, minutes and seconds. Note that the number of hours extends past the, possible, 23 hours.

A typical communication dump looks like this:

```
-----------------hex data ----------------------*-------ASCII------
00    74 69 6D 65 0D                            1  . time.
   21               74 69 6D 65 20 3D 20 32 39  2   !     time = 29
                                                1
39 37 32 37 0A                                   2  9727.
```

In this response we would translate the number '299727' in the following way:
Hours (299727 div 3600) mod 24 == 83 mod 24 == **11**
Minutes (299727 div 60) mod 60 == 4995 mod 60 == **15**
Seconds 299727 mod 60 == **27**
Which results in 11:15:27 as the time.

Note: this is one of the few responses which only adds a single linefeed ($0A) instead of a carriage return ($0D) and a linefeed.

Typical response time:

!	515 ms
answer	15 ms (time started after sending the command and stopped after receiving the complete answer)

reboot

The reboot command is sometimes necessary when you communicate with the Pronto. At times, especially when communication fails, the Pronto seems to get 'hang-up' after a certain period of time (if you are lucky the Pronto itself will reboot itself using its internal watchdog mechanism but this is not always the case). If the Pronto 'hangs' you will have no visible display anymore although the remote control buttons could work correctly. If this happens you have to activate the reset button on the Pronto (back) to have it restarted. To circumvent this problem we can reboot the Pronto using the 'reboot' command. This command is (internally) executed by the Pronto itself when a new CCF file is uploaded to the Pronto. The 'reboot' command will restart the Pronto. Keep in mind that the rebooting process is not one continuous stream of data. The delays between receiving data can be long (more than 3 seconds in some cases). The last received '!' from the Pronto is received when the Pronto itself issues a beep sound.

```
-----------------hex data ----------------------*-------ASCII------
00    72 65 62 6F 6F 74 0D                      1  . reboot.
   21                      0A 0D 0A 0D 42 6F 6F  2   !       ....Boo
                                                 1
74 69 6E 67 2E 2E 2E 0A 0D 42 4F 4F 54 2C 34 30  2  ting.....BOOT,40
                                                 1
30 30 30 30 5B 30 30 5D 2C 56 31 2E 33 2E 32 20  2  0000[00],V1.3.2
                                                 1
41 4D 44 20 20 20 20 20 20 20 20 20 20 2C 31 30  2  AMD          ,10
                                                 1
2F 31 33 2F 31 39 39 39 2C 30 38 3A 33 34 3A 32  2  /13/1999,08:34:2
                                                 1
35 2C 43 52 43 20 4F 4B 0A 0D 32 30 0A 0D 50 53  2  5,CRC OK..20..PS
                                                 1
4F 53 2C 34 30 34 30 30 30 5B 30 32 5D 2C 56 31  2  OS,404000[02],V1
                                                 1
2E 30 20 20 20 20 20 20 20 20 20 20 20 20 20 20  2  .0
                                                 1
20 20 2C 30 33 2F 31 37 2F 32 30 30 30 2C 31 32  2    ,03/17/2000,12
                                                 1
3A 35 33 3A 35 30 2C 43 52 43 20 4F 4B 0A 0D 33  2  :53:50,CRC OK..3
                                                 1
30 0A 0D 5F 53 59 53 2C 34 31 30 30 30 30 5B 30  2  0.._SYS,410000[0
                                                 1
38 5D 2C 56 33 2E 36 32 20 20 20 20 20 20 20 20  2  8],V3.62
                                                 1
20 20 20 20 20 20 20 2C 30 33 2F 31 37 2F 32 30  2         ,03/17/20
                                                 1
30 30 2C 31 32 3A 35 35 3A 30 35 2C 43 52 43 20  2  00,12:55:05,CRC
                                                 1
4F 4B 0A 0D 36 30 0A 0D 5F 41 50 50 2C 34 34 30  2  OK..60.._APP,440
                                                 1
30 30 30 5B 31 31 5D 2C 41 70 70 20 56 34 2E 38  2  000[11],App V4.8
                                                 1
