SXH Protocol For SC20MPF Camera

Version 4.0
Date: 03.2022

Revision History

Vision	Date	Description
V1.0	September, 2016	Initial release
V2.0	December, 2016	Added LED control, OSD, Cmd 0x30
V3.0	April, 2018	Added all 16:9 ratio image sizes Keep only one command for snapshotting images Keep only one command for set baud rate and address ID Deleted auto-focusing, motion detection Added CRC computation reference Notes: all new camera released from April 2018 are compatible with older version protocol V2.0, no coding change needed for upgraded version cameras.
V4.0	March,2022	Disabled manual LED control

Command Structure (in HEX format):

D0	D1	D2	D3	D4	D5	D6~Dn	C1	C2
Frame header 1	Frame header 2	add	Command	Data length Low bit	Data length High bit	Data	CRC check Low bit	CRC check High bit
90	EB	ID	Cmd	LenL	LenH	D……	CrcL	CrcH

General guidance:

1). Frame header is 2 bytes, and is always 90 EB

2). ID is the address of the camera, ranges from 0x01 to 0xfe, 0x00 and 0xff are reserved ID (broadcasting ID), if the camera receives the command which is not for same ID of the camera, the camera will discard this command and will not respond.

3). Cmd is the command issued to the camera.

4). Data length LenH and LenL are the length of D6~Dn, represents the total length of the data (not the length of the frame), and the length can be 0~65535.

5). CrcH, CrcL are 16 bit CRC check, CRC check computed from address "Add" to Dn, checking length equals data length + 4; when sending testing command, if CRC check is not computed, 0xc1, 0xc2 can be used instead.

6). Frame=2 (frame header)+ 1 (ID)+ 1 (command)+ 2 (data length)+ n (data)+ 2 (Check Value)

=8+ n (data length)

=8+D4+D5*256

7). Minimum length is 8 bytes; D4 and D5 can be used to calculate the total length.

1. Testing (Cmd=0x01)

Data bit	D0	D1	D2	D3	D4	D5	D6~Dn	C1	C2
	Frame	Frame	add	Command	Data	Data	Data	CRC check	CRC check

	header 1	header 2			length Low bit	length High bit		Low bit	High bit
Send	90	EB	01	01	02	00	55 aa	C1	C2
Return	90	EB	01	01	03	00	00 aa 55	F6	EB

Send (frame length =10): 90 eb 01 01 02 00 55 aa c1 c2
Return (frame length =11): 90 EB 01 01 01 03 00 aa 55 F6 EB

2. Snapshot (Cmd=0x40)

The host can send this command to capture an image, the image gets stored on the camera, waiting for transferring.

Data bit	D0	D1	D2	D3	D4	D5	D6~Dn	C1	C2
	Frame header 1	Frame header 2	add	Command	Data length Low bit	Data length High bit	Data	CRC check Low bit	CRC check High bit
Send	90	EB	01	40	04	00	00 02 05 01	C1	C2

Send: 90 eb 01 40 04 00 00 02 05 01 c1 c2

90 EB	01	40	04 00	00 02	05	01	C1 C2
Frame header	ID	Command	Data length	Sub-packet size	Resolution	Compression ratio	CRC

Return: 90 EB 01 40 0b 00 00 83 16 00 00 00 00 00 02 05 01 C1 C2

90 EB	01	40	0b 00	00	83 16 00 00	00 00	00 02	05	01	C1 C2
Frame header	ID	Command	Packet length	status	Image size	reserved	reserved	Resol -ution	Compres sion ratio	CRC

Example: Send: 90 EB 01 40 04 00 00 02 05 01 C1 C2
Return: 90 EB 01 40 0B 00 00 76 94 00 00 4B 00 00 02 05 01 27 B5

Compression ratio ranges from 1 to 5, compression ratio value greater, the image gets more compressed, which will sacrifice the image quality.

Parameter	Resolution	Ratio	Notes
5	640*480	4:3	
6	1280*960	4:3	
7	800*600	4:3	
8	1024*768	4:3	
9	1280*800	4:3	
10	1600*1024	4:3	
11	1600*1200	4:3	
15	1280*720	16:9	
16	1920*1080	16:9	Best resolution for 2MP camera
17	1280*1024	5:4	
30	480*270	16:9	Newly added
31	640*360	16:9	Newly added
32	800*450	16:9	Newly added
33	960*540	16:9	Newly added
34	1024*576	16:9	Newly added
35	1280*720	16:9	Newly added

36	1366*768	16:9	Newly added
37	1440*810	16:9	Newly added
38	1600*900	16:9	Newly added

3. Get image by sub-packet (Cmd=0x48)

The host can send this command to camera to get certain length of the image data from start address, sub-packet by sub-packet, and then it combines all data into one JPEG image.

