

Document	: GenerateRCx
Author	: M. Majoor
Subject	: Pronto utility to generate/convert remote control data

Revision overview

Revision (date)	Author	Description
20030323	MM	Initial draft

Pronto utility

This utility (GenerateRCx) is a command line based application which can convert between different Pronto formats. Basically the following conversions are available:

Formats	Example	Explanation Note: some remarks might only be applicable to the SBC RU890 model
Short form 1	RC5 0 0	Format used in ProntoEdit. This is typically converted into the short form 2 presentation by ProntoEdit (except for RC6A: short form 2 is not directly supported by the Pronto RU890 so ProntoEdit converts it directly into learned format).
Short form 2	5000 0000 0000 0001 0000 0000	This is the shortest internal representation of a remote control code in the Pronto itself. The other format the Pronto uses is the learned format .
Bitstream	110000000000000	Intermediate, internally used, bit stream of the application. These is the rough bit data (command, system and such converted into bit values). Depending on the type of code (RC5/5X/6/6A) specific bits will be processed differently, resulting in the encoded bitstream .
Encoded bitstream	0101101010101010101010101010	These is the, internally used, bit stream which results in the learned data .
Learned data	0000 0073 0000 000D 0020 0020 0040 0020 09B7	This is the second format the Pronto uses to represent remote control data. The other format it uses is short form 2 . This is the converted encoded bitstream .

The internal conversion which takes place is:

Short form 1 or

Short form 2 -> Bit stream -> Encoded bit stream -> Learned data

Running the application

The application is basically a console application. The application requires parameters which can be passed on to it in two ways:

- By means of the command line.
- Through the clipboard.

By default the application expect parameters passed on by means of command line parameters. If no parameters are passed on then the clipboard is used instead.

When command line parameters are used then the output is send to the console. When the clipboard is used then results are passed on to the clipboard.

The application will only generate a result when the conversion succeeds. When some error is detected then this is indicated in the returned result code:

0 = Success

1 = Number of parameters not correct

1x = Error in parameter x, eg. 12 means that parameter 2 was incorrect

Parameters and formats

Although there are only two presentations of the remote control to be generated, the application allows a number of additional parameters for better control of the generated remote control data.

When only the basic short forms are used then the application uses default settings for generating the correct data. These optional settings are:

- Carrier frequency to use (default 36000 Hz)
- Toggle state (default off)
- 'Repeat' code or 'once' code (default on!)
- Delay (off-time) following the remote control data (default 69 ms)
- Output format (default 0 == learned code)

The simplest form of the parameters are the direct input of either **short form 1** or **short form 2**:

GenerateRCx RC5 0 0

Short form 1 command line example

GenerateRCx 5000 0000 0000 0001 0000 0000

Short form 2 command line example

This will return the **learned code** data in both cases. If another result is required then this can be indicated by adding the required output format parameter:

GenerateRCx RC5 0 0 1

Short form 1 command line example

Here output format '1' will be generated. Since output format '1' is **short form 1** this effectively results in the same data as we wanted to convert. The result in this example would be

RC5 0 0

Parameters Format is always optional								
1	2	3	4	5	6	7	8	9
RC5	Frequency	Delay	Toggle	Repeat	System	Command	<i>Format</i>	
RC5	System	Command	<i>Format</i>					
RC5X	Frequency	Delay	Toggle	Repeat	System	Command	Data	<i>Format</i>
RC5X	System	Command	Data	<i>Format</i>				
RC6	Frequency	Delay	Toggle	Repeat	System	Command	<i>Format</i>	
RC6	System	Command	<i>Format</i>					
RC6A	Frequency	Delay	Toggle	Repeat	Customer	System	Command	<i>Format</i>
RC6A	Customer	System	Command	<i>Format</i>				
5000	0000	0000	0001	ssss	cccc	<i>Format</i>		
5000	ffff	0000	0001	ssss	cccc	<i>Format</i>		
5001	0000	0000	0002	ssss	cccc	dddd	0000	<i>Format</i>

5001	ffff	0000	0002	ssss	cccc	dddd	0000	<i>Format</i>
6000	0000	0000	0001	ssss	cccc	<i>Format</i>		
6000	ffff	0000	0001	ssss	cccc	<i>Format</i>		
6001	0000	0000	0002	uuuu	ssss	cccc	0000	<i>Format</i>
6001	ffff	0000	0002	uuuu	ssss	cccc	0000	<i>Format</i>

Parameters/formats explanation

Bitstream

This is the bit-data without the bi-phase encoding applied to it. This is an intermediate bit stream only. This bit stream will have some, but not necessarily all, bits bi-phase encoded (only the RC5 will have all bits bi-phase encoded). The encoded data is available as <EncodedBitstream>.