35 20 20 20 20 20 20 20 20 20 20 20 2C 30 33 2F  2  5           ,03/
                                                 1
31 37 2F 32 30 30 30 2C 31 32 3A 35 35 3A 30 30  2  17/2000,12:55:00
                                                 1
```

```
2C 43 52 43 20 4F 4B 0A 0D 34 30 0A 0D 5F 43 43  2  ,CRC OK..40.._CC
                                                  1
46 2C 34 61 31 30 30 30 5B 31 37 5D 2C 4E 2F 41  2  F,4a1000[17],N/A
                                                  1
20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20  2
                                                  1
20 2C 31 32 2F 30 36 2F 32 30 30 30 2C 32 33 3A  2   ,12/06/2000,23:
                                                  1
31 32 3A 34 37 2C 43 52 43 20 53 4B 50 0A 0D 35  2  12:47,CRC SKP..5
                                                  1
32 0A 0D 37 30 0A 0D 37 34 0A 0D 37 38 0A 0D 38  2  2..70..74..78..8
                                                  1
30 0A 0D 39 32 0A 0D 38 32 0A 0D 38 34 0A 0D 38  2  0..92..82..84..8
                                                  1
36 0A 0D 38 38 0A 0D 39 30 0A 0D 39 34 0A 0D 39  2  6..88..90..94..9
                                                  1
36 0A 0D 39 38 0A 0D 0A 0D 4C 5A 4F 20 72 65 61  2  6..98....LZO rea
                                                  1
6C 2D 74 69 6D 65 20 64 61 74 61 20 63 6F 6D 70  2  l-time data comp
                                                  1
72 65 73 73 69 6F 6E 20 6C 69 62 72 61 72 79 20  2  ression library
                                                  1
28 76 31 2E 30 34 2C 20 4D 61 72 20 31 35 20 31  2  (v1.04, Mar 15 1
                                                  1
39 39 38 29 2E 0A 0D 43 6F 70 79 72 69 67 68 74  2  998)...Copyright
                                                  1
20 28 43 29 20 31 39 39 36 2C 20 31 39 39 37 2C  2   (C) 1996, 1997,
                                                  1
20 31 39 39 38 20 4D 61 72 6B 75 73 20 46 72 61  2   1998 Markus Fra
                                                  1
6E 7A 20 58 61 76 65 72 20 4A 6F 68 61 6E 6E 65  2  nz Xaver Johanne
                                                  1
73 20 4F 62 65 72 68 75 6D 65 72 0A 0D 0A 0D 75  2  s Oberhumer....u
                                                  1
73 69 6E 67 20 65 6D 62 65 64 64 65 64 20 66 6F  2  sing embedded fo
                                                  1
6E 74 73 0A 0D 66 6F 6E 74 20 30 20 3D 20 50 72  2  nts..font 0 = Pr
                                                  1
6F 6E 74 6F 20 38 0A 0D 66 6F 6E 74 20 31 20 3D  2  onto 8..font 1 =
                                                  1
20 70 72 6F 6E 74 6F 31 30 0A 0D 66 6F 6E 74 20  2   pronto10..font
                                                  1
32 20 3D 20 50 72 6F 6E 74 6F 20 31 32 0A 0D 66  2  2 = Pronto 12..f
                                                  1
6F 6E 74 20 33 20 3D 20 50 72 6F 6E 74 6F 20 31  2  ont 3 = Pronto 1
                                                  1
34 0A 0D 66 6F 6E 74 20 34 20 3D 20 50 72 6F 6E  2  4..font 4 = Pron
                                                  1
74 6F 20 31 36 0A 0D 66 6F 6E 74 20 35 20 3D 20  2  to 16..font 5 =
                                                  1
50 72 6F 6E 74 6F 20 31 38 0A 0D 0A 0D 5F 43 43  2  Pronto 18...._CC
```

```
                                                        1
46 2C 52 55 38 39 30 20 56 31 2E 30 20 5B 31 32  2  F,RU890 V1.0 [12
                                                        1
2F 30 36 2F 32 30 30 30 20 32 33 3A 31 32 5D 0A  2  /06/2000 23:12].
                                                        1
0D 21                                            2  .!
```

Typical response time:

!	515 ms
Booting...	105 ms
<LF>	2006 ms
BOOT,...CRC OK 20	198 ms
PSOS,...CRC OK 30	508 ms
_SYS,...CRC OK 60	1081 ms
_APP,...CRC OK 40	4416 ms
_CCF,...CRC SKP 52	3 ms
70 74 78	231 ms
80 92 98	160 ms
LZO ...	3 ms
Copyright ...	11 ms
using embedded ...	108 ms
font 0	1 ms
<CR><LF>	170 ms
_CCF,...	3 ms
!	3312 ms

