COOLPIX REMOTE CONTROL PROTOCOLS

Why and who

Having a look at this document, some of you would say “Oh no, another dummy that stole and use others’ works!”. Not at all, indeed! The reason why I have written this document is that I have tried to write my own Coolpix remote control software on Palm platform since the old one won’t work anymore on new devices. Thus my research on the net leads me to found two web site dealing with remote control protocol : the one of Vladimir Vyskocil for the MC-EU1 one and the one of Eugene Crosser for the general camera protocol. They must be really greaten here.

Then it is well known that information on the WWW can disappear as quick as it appear : that is why I have written and shared this document. It is mainly based on a cut and paste from the previously quoted web sites but I have corrected some little mistakes and added some information that I found by analysing the Coolpix 995 serial protocol. Finally, I have also had things that I found useful for writing my software.

This document is free and for personal use. It cannot be sell or used to design a commercial software without my permission. The last version can be found at http://f4hla.free.fr .

On this last web site you will find CoolPalm© that is a software design with the use of this document for remotely controlled Coolpix camera from a Palm computing platform that has been successfully tested with a Nikon Coolpix 995 and a Palm Tungsten. This program is free but offers no guaranty and the author cannot be responsible for damages caused to the Palm or to the camera.

I hope you will enjoy the use both of this document and program as much as I enjoy doing them.

Regards

Gilles, f4hla

Coolpix MC-EU1 Protocol

Introduction

Here are some informations about the serial protocol used between the Coolpix and the MC-EU1 remote.

Speed is set to 19200 bauds.

This protocol use 1 or 4 bytes "packets". 4 bytes packets are used to send command to the Coolpix and receive information from it.

1 byte packet are used as acknowledge (0x86), not acknowledge (0x15), "attention packet" (0xFF).

Each byte in these packet is divided in two part, bit 0-6 is the value on 7 bits and bit 7 is the odd parity bit (thanks Mark Roberts !)

Returned value by the coolpix (in the two or three? last byte) have 0x1C offset, and least significant byte is sent first, for example the number of picture is :

nb picture = ((byte[2] & 0x7F) - 0x1C) + 100 * ((byte[3] & 0x7F) - 0x1C)
byte[0] == 0x9B, byte[1] == 0x10

Commands

Send	Receive	Info
Going to MC-EU1 protocol		
0x00	0x15	
0x1B,0x53,0x06,0x00,0x00,0x11 0x02,0x00,0x00,0x00,0x13,0x00	0x06	Standard SetSpeed to 19200 command packet
0x1B,0x53,0x06,0x00,0x00,0x11 0x02,0x00,0x00,0x10,0x23,0x00	0x06	Go to MC-EU1 protocol packet, switch LCD ON
0x9B, 0x85, 0x1C, 0x1C	0x9B, 0x13, 0x1C, 0x1C	Magic init string
0x86		
Has camera power down ?		
0x9B, 0x08, 0x1C, 0x1C	0x9B, 0x92, 0x7F, 0x7F	Coolpix is ON
	0x9B, 0x19, 0x7F, 0x7F	Coolpix powerdown
0x86		
Take a shot		
0x9B, 0x01, 0x1C, 0x1C	0x86	Half press the shutter button
0x9B, 0x01, 0x7F, 0x1C	0x86	Full press the shutter button, take the picture
0x9B, 0x01, 0x7F, 0x7F	0x8F	Release the shutter button, usefull in bulb mode
0x9B, 0x01, 0x1C, 0x7F	0x86	Half press release (unlock shutter button)
Zoom IN		
0x9B, 0x02, 0x1C, 0x1C	0x86	"Press" the zoom in button
0x9B, 0x02, 0x1C, 0x7F	0x86	"Release" the zoom in button
Zoom OUT		
0x9B, 0x02, 0x7F, 0x1C	0x86	"Press" the zoom out button
0x9B, 0x02, 0x7F, 0x7F	0x86	"Release" the zoom out button
Next picture		
0x9B, 0x04, 0x1C, 0x1C	0x86	"Press" right
0x9B, 0x04, 0x1C, 0x7F	0x86	"Release" right
Previous picture		
0x9B, 0x04, 0x7F, 0x1C	0x86	"Press" left
0x9B, 0x04, 0x7F, 0x7F	0x86	"Release" left
Number of picture left		
0x9B, 0x07, 0x1C, 0x1C	0x9B, 0x10, 0xXX, 0xYY	(0xXX & 0x7F) - 0x1C + 100 * ((0xYY & 0x7F) - 0x1C) is the number of pictures left
0x86		