Cccc

<Command> for the remote control. Value is a 4 digit hexadecimal number.

Command

<Command> for the remote control.

Customer

<Customer> for the remote control.

Data

<Data> for the remote control.

Dddd

<Data> for the remote control. Value is a 4 digit hexadecimal number.

Delay

Delay in ms to introduce after the infrared command. Required when multiple remote control commands are appended or a repeat code is generated.

EncodedBitstream

This is the bit-data which is used to convert to the actual Pronto format. This is an intermediate bit stream only. The result of this conversion is available as <Learned> data.

Ffff

The carrier frequency, in Pronto units. Value is a 4 digit hexadecimal number.

Conversion from Pronto units to actual frequency can be calculated using the formula:

$$\text{Frequency} = \text{round}(1\text{E}6 / (\text{Ffff} * 0.241246))$$

Format

The output format to generate.

(Default) 0/10 Learned

1/11 ShortformRC

2/12 Shortform

3/13 Bitstream

4/14 EncodedBitstream

The values 10..14 will include the original parameters in the output as well.

Frequency

Frequency in Hz of the generated remote control data (carrier).

Learned

This is the Pronto format for learned remote control data.

Repeat

Indicates if the <Learned> data should indicate a 'once' code data stream or a 'repeat' code data stream. '0' represents inactive, '1' represents active.

Shortform

This is the Pronto format of the proprietary formats of the Pronto. This can be used instead of the <Learned> data.

Note that you should not use this for RC6A remote control codes. Most likely the Pronto will not use it correctly. Use the <Learned> data instead.

ShortformRC

This is the code ProntoEdit uses to represent the current remote control code (eg. RC5 0 0).

System

<System> for the remote control.

Ssss

<System> for the remote control. Value is a 4 digit hexadecimal number.

Toggle

Indicates what state the toggle in the bit stream should be. '0' represents inactive, '1' represents active.

Uuuu

<Customer> for the remote control.. Value is a 4 digit hexadecimal number.

Examples:

GenerateRCx RC5 0 10

Most basic form of short form 1, output format default 0

GenerateRCx RC5 0 10 10

Output format 10 (=0 but with original parameter output)

GenerateRCx RC5 36000 69 0 0 0 10 10

36000 Hz, 69 ms, no toggle, no repeat, output format 10

GenerateRCx 5000 0000 0000 0001 0000 000A

Most basic form of short form 2

GenerateRCx 5000 0000 0000 0001 0000 000A 10

Using default frequency of 36000

GenerateRCx 5000 006B 0000 0001 0000 000A 10

Specific carrier frequency

GenerateRCx RC5X 0 10 20

GenerateRCx RC5X 0 10 20 10

GenerateRCx RC5X 36000 69 0 0 0 10 20 10

GenerateRCx 5001 0000 0000 0002 0000 000A 0014 0000 10

GenerateRCx 5001 006B 0000 0002 0000 000A 0014 0000 10

GenerateRCx RC6 0 10

GenerateRCx RC6 0 10 10

GenerateRCx RC6 36000 69 0 0 0 10 10

GenerateRCx 6000 0000 0000 0001 0000 000A 10

GenerateRCx 6000 006B 0000 0001 0000 000A 10

GenerateRCx RC6A 20 0 10

GenerateRCx RC6A 20 0 10 10

GenerateRCx RC6A 36000 69 0 0 10 0 10 10

GenerateRCx 6001 0000 0000 0002 0014 0000 000A 0000 10

GenerateRCx 6001 006B 0000 0002 0014 0000 000A 0000 10

Note: If using a batch file for converting you can redirect the output to a file by means of the redirection directive '>'. If you want to append the result to a file use '>>' instead. Example of start of a batch file:

```
@echo off
```

```
@rem First clear the resulting file (RESULT.RCX is the file)
```

```
@echo Note: Errors in the original data will generate no result at all! >RESULT.RCX
```

```
@rem Next data is appended
```

```
GenerateRCx RC5 0 0 >>RESULT.RCX
```

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

$$\text{Frequency} = 1000000 / (\text{CCCC} * 0.241246)$$

$$\text{CCCC} = (1000000 / 0.241246) / \text{Frequency} = 4145146 / \text{Frequency}$$

or the period time (IIII='0100')

$$\text{Period} = 1 / \text{Frequency}$$

$$\text{Period} = (\text{CCCC} * 0.241246) / 1000000$$

$$\text{CCCC} = (\text{Period} * 0.241246) / 1000000 = \text{Period} / 4145146$$

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 0000 RRRR 1111 1111 2222 2222 3333 3333