Upload/Download protocol

When a lot of data has to be transferred between the Pronto and the PC a XModem/CRC protocol mechanism is used. This protocol is based upon transmitting blocks of data with a checksum. The received data can be checked using the CRC checksum and is acknowledged by the receiving end. The following is a typical layout of the transmission (note that there a timeout issues involved in the whole protocol):

<C> Request a 2 byte CRC checksum to follow the data
Because the Pronto always uses a 2 byte checksum this is the only possible option.

<$02> We send a block of 1024 bytes
If we would send a <$01> this would indicate that 128 bytes of data would follow.

<$01> Block number $01

<$FE> Block number $01 (=$FF minus block number)

..... 1024 bytes of data
Or 128 bytes of data if we started with <$01> instead of <$02>.

<CRC> The CRC checksum (2 bytes). The high byte is transmitted first.

<CRC>

<$06> <ACK> Acknowledgment (for correct reception) of data block

At this point the next block can be send. The blocknumber is increased by 1.

Should something go wrong at the reception of a block of data the response is, obviously, not a <ACK>. The response would be:

<$15> <NAK> Negative acknowledgment (incorrect reception)

At this point the (same) data should be send again with the same block number.

When there is no more data to send (we just received the last block), the 'ending' part of such a communication cycle would be:

<CRC>

<CRC>

<$06> <ACK> Acknowledgment (for correct reception) of data block

<$04> <EOT> End of Text (no more data)

<$15> <NAK> Negative Acknowledgment (acknowledgment of EOT)
The, normal, XModem protocol would expect a <ACK> at this point and would end here. The Pronto/PC always assumes that the <EOT> has been detected incorrectly and request a resend of the <EOT> using a <NAK> instead of a <ACK>.

<$04> <EOT> End of Text (acknowledgment of NAK)

<$06> <ACK> Acknowledgment (of EOT)

If communication is to be aborted we would use multiple cancellations to inform the pronto/PC. This however does not always work because most of the time the other side is expecting a defined number of data bytes and does not check the data itself at those points. In such a case we need to continue sending data until we get a response or use a timeout. At any case we would send a number of cancellations to the Pronto/PC:

<$18><$18><$18><$18>

Calculation of the CRC checksum is done using the following algorithm:

1. Reset the checksum to zero
2. For each byte to be transmitted in the block of data ('Data') adjust the checksum accordingly (in Pascal):

```
newCRC := oldCRC xor (Data shl 8);
for Loop:= 0 to 7 do
begin
  if (newCRC and $8000)<>0 then
    newCRC := (newCRC shl 1) xor $1021
  else
    newCRC := (newCRC shl 1);
end;
```

Pronto learned code format

When using 'irlearn' and 'irstart' data is sent/received from the Pronto in specific formats. All formats use the same structure for their data. All data is grouped into pairs of one word (two bytes). The generic structure of these formats is:

IIII CCCC 0000 RRRR xxxx xxxx ....

Where

IIII	The format identifier
CCCC	The carrier frequency Frequency = 1000000/(CCCC * 0.241246) CCCC = (1000000 / 0.241246) / Frequency = 4145146 / Frequency or the period time (IIII='0100') Period = 1 / Frequency Period = (CCCC * 0.241246) / 1000000 CCCC = (Period * 0.241246) / 1000000 = Period / 4145146
0000	Number of burst pairs (once code). The 'once code' is used when a single code is to be transmitted.
RRRR	Number of burst pairs (repeat code). The 'repeat code' is used when a code is to be repeatedly send. This happens when you keep a key pressed.
xxxx xxxx	First burst pair. A burst pair is a sequence of two words of data.
....	Next burst pair.

All numbers are made up of 4 digits and are in hexadecimal. This means that if you see the number '1234' this actually means $1234 (which is 4660 in decimal notation).

Each burst pair consist of two numbers:

AAAA IIII

Where

AAAA	The number of active cycles (or periods)
IIII	The number of in-active cycles (or periods)

First the burst pairs belonging to the 'once code' are placed followed by the burst pairs for the 'repeat code'. If the number of burst pairs for the 'once code' or 'repeat code' is zero then no burst pairs are used for those.