A-REC/M-Rec		
0x9B, 0x89, 0x1C, 0x1C	0x9B, 0x91, 0x9D, 0x1C	A Rec mode
	0x9B, 0x91, 0x0D, 0x7F	M Rec mode
	0x9B, 0x91, 0x7F, 0x9D	Play mode
0x86		
Go back to standard protocol		
0x9B, 0x8A, 0x1C, 0x1C	0x86	LCD OFF

Notes

- At first connection after Coolpix has been powerup, first 0x00 don't work, Coolpix respond 0xFF 0xFF then nothing. A second 0x00 do the job.
- "Attention" packet 0xFF from Coolpix

Coolpix send 0xFF in many cases :

- After initialisation in response to 0x00.
- When Coolpix self powerdown (powersave)
- When Coolpix is powerdown
- When a picture has been recorded
- When mode selector is operated (A-Rec, M-Rec, Play)

- It's a good idea to send 0x9B, 0x08, 0x1C, 0x1C packet to check if Coolpix has powerdown when a 0xFF is received, else check the picture number and current mode.
- Camera send NAK (0x15) in response to bad command packet.
- All the four step, in previous order must be followed in order to take one shot.

Standard protocol

Introduction

Several models of digital cameras, namely Epson, Sanyo, Agfa and Olympus cameras, seem to use the same protocol for communication with the host. Follows the description of the high-level protocol they use over the serial line.

The host and the camera exchange with data packets and individual bytes. Serial line paramaters used are: 8bit, no parity. No flow control is used. All arithmetic data is transmitted least significant byte first ("little endian").

Protocol elements

The elementary units of the protocol are:

Initialisation Byte	NUL	0x00
Action Complete Notification	ENQ	0x05
Positive Acknowledgement	ACK	0x06
Unable to Execute Command	DC1	0x11
Negative Acknowledgement, also Camera Signature	NAK	0x15
Packet	Variable length sequence of bytes	
Termination Byte		0xff

Packet structure

The packet has the following structure:

Offset	Length	Meaning
0	1	Packet type
1	1	Packet subtype/sequence
2	2	Length of data
4	variable	Data
variable	2	Checksum

Known packet types are:

Type	Description
0x02	Data packet that is not last in sequence
0x03	Data packet that is last in sequence
0x1b	Command packet
0x9b	Nikon MC-EU1 protocol

Data packets that are sent in response to a single command are numbered starting from zero. If all requested data fits in one packet, it has type 0x03 and sequence 0.

Command packet has subtype 0x43 or 0x53. Only the first command packet in a session has subtype 0x53.

Maximum length of data field in a packet is 2048 bytes, which yields in 2054 total packet length.

Checksum is a simple 16 bit arithmetic sum of all bytes in the data field. As already mentioned above, length and checksum values are transmitted least significant byte first.

Flow of Control

A communication session flow is as follows:

Host	Camera
Port speed set to 19200 baud	
Host sends init byte 0x00	Camera responds with signature 0x15
Host sends command packet with subtype 0x53 and "set speed" command	Camera sends ACK 0x06

Port speed set to the new value	
Host sends command	Camera responds with either ACK plus optionally "action taken" notifier or data packet sequence
Host sends ACK to every data packet	
... Command - reply cycle repeated ...	
	Camera sends 0xff and resets after a few seconds (value is model-dependant) of inactivity

If the camera does not respond to a command in reasonable time, or responds with a NAK, the command can be resent. If the camera does not provide a complete data packet in reasonable time, or the data packet is corrupt (checksum does not match), the host can request resending of the packet by sending NAK instead of ACK.