This means:

IIII	Format = '0000' is 'modulated learned code format'
CCCC	Period time = \$0073 = 115 == 36045 Hz
0000	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 thirth 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 0000 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 0000 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 0000 RRRR 1111 1111 2222 2222 3333 3333

This means:

IIII	Format = '0100' is 'unmodulated learned code format'
CCCC	Period ($1/\text{Frequency}$) = $00073 = 115 \Rightarrow 36045 \text{ Hz} \Rightarrow 0.000027743 \text{ s} \Rightarrow 27.743 \text{ us}$
0000	There is $00001 = 1$ once code burst pairs
RRRR	There are $00002 = 2$ 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 $00010 = 16$ periods and off for $00020 = 32$ 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 $00010 = 16$ periods and off for $00010 = 16$ periods
3333 3333	The thirist burst pair (2nd burst pair for the repeat code burst pairs) $0030 0040$ indicates that the IR signal is on for $00030 = 48$ periods and off for $00040 = 64$ periods

Pronto proprietary remote control formats

The Pronto supports besides the learned format also it's proprietary formats. These formats are RC5, RC5x, RC6 and RC6A. The next tables show what these formats are made up and how they can be used to generate learned code.

The carrier in the 'shortform' is the same as that for the learned code format. The same calculations apply for it.

RC5	ShortformRC	RC5 S C				
	Shortform	5000 0000 0000 0001 SSSS CCCC or 5000 FFFF 0000 0001 SSSS CCCC where FFFF = carrier				
	Bitstream	ss T SSSSS CCCCCC				
Notes	Carrier	==	36000 .. 38000 kHz			
	Halfbit time	==	889 us			
ID	Bits	Descriptor	Range		'0' encode	'1' encode
ss	2	Start	ss = '10'	Command range 64 .. 127	'10'	'01'
			ss = '11'	Command range 0 .. 63		
T	1	Toggle			'10'	'01'
SSSSS	5	System	0 .. 31		'10'	'01'
CCCCC	6	Command	0 .. 63	ss = '11'	'10'	'01'
			64 .. 127	ss = '10'		

RC5x	ShortformRC	RC5x S C D						
	Shortform	5001 0000 0000 0002 SSSS CCCC DDDD 0000 5001 FFFF 0000 0002 SSSS CCCC DDDD 0000						or where FFFF = carrier
	Bitstream	ss T SSSSS dd CCCCCC DDDDDD						
Notes	Carrier ==	36000 .. 38000						
	Halfbit time ==	889 us						
ID	Bits	Descriptor	Range				'0' encode	'1' encode
ss	2	Start	ss = '10'	Command range 64 .. 127			'10'	'01'
			ss = '11'	Command range 0 .. 63				
T	1	Toggle					'10'	'01'
SSSSS	5	System	0 .. 31				'10'	'01'
dd	2	Divider	'00'				'00'	'11'
CCCCCC	6	Command	0 .. 63	ss = '11'			'10'	'01'
			64 .. 127	ss = '10'				
DDDDD	6	Data	0 .. 63				'10'	'01'

RC6	ShortformRC	RC6 S C			
	Shortform	6000 0000 0000 0001 SSSS CCCC or 6000 FFFF 0000 0001 SSSS CCCC where FFFF = carrier			
	Bitstream	hhhhhhhh TT SSSSSSSS CCCCCCCC			
Notes	Carrier	==	36000 .. 38000		
	Halfbit time	==	445.5 us		
ID	Bits	Descriptor	Range	'0' encode	'1' encode
hhhhhhhh	8	Header		'1111110010010101'	
TT	2	Toggle	TT = '01' TT = '10'	'00'	'11'
SSSSSSSS	8	System	0 .. 255	'01'	'10'
CCCCCCCC	8	Command	0 .. 255	'01'	'10'