This indicates how long a signal is sending the carrier frequency (AAAA) and how long it is not (IIII). Data transmission is done by activating and de-activating the carrier frequency. The way this carrier frequency is used determines the remote control data (the type of remote control data).

The format identifier indicates what kind of information is contained in the burst pairs. Here are some of the formats used by the Pronto:

0000	Learned code format (modulated)
0100	Learned code format (unmodulated)
5000	RC5 code format
5001	RC5x code format
6000	RC6 code format
6001	RC5 mode A code format

Other formats exist but are typically database 'formats' and are not discussed here.

All but the '0000'/'0100' formats are Pronto proprietary formats and are discussed in the next chapter. We go into some detail on the learned code format here ('0000'/'0100'). Because there are other documents on the learned code available, we won't go into much detail here but just give a summary. Lets explain it using some examples.

0000 Learned code format (modulated)

```
   0000 0073 0001 0002 0010 0020 0010 0010 0030 0040
= IIII CCCC OOOO RRRR 1111 1111 2222 2222 3333 3333
```

This means:

IIII	Format = '0000' is 'modulated learned code format'
CCCC	Period time = $0073 = 115 == 36045 Hz
OOOO	There is $0001 = 1 once code burst pairs
RRRR	There are $0002 = 2 repeat code burst pairs
1111 1111	The first burst pair (belongs to the once code burst pairs)
	0010 0020 indicates that the carrier signal is on for $0010 = 16 cycles and off for $0020 = 32 cycles
2222 2222	The second burst pair (1st burst pair for the repeat code burst pairs)
	0010 0010 indicates that the carrier signal is on for $0010 = 16 cycles and off for $0010 = 16 cycles
3333 3333	The thirst burst pair (2nd burst pair for the repeat code burst pairs)
	0030 0040 indicates that the carrier signal is on for $0030 = 48 cycles and off for $0040 = 64 cycles

Other examples:

```
   0000 0073 0000 0002 0010 0020 0010
= IIII CCCC OOOO RRRR 1111 1111 2222
```

Here there are no burst pairs for the 'once code', only burst pairs for the 'repeat code'.

Other examples:

```
   0000 0073 0002 0000 0010 0020 0010
= IIII CCCC OOOO RRRR 1111 1111 2222
```

Here there are no burst pairs for the 'repeat code', only burst pairs for the 'once code'.

0100 Learned code format (unmodulated)

```
   0100 0073 0001 0002 0010 0020 0010 0010 0030 0040
= IIII CCCC OOOO RRRR 1111 1111 2222 2222 3333 3333
```

This means:

IIII	Format = '0100' is 'unmodulated learned code format'
CCCC	Period (1/ Frequency) = $0073 = 115 == 36045 Hz == 0.000027743 s == 27.743 us
OOOO	There is $0001 = 1 once code burst pairs
RRRR	There are $0002 = 2 repeat code burst pairs
1111 1111	The first burst pair (belongs to the once code burst pairs) 0010 0020 indicates that the IR signal is on for $0010 = 16 periods and off for $0020 = 32 periods
2222 2222	The second burst pair (1st burst pair for the repeat code burst pairs) 0010 0010 indicates that the IR signal is on for $0010 = 16 periods and off for $0010 = 16 periods
3333 3333	The thirst burst pair (2nd burst pair for the repeat code burst pairs) 0030 0040 indicates that the IR signal is on for $0030 = 48 periods and off for $0040 = 64 periods

Pronto proprietary remote control formats

The Pronto has different ways of using remote control commands. The simplest way is to learn the code from the original remote control and use this learned code. Another method is using codes from the databases (which probably are just pointers to learned code). Besides these, the Pronto also has some proprietary formats you can choose from. These are the RC5, RC5X, RC6 and RC6A formats.

Most noticeable is that, when you define a remote control code using these proprietary formats, that it is made up of much less information than a learned code needs.

Below we will go into more detail on what these proprietary formats are made of.