Command format and codes (data field)

Command is a sequence of bytes sent in the data field of a command packet. Command format is as follows:

Offset	Length	Description
0	1	Command code
1	1	Register number or subcode
2	variable	Optional argument

Five command codes are known:

Code	Argument	Description
0	int32	Set value of integer register
1	none	Read value of integer register
2	vdata	Take action unrelated to registers
3	vdata	Set value of vdata register
4	none	Read value of vdata register

Commands 0 and 3 are replied with a single ACK 0x06.

Command 2 is replied with an ACK 0x06 followed by an "action complete" notifier 0x05.

Commands 1 and 4 are replied with a sequence of data packets, each of them must be ACK'ed by the host.

Command 0 must be issued with a 4 byte argument containg the new value for the register (bytes in "LSB first" order).

Command 2 typically is issued with a single zero byte as an argument.

Command 3 is issued with an argument of variable number of bytes. If this is a printable string, it should not include the trailing zero byte.

Camera replies to the command 1 with a single data packet containing 4 bytes of a 32bit integer (in "LSB first" order).

Camera replies to the command 4 with a sequence of data packets with variable number of data bytes. Note that if a printable string is returned, it is terminated with a zero byte, and thus may be safely printed or otherwise treated as a normal C language character string.

Registers

The following registers are known (read/writablity info may be inaccurate):

No.	Type	R/W	Description
1	int32	R/W	Resolution (see next table)
2	int32	R/W	Clock in UNIX time_t format
3	int32	R/W	Shutter speed (microseconds), 0 - Auto
4	int32	W	Current frame number (or animation number if hi order byte is 0xff)
5	int32	R/W	Aperture: 0 - Auto, 1 - Low, 2 - Med, 3 - 10 Hi (model dependent)
6	int32	R/W	Color mode: 1 - Color, 2 - B/W
7	int32	R/W	Flash mode: 0 - Auto, 1 - Force, 2 - Off, 3 - Anti Redeye, 4 - Slow sync
8	int32	R/W	Unknown (128)

9	int32	R/W	Unknown (128)
10	int32	R	No. of frames in current folder
11	int32	R	No. of frames left
12	int32	R	Length of current frame *
13	int32	R	Length of current thumbnail *
14	vdata	R	Current frame data *
15	vdata	R	Current thumbnail data *
16	int32	R	Battery capacity percentage
17	int32	R/W	Communication speed 1 - 9600 .. 5 - 115200, 6 - 230400, 256 - 9600 .. 264 - 911600 (sync?)
18	int32	R	Unknown (1)
19	int32	R/W	Bright/Contrast: 0 - Standard, 1 - Contrast+, 2 - Contrast-, 3 - Brighten+, 4 – Brighten
20	int32	R/W	White balance: 0 - Auto, 1 - Sunny, 2 - Incandescent, 3 - Fluorescent, 5 - Flash, 6 - White preset, 255 - Cloudy
21	vdata	R	Unused
22	vdata	R/W	Camera I.D.
23	int32	R/W	Autoshut on host timer (seconds)
24	int32	R/W	Autoshut in field timer (seconds)
25	vdata	R/W	Serial No. (string)
26	vdata	R	Version
27	vdata	R/W	Model
28	int32	R	Available memory left
29	vdata	R/W	Upload image data to this register
30	int32	W	LED: 0 - Off, 1 - On, 2 - Blink
31	vdata	R/W	Unknown ("\0")
32	int32	R/W	Put "magic spell" 0x0FEC000E here before uploading image data
33	int32	R/W	Focus mode: 1 - Macro, 2 - Normal, 3 - Infinity/Fisheye
34	int32	R	Operation mode: 1 - Off, 2 - Record, 3-Play, 6-Thumbnail
35	int32	R/W	LCD brightness 1 to 7
36	int32	R/W	Unknown 1-65535 (3)
37	vdata	R	Unknown ("\0")
38	int32	R	LCD autoshut timer (seconds)
39	int32	R	Protection state of current frame *
40	int32	R	True No. of frames taken
41	int32	R/W	LCD date format: 1 - 'YY MM DD, 2 - DD MM 'HH
42	vdata	R	Unknown ("")
43	vdata	R	Audio data description block * 0: expanded .wav length 1: compressed .wav length 3: Unknown (0) 4: Unknown (0) 5: Unknown (0) 6: Unknown (0) 7: Unknown (0)
44	vdata	R	Audio data *
45	vdata	R	Unknown ("")
46	vdata	R	Camera summary data: 32 bytes with copies of 8 other registers 0: Reg 1 (Resolution) 1: Reg 35 (LCD brightness) or Reg 7 (Flash mode) 2: Reg 10 (Frames taken) or Unknown 3: Unknown (0) 4: Unknown (0) or Reg 16 (Battery capacity) 5: Unknown (0) or Reg 10 (Frames taken) 6: Unknown (0) or Reg 11 (Frames left) 7: Number of animations taken
47	vdata	R	Picture summary data: 32 bytes or 8 int32's * 0: Hi order byte: unknown, next 3 bytes: Length of current image 1: Length of current thumbnail