RC6A	ShortformRC	RC6A U S C				
	Shortform	6001 0000 0000 0002 UUUU SSSS CCCC 0000	or			
		6001 FFFF 0000 0002 UUUU SSSS CCCC 0000	where FFFF = carrier			
	Bitstream	hhhhhhhhh TT s UUUUUUU SSSSSSS CCCCCCC	or			
		hhhhhhhhh TT s UUUUUUUUUUUUUUU SSSSSSS CCCCCCC				
Notes						
	Carrier	==	36000 .. 38000			
	Halfbit time	==	445.5 us			
	This '6001 ..' sequence is actually never used by the Pronto itself. ProntoEdit does not send the '6001 ..' sequence but instead uses the learned code format ('0000 ..')!					
ID	Bits	Descriptor	Range		'0' encode	'1' encode
hhhhhhhhh	9	Header			'011111110010101001'	
TT	2	Toggle	TT = '01'		'00'	'11'
			TT = '10'			
s	1	Range	s = '0'	Customer range 0 .. 127	'01'	'10'
			s = '1'	Customer range 32768 .. 65535		
UUUUUUUU	7	Customer	0 .. 127	s = '0'	'01'	'10'
	15		32768 .. 65535	s = '1'		
SSSSSSSS	8	System	0 .. 255		'01'	'10'
CCCCCCCC	8	Command	0 .. 255		'01'	'10'

Converting to learned code

If a learned code format is to be generated, then the **bitstream** is used as the source. This bitstream is first translated into a **halfbit** bitstream using the 'encode' data (thus each bit in the bitstream is translated into two halfbits). From this encoded data the bits are grouped into groups of one's and zeroes. Then the number of bits in each group are counted. These counts are then translated into timing equivalents (carrier frequency based). Let's explain this with an example:

```
ShortformRC:      RC5 5 0                      System = 5      Command = 0
Bitstream:        ss T SSSSS CCCCCC
                  11 0 00101 000000
                  1 1 0 0 0 1 0 1 0 0 0 0 0 0

Halfbit bitstream:
1 1 0 0 0 1 0 1 0 0 0 0 0 0 (bitstream)
01 01 10 10 10 01 10 01 10 10 10 10 10 10 (halfbit encoded)
0 1 0 11 0 1 0 1 00 11 00 11 0 1 0 1 0 1 0 1 0 1 0 (grouped)
1 0 11 0 1 0 1 00 11 00 11 0 1 0 1 0 1 0 1 0 1 0 (-leading 0)
1 1 2 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 (counted)
```

Using 36000 kHz as carrier and 889 us as halfbit timing we calculate that a halfbit is equal to $889\text{E-}6 / (1/36000) = 32$ cycles (\$20). We now have to multiply our counted numbers with this value to get the learned values.

```
1 1 2 1 1 1 1 2 2 2 2 1 1 1 1 1 1 1 1 1 1 1 (counted)
(* $20) 20 20 40 20 20 20 20 40 40 40 40 20 20 20 20 20 20 20 20 20 20 20
```

the learned data now becomes (including header and such):

```
0000 0073 000B 0000 20 20 40 20 20 20 20 40 40 40 40 20 20 20 20 20 20 20 20
```

Note: We have to make sure that we always start the encoded data with a '1' and end the encoded data with a '0'. This can simply be done by removing leading zeroes and pad it with a zero. Usually we pad this with a number of zeroes which represent the delay we would like if we were to send the code repeatedly.

Alternatively we can always end the encoded data with a '1' and add the delay implicitly (easier).

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 dd CCCCC DDDDD

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)
dd	Divider bits (2)	(NOT to be biphase encoded)
	'00'	
CCCCC	Command bits (6)	(To be biphase encoded)
DDDDDD	Data bits (6)	(To be biphase encoded)

ALL bits are biphase encoded, except for the dd bits:

$$0 \rightarrow 10$$

1 \rightarrow 01

ss	T	SSSSS	dd	CCCCC	DDDDDD	
10	0	11111	00	111111	111111	(Not yet biphas 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 separate them in

1-0 transitions and then count the consecutive one and zeroes:

[illegible]

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:

[illegible]

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 UUUUUUUU SSSSSSSS CCCCCCCC

hhhhhhhh TT s UUUUUUUUUUUUUUUU SSSSSSSS CCCCCCCC

hhhhhhhh Header data (18)

'011111110010101001' (NOT to be biphase encoded!)

TT Toggle (2/4)

'01' or (NOT to be biphase encoded!)

'10' (NOT to be biphase encoded!)

s = 0 Customer code 0..127 (To be biphase encoded)

s = 1 Customer 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

SSSSSSS System bits (8) (To be biphase encoded)

CCCCCCC 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

----- 01 1 010000100100011 00000001 00000001 (Not yet biphase encoded)

011111110010101001 0011 10 011001010101100101100101011010 01010101010110 01010101010110 (Biphase encoded)

11111110 10 10 100 100 11100 1100 10 10 10 1100 10 1100 10 10 110 100 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 ...