RC5

Generates the following code:

```
5000 0000 0000 0001 SSSS CCCC
```

5000	Identifier for the RC5 format
0000	Initially 0000, but when reviewing later it can contain the carrier frequency
0000	Always 0000, indicating no 'one time only' code data
0001	Always 0001, indication one 'repeat' code data follows (i.e. SSSS CCCC)
SSSS	The SYSTEM identifier: 0..31
CCCC	The COMMAND identifier: 0..127

The RC5 format itself is defined as follows (represented as a bit stream):

ss T SSSSS CCCCCC

ss = 10	Add 64 to command
ss = 11	Use command as it is
T	Toggle bit
SSSSS	System bits (5)
CCCCCC	Command bits (6)

All these bits are to be biphase encoded, meaning that a single bit is split up into two half bits:

0 -> 10

1 -> 01

RC5X Generates the following code:

```
5001 0000 0000 0002 SSSS CCCC DDDD 0000
```

5001	Identifier for the RC5X format
0000	Initially 0000, but when reviewing later it can contain the carrier frequency
0000	Always 0000, indicating no 'one time only' code data
0002	Always 0002, indication two 'repeat' code data follows (i.e. SSSS CCCC DDDD 0000)
SSSS	The SYSTEM identifier: 0..31
CCCC	The COMMAND identifier: 0..127
DDDD	The DATA identifier: 0..63

The RC5X format itself is defined as follows (represented as a bit stream):

ss T SSSSS dddd CCCCCC DDDDDD

ss = 10	Add 64 to command	(to be biphase encoded)
ss = 11	Use command as it is	(to be biphase encoded)
T	Toggle bit	(to be biphase encoded)
SSSSS	System bits (5)	(to be biphase encoded)
dddd	Divider bits (4) '0000'	(NOT to be biphase encoded)
CCCCCC	Command bits (6)	(to be biphase encoded)
DDDDD	Data bits (6)	(to be biphase encoded)

All bits to be biphase encoded will have a single bit split up into two half bits:

0 -> 10

1 -> 01

RC6 Generates the following code:

```
6000 0000 0000 0001 SSSS CCCC
```

6000	Identifier for the RC6 format
0000	Initially 0000, but when reviewing later it can contain the carrier frequency
0000	Always 0000, indicating no 'one time only' code data
0001	Always 0001, indication one 'repeat' code data follows (i.e. SSSS CCCC)
SSSS	The SYSTEM identifier: 0..255
CCCC	The COMMAND identifier: 0..255

The RC6 format itself is defined as follows (represented as a bit stream):

hhhhhhhh TT SSSSSSSS CCCCCCCC

hhhhhhhh	Header data (16) '1111110010010101'	(NOT to be biphase encoded)
TT	Toggle bits (4) '0011' or '1100'	(NOT to be biphase encoded)
SSSSSSSS	System bits (8)	(to be biphase encoded)
CCCCCCCC	Command bits (8)	(to be biphase encoded)

All bits to be biphase encoded will have a single bit split up into two half bits:

0 -> 01

1 -> 10 Note: inverse from RC5/RC5X biphase format!

Note: a RC6 bit transmission takes half the time of a RC5/RC5X bit transmission!

RC6A Generates the following code:

```
6001 0000 0000 0002 UUUU SSSS CCCC 0000
```

6001	Identifier for the RC6A format
0000	Initially 0000, but when reviewing later it can contain the carrier frequency
0000	Always 0000, indicating no 'one time only' code data
0002	Always 0002, indication two 'repeat' code data follows (i.e. UUUU SSSS CCCC 0000)
UUUU	The CUSTOMER identifier: 0..127 and 32768..65535
SSSS	The SYSTEM identifier: 0..255
CCCC	The COMMAND identifier: 0..255

The RC6A format itself is defined as follows (represented as a bit stream):

hhhhhhhh TT s UUUUUUU SSSSSSSS CCCCCCCC or
hhhhhhhh TT s UUUUUUUUUUUUUUU SSSSSSSS CCCCCCCC

hhhhhhhh	Header data (17) '11111110010101001'	(NOT to be biphase encoded)
TT	Toggle bits (4) '0011' or '1100'	(NOT to be biphase encoded)
s = 0	Customer range 0..127	(to be biphase encoded)
s = 1	Customer range 32768..65535	(to be biphase encoded)
UUUUUUU	CUSTOMER identifier (7 or 15) 7 or 15 bits depending on the 's' bit	(to be biphase encoded)
SSSSSSSS	System bits (8)	(to be biphase encoded)
CCCCCCCC	Command bits (8)	(to be biphase encoded)

All bits to be biphase encoded will have a single bit split up into two half bits:

0 -> 01
1 -> 10 Note: inverse from RC5/RC5X biphase format!