			2: Audio data length (expanded) 3: Resolution 4: Protection state 5: TimeDate 6: Unknown (0) 7: Animation type: 1 - 10ms, 2 - 20ms
48	vdata	R	Manufacturer
49	vdata	R	Unknown ("")
50	int32	R/W	Unknown (0)
51	int32	R/W	Card detected: 1 - No, 2 - Yes
52	vdata	R	Unknown ("")
53	int32	R/W	Language: 3 - english, 4 - french, 5 - german, 6 - italian, 8 - spanish, 10 - dutch
54	int32	R/W	Unknown (30)
55-58	vdata	R	Unknown ("")
59	int32	R	Unknown (1)
60	int32	R	True No. of frames taken
61-64	vdata	R	Unknown ("")
65	int32	R	Unknown (1)
66-67	vdata	R	Unknown ("")
68	int32	R	Unknown (0)
69	vdata	R/W	Exposure Compensation 8 bytes 0: compensation value -20 to +20 (tenths) 1: 0 2: 0 3: 0 4: 10 5: 0 6: 0 7: 0
70	int32	R/W	Exp. meter: 2 - Center-weighted, 3 - Spot, 5 - Multi element matrix
71	vdata	R/W	Effective zoom in tenths of millimeters: 8 bytes 0: LSB 1: MSB 2: 0 3: 0 4: 10 5: 0 6: 0 7: 0
72	int32	R/W	Bitmap: 1 - AEL/WBL, 2 - Fisheye, 4 - Wide, 8 - Manual zoom, 16 - B/W, 256 - 1.25x, 512 - 1.6x, 768 - 2.0x, 1024 - 2.5x, 1280 - off
73-76	vdata	R	Unknown ("")
77	int32	W	Size of data packet from camera (default 0x800)
78	vdata	R	Unknown ("")
79	vdata	R	Filename of current frame *
80-81	vdata	R	Unknown ("")
82	int32	W	Unknown (enable folder features? Write 60 here)
83	int32	R/W	Folder navigation When read, return number of folders on the card. When written without data, reset folder system (?) Or select current folder by its number
84	vdata	R/W	Current folder name (may read or set)
85-90	vdata	R	Unknown ("")
91	vdata	R	Current folder I.D. and name

* Note: Marked registers only become useful for reading after setting register 4. If value of 0 assigned to register 4 after doing action 5, subsequent retrieval of picture data gives the "live preview".