Note: a RC6A bit transmission takes half the time of a RC5/RC5X bit transmission!

Note: this '6001 ..' sequence is actually never used by the Pronto. ProntoEdit does not send the '6001 ..' sequence but instead uses the learned code format!

Example converting an RC5X command into a Pronto learned code format

```
RC5X 31 127 63       -> RC5X, System=31, Command=127, Data = 63

Pronto format: 5001 0000 0000 0002 001F 007F 003F 0000


Bitstream:
  ss T SSSSS dddd CCCCCC DDDDDD

  ss = 10   Add 64 to command      (To be biphase encoded)
  ss = 11   Use command as it is   (To be biphase encoded)
  T         Toggle bit             (To be biphase encoded)
  SSSSS     System bits (5)        (To be biphase encoded)
  dddd      Divider bits (4)       (NOT to be biphase encoded)
            '0000'
  CCCCCC    Command bits (6)       (To be biphase encoded)
  DDDDDD    Data bits (6)          (To be biphase encoded)

  ALL bits are biphase encoded, except for the dddd bits:
  0 -> 10
  1 -> 01

  ss   T  SSSSS      dddd CCCCCC       DDDDDD
  10   0  11111      0000 111111       111111        (Not yet biphase encoded)
  0110 10 0101010101 0000 010101010101 010101010101  (Biphase encoded, except for dddd)

  If this were to be translated into a learned Pronto command then we would seperate them in
  1->0 transitions and then count the consecutive one and zeroes:
  0110 10 0101010101 0000 010101010101 010101010101              (Original bitstream)
  0110  100  10   10   10   10   100000  10   10   10   10   10   10   10   10   10   10   10   1    (Spaced by 1->0 transitions)
  2 1   1 2  1 1  1 1  1 1  1 1  1 5     1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 ?  (Counted)

  These counted numbers are then converted into timing numbers. This means that they will be related
  to the carrier frequency. Assuming our unit base is 0010 (hex) then we have to multiply this base with
  the counter numbers

  2 1         1 2        1 1        1 1        1 1        1 1    1 5     1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 1  1 ?
  0040 0020   0020 0040  0020 0020  0020 0020  0020 0020  0020 0020  0020 00A0    etc

  The last, implicit, 0 timing is usually long enough time to separate individual command (ie. 0700).
  The complete result (including carrier leader and such) would be:

  0000 006D 0013 0000 0040 0020 0020 0040 0020 0020 0020 0020 0020 0020 0020 0020 0020 00A0 0020 0020 0020
  0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0020 0700
```

Example converting an RC6A command into a Pronto learned code format

```
RC6A 41251 1 1       -> RC6A, Customer code=41251, System=1, Command=1

Pronto format (hex): 6001 0000 0000 0002 A123 0001 0001 0000


Bitstream:
  hhhhhhhh TT s UUUUUUU SSSSSSSS CCCCCCCC
  hhhhhhhh TT s UUUUUUUUUUUUUUU SSSSSSSS CCCCCCCC

  hhhhhhhh  Header data (17)
            '11111110010101001'  (NOT to be biphase encoded!)
  TT        Toggle (2/4)
            '0011' or            (NOT to be biphase encoded!)
            '1100'               (NOT to be biphase encoded!)
  s = 0     Custumer code 0..127       (To be biphase encoded)
  s = 1     Custumer code 32768..65536 (To be biphase encoded)
            Add 32768 to customer code
  UUUUUUU   Customer Code (7/15)       (To be biphase encoded)
            These are either 7 or 15 bits depending on the 's' size bit
  SSSSSSSS  System bits  (8)           (To be biphase encoded)
  CCCCCCCC  Command bits (8)           (To be biphase encoded)

  ONLY system and command bits are biphase encoded:
  0 -> 01
  1 -> 10               Note the inverted nature in comparison with RC5


  hhhhhhhh          TT   s  UUUUUUUUUUUUUUU                SSSSSSSS         CCCCCCCC
  11111110010101001 0011 1  010000100100011                00000001         00000001          (Not yet biphase encoded)
  11111110010101001 0011 10 01100101010110010110010101011010 0101010101010110 0101010101010110  (Biphase encoded)
  111111100 10 10 100 100 11100 1100 10 10 10 1100 10 1100 10 10 110 100 10 10 10 10 10 10 10 1100 10 10 10 10 10 10 110   (1->0 seperated)
  etcetera

  Note that, with reference to the RC5X generated timing, the timings will be halved:
  RC5X:  0000 006D xxxx 0000 0040 0020 0020 0040 0020 0020 ...
  RC6A:  0000 006D xxxx 0000 0020 0010 0010 0020 0010 0010 ...
```

Pronto hardware

For those especially interested in the hardware of the Pronto, these are the main components the Pronto is made of:

- Motorola DragonBall MC68328PV16VA Microprocessor
- Samsumg (SEC) KM616V4000BLT-7L RAM, 256k x 16 bit, 3-3.6V, 70ns
- AMD AM29LV800BB-70EC Flash, 1M x 8 bit (512k x 16 bit), 3V, boot sector, 70ns
- Maxim MAX3221 RS232, 3-5.5V
- Touchscreen from Samsung

ISO 8601

The following is a description of the ISO (International Standards Organization) standard for numerical date and time interchange formats. The standard, 8601:1988, supersedes the ISO standards: 2014, 2015, 2711, 3307 and 4031. A copy of the actual standard may be obtained from ISO. The following is ONLY a description of this standard. If your organization is performing or considering performing any type of electronic data exchange you should obtain a copy of the standard.

General Description

ISO 8601 defines formats for the representation of dates, times and date/time combinations. Both Local Time and Coordinated Universal Time (UTC) are supported by ISO 8601. Dates are for the Gregorian calendar and can be given in year-month-day, year-week-day or year-day formats. A 24-hour format is used to represent Time.

All date and time formats are represented with the largest units given first. The raking of these units is years, months, weeks, days, hours, minutes and then seconds. Specific subsets are specified in the standard. An optional hyphen, "-", character used for separation.

Calendar Date Identification

Calendar dates are identified by year number (0-9999), month number (1-12) and a day number (1-31), based on the Gregorian calendar standard.

Leap Year Identification

A leap years are identified by the following rules:

- The year number is evenly divisible by four, unless
- The year number is evenly divisible by 100, unless
- The year number is evenly divisible by 400

Leap years have an additional day during February, resulting in a 366-day year.

Ordinal Date Identification

Ordinal dates are identified by a year number (0-9999) and a day number (0-365, or for a leap year 0-366).

Week Identification

A week is identified by a number between one and 52. The standard defines a week as starting Monday and ending Sunday. The days of a week are Monday, Tuesday, Wednesday, Thursday, Friday, Saturday and Sunday. The first week of a year is the one that includes the first Thursday, or equivalently the one that includes January 4.

Time Identification

Time is represented by an hour number (0-24), a separation character, ":", a minute number (0-59), another separation character, ":", and the second number (0-59). Midnight may be expressed as either 00:00:00 or 24:00:00.

Value Representation

Unless otherwise stated, all values are fixed width. Leading zeros are used to fill empty digits.
Alternative value representation is allowed with mutual agreement and based on other ISO standards.

Date Formats

The various accepted date formats are described below in this section.

Note: All the examples shown below use the date, December 08, 1997, and the time, 14:07:01 (time is based on a 24-hour clock). The ordinal day number for the example date is 343, the week number is 49 and the week day number is 1.

Calendar Date Formats

The following complete, abbreviated or truncated formats are permissible:

- "19971208" or "1997-12-08" (complete representation)
- "1997-12" (reduced precision)
- "1997"
- "19"
- "971208" or "97-12-08" (truncated, current century assumed)
- "-9712" or "-97-12"
- "-97"
- "–1209" or "–12-09"
- "–12"
- "—09"

Ordinal Date Formats

The day number within a given year can be expressed as:

- "1997343" or "1997-343" (complete representation)
- "97343" or "97-343"
- "-343"

Week/Day Formats

Dates with a given week number may be expressed as:

- "1997W491" or "1997-W49-1" (complete representation)
- "1997W49" or "1997-W49"
- "97W491" or "97-W49-1"
- "97W49" or "97-W49"
- "-7W491" or "-7-W49-1"
- "-W491" or "-W49-1"
- "-W49"
- "-W-1" (day of current week)
- "—1" (day of any week)

Time Formats

Time may be represented in local time and fractional local time formats. When used without the date, the time value should be preceded by a "T" to avoid confusion with date values. The rules for date/time formats forbid the replacing of leading time values with hyphens.