Resolutions codes must be checked for every kinds of camera but for the Cooplix 995 they are :

Quality\Size	Fine	Normal	Basic
Hi	0x13	0x12	0x11
UXVGA	0x0c	0x0b	0x0a
SXVGA	0x06	0x05	0x04
XVGA	0x09	0x08	0x07
VGA	0x03	0x02	0x01
3:2	0x010	0x0f	0x0e

For command 2, the second byte is action code not register number. The following action codes are known:

Code	Argument	Description
0	single zero byte	Erase last picture
1	single zero byte	Erase all pictures (but not animations)
2	single zero byte	Take picture
3	single byte	
4	single zero byte	Finish session immediately
5	single zero byte	Take preview snapshot (retrievable as frame zero)
6	single byte	Calibration / testing. Arg value: 1 Calibrate autofocus 3 Calibrate white balance 4-6 Store 0 in Reg 32 9 Load LCD Brightness (0-31) from Reg 32 10 Load LCD size (25 for Nikon Coolpix 950) from Reg 32 11 LCD Saturation (0-32) from Reg 32 13 LCD Red-Green (0-32) from Reg 32 14 LCD Blue (0-32) from Reg 32 15 Store -1 in Reg 32 16 Calibrate color 17 Take picture and reset LCD 18 Store -1 in Reg 32 20-23 locks up if lcd is on 24-255 Store -1 in Reg 32
7	single zero byte	Erase current frame *
8	single byte	Switch LCD mode. Arg value: 1 - Off 2 - Record 3 - Play 4 - preview thumbnails (?) 5 - Thumbnail (?) 6 - Thumbnail (?) 7 - Next 8 - Previous
9	single byte	Set protection state of current frame to the value of parameter (binary 0 or 1)*
11	single zero byte	Store freshly uploaded image into NVRAM
12	single byte	LCD test. Arg value: 0 - white 1 - gray 2 - black 3 - red 4 - green 5 - blue 6 - test pattern
16	zero single byte	?Store 1 in Reg 83

* Note: actions 7 and 9 only useful after setting register 0x04.

Example

Finally, if you want to transmit some data with the normal protocol (except special initialisation cases), you should send or receive one of the following sequence:

	Packet type	Packet subtype	Length of data	data	Checksum
Offset	0	1	2	4	4+data
Length	1	1	2 LSB	variable	2 LSB
Send	0x1b	0x43	length of data	Code+Reg/subcode+opt	Σdata
Receive	0x02	Seq#	length of data	Data	Σdata
	0x03	Seq#	length of data	Data	Σdata

Example :

Send a command (read resolution)

```
commande[0]=0x1b;
commande[1]=0x43;
commande[2]=0x04;//Length LSB
commande[3]=0x00;//Length MSB
commande[4]=0x01;//Get Int32
commande[5]=0x01;//Res offset
commande[6]=0x00;//0
commande[7]=0x00;
commande[8]=0x02;//Checksum LSB
commande[9]=0x00;//Checksum MSB
SrmReceiveFlush(SerialId,0);
SrmSend (SerialId, &commande,10, &err);
```

Receive data (read resolution)

```
SrmReceive (SerialId, &commande,1, timeout, &err);
if ((err==0)&&(commande[0]==0x03))//DATA in one seq
{
  SrmReceive (SerialId, &commande,1, timeout, &err);
  if ((err==0)&&(commande[0]==0x00))//Seq #0
  {
    SrmReceive (SerialId, &commande,2, timeout, &err);
    if ((err==0))//Data length
    {
      UInt16 size=commande[0]+256*commande[1];
      SrmReceive (SerialId, &commande,size, timeout, &err);
      if (!err)
      {
        Char tmp[10];
        StrPrintF(tmp,"%x%x", 256*commande[3]+commande[2],
                              256*commande[1]+commande[0]);
       FrmCustomAlert(AlertAlert,"Resolution : ",tmp, " ");
       commande[0]=0x06;
       SrmSend (SerialId, &commande,1, &err);
      }
    }
  }
}
else FrmCustomAlert(AlertAlert,"error"," ", " ");
```