Note: As noted above, the time used for the examples given is 14:07:01, which is based on a 24-hour clock.

Local Time of Day

Local time of day may be expressed as:

- "140701" or "14:07:01" (complete representation)
- "1407" or "14:07"
- "14"
- "-0701" or "-07:01"
- "-07"
- "–01"

Fractional Local Time of Day

Decimal fractions may be included with an hour, minute or second. The decimal sign should be either a comma (preferred) or a full stop. If the value is less than one then the decimal sign should be preceded by a zero. The number of decimal places is set depending on the application.

The following formats are permitted (given two decimal places):

- "140701,02" or "14:07:01,02"
- "1407,12" or "14:07,12"
- "14,5"
- "-0701,02" or "-07:01,02"
- "-07,12"
- "–01,02"

Coordinated Universal Time (UTC)

Time can be expressed in UTC by appending the symbol "Z" , without spaces to any of the local time or fractional local time formats.

For example:

- "Z140701"
- "Z14,5"

Difference between Local and UTC Times

The relationship of a local time to UTC can be expressed by appending a time zone indicator without spaces to the right-hand side of the local time representation, which must include hours.

Omitting the minutes implies a lower precision for the time zone value, and is independent of the precision of the time value to which the zone is attached. Time zones behind UTC use the "-" sign.
The standard implies (but does not state explicitly) that the extended zone format ("14:00") is used with extended format times, and the basic zone format ("1400") with basic format times.

Combined Date/Time Formats

The symbol "T" is used to separate the date and time parts of the combined representation. This may be omitted by mutual consent of those interchanging data, if ambiguity can be avoided.

The complete representation is as follows:

- "19971208T140701" or "1997-12-08T14:07:01"

The date and/or time components independently obey the rules already given in the sections above, with the restrictions:

- The date format should not be truncated on the right (i.e., represented with lower precision) and the time format should not be truncated on the left (i.e., no leading hyphens).

- When the date format is truncated on the left, the leading hyphens may be omitted.

Periods of Time

There are four basic representations allowed for specifying periods of time. Where two time values are required, they are separated with a forward slash, "/"; a double hyphen may be used in certain application areas. The dates are given in calendar form, but ordinal or week/day may alternatively be used. Basic representation can be replaced by the appropriate extended representation.
Likewise, valid reduced precision, truncated or decimal formats may be used.

Specific Start and Specific End

Date and Time start and end periods may be represented using the following format:

- "19971209T140701/19971209T223012"

Note: If the higher order components of the second period are omitted, the corresponding values from the first period are used. Likewise, if a time zone is supplied for the first period but not the second, it is assumed to be used for both.

Periods of Time, no Specific Start or End

When representing a period of time with no specific start or end then variable width values are used.
The value starts with "P", and is followed by a list of periods, each appended by a single letter designator: "Y" for years, "M" for months, "D" for days, "W" for weeks, "H" for hours, "M" for minutes, and "S" for seconds. Time components must be preceded by the "T" character. There is no extended representation defined for this format.

For example:

A period of 2 years, 5 months, 6 days, 10 hours, 20 minutes and 59 seconds is represented by:

- "P2Y5M6DT10H20M59S"

A period of 30 seconds by:

- "T30S"

A period of 10 days by:

- "P10D"

Periods of Time, Specific Points In Time

A period of time may also be expressed using the format specified for points in time, provided the values do not exceed 12 months, 30 days, 24 hours, 60 minutes, and 60 seconds.

Period with Specific Start

A specific point in time with a specified starting point and duration may be represented by providing a specific date and time with the duration following, as shown below:

- "19971208T140701/P10Y5M20DT10H42M11S"

Period with Specific End

A specific point in time with a specific duration and end point may be represented a duration of time followed by a specific point of time, as shown below:

- "P10Y5M20DT10H42M11S/19971209T140701"