Data bit	D0	D1	D2	D3	D4	D5	D6~D9	D10~D11	C1	C2
	Frame header 1	Frame header 2	add	Cmd	Data length Low bit	Data length High bit	Start address	Length	CRC check Low bit	CRC check High bit
Send	90	EB	01	48	08	00	00 00 00 00	00 03	C1	C2
Return	90	EB	01	49	XX	YY	D0 D1…Dn 83 16 00 00 ……		C1	C2

// Get the first sub-packet, packet size 768 bytes (low bit first)
Send: 90 EB 01 48 06 00 00 00 00 00 00 03 C1 C2 //request 768 bytes image data from the start address 00 (Max 0xffff, 65k)
Return: 90 EB 01 49 00 03 d0 d1…dn c1 c2 // returned 0x0300 (768 bytes) image data back

// Get the second sub-packet, packet size 768 bytes
Send: 90 EB 01 48 06 00 00 03 00 00 00 03 C1 C2 //request 768 bytes image data from the start address 768, (Max 0xffff, 65k)
Return: 90 EB 01 49 00 03 d0 d1…dn c1 c2 // returned 0x0300 (768 bytes) image data back

……

Notes: If it gets more image data than the total length from capturing, it will be processed as error, there will be no valid image returned.

4. Snapshot procedures guide

Step 1: **send snapshot command (Cmd=0x40)**, the camera captures a still image (at night time, the camera will turn the LED's on first before capturing), if the return includes the length of the image, it means the capturing is complete; the image is stored on the camera.

Step 2: **send get image by sub-packet command (Cmd=0x48)**, start from 0, each packet can be 1k bytes (the range can be from 512 bytes to 2K bytes), till it gets all the data for the image.

Step 3: Combine all image data, combine all image data in order, then it gets a JPEG image.

5. Set baud rate and address ID (Cmd=0x44)

Data bit	D0	D1	D2	D3	D4	D5	D6~Dn	C1	C2
	Frame header 1	Frame header 2	add	Command	Data length Low bit	Data length High bit	Data	CRC check Low bit	CRC check High bit
Send	90	EB	01	44	04	00	02 01 0a 0a	C1	C2
Return	90	EB	01	45	04	00	02 01 0a 0a	C1	C2

Send	90 EB	01	44	04 00	02	01	Add add	C1 C2

	Frame header	ID	Command	Data length	Baud rate	Save or not?	New ID	CRC

Notes: if the baud rate parameter is set as 00, the baud rate doesn't get modified, if the new ID is set as 00 or FF, the camera ID doesn't get modified. Address 00 and FF are broadcasting address, when they are being used in command, the command gets sent to every camera in the same bus.

Return	90 EB	01	45	04 00	02	01	Add add	C1 C2
	Frame header	ID	Command	Data length	Baud rate	Saved	New ID	CRC

For different baud rate parameter, please use the table below as a guide.

Baud rate (bps)	**Parameter**
9600	01
19200	02
28800	03
38400	04
57600	05
115200	06

6. OSD Control (Cmd=0x52)

The host can display characters at X and Y axis on images by issuing this command.
Send: 90 EB 01 52 00 00 XL XH YL YH 0B D1 D2 …… C1 C2

Data bit	D0	D1	D2	D3	D4	D5	D6~Dn	C1	C2
	Frame header 1	Frame header 2	add	Command	Data length Low bit	Data length High bit	Data	CRC check Low bit	CRC check High bit
Send	90	EB	01	52	00	00	XL XH YL YH 0B D1 D2……..	C1	C2

Send	90 ED	01	52	00 00	XL XH	YL YH	0B	D1 D2…	C1 C2
	Frame header	ID	Command	Data Length	X axis	Y axis	Font size	Characters	CRC

Note: This command is to display characters at X, Y axis, the characters use ASCII table. If the font size is set as 0, then the font size is default size. Characters length=data length - 5.

For instance, to add "1234e" on coordinate 06, 08, font size is 12, then:
Send: 90 EB 01 52 0A 00 06 00 08 00 0C 31 32 33 34 65 C1 C2
Return: 90 EB 01 52 00 00 C1 C2

Common ASCII Table:

Character	ASCII Value			Character	ASCII Value		
	Decimal	Binary	HEX		Decimal	Binary	HEX
NUL	0	0000000	00	M	77	1001101	4D
Line Feed	10	0001010	A	N	78	1001110	4E
Space	32	0100000	20	O	79	1001111	4F
!	33	0100001	21	P	80	1010000	50

”	34	0100010	22	Q	81	1010001	51
#	35	0100011	23	R	82	1010010	52
$	36	0100100	24	S	83	1010011	53
%	37	0100101	25	T	84	1010100	54
&	38	0100110	26	U	85	1010101	55
‘	39	0100111	27	V	86	1010110	56
(	40	0101000	28	W	87	1010111	57
)	41	0101001	29	X	88	1011000	58
*	42	0101010	2A	Y	89	1011001	59
+	43	0101011	2B	Z	90	1011010	5A
,	44	0101100	2C	[	91	1011011	5B
-()	45	0101101	2D	\	92	1011100	5C
.	46	0101110	2E	]	93	1011101	5D
/	47	0101111	2F	^	94	1011110	5E
0	48	0110000	30	-	95	1011111	5F
1	49	0110001	31	a	97	1100001	61
2	50	0110010	32	b	98	1100010	62
3	51	0110011	33	c	99	1100011	63
4	52	0110100	34	d	100	1100100	64
5	53	0110101	35	e	101	1100101	65
6	54	0110110	36	f	102	1100110	66
7	55	0110111	37	g	103	1100111	67
8	56	0111000	38	h	104	1101000	68
9	57	0111001	39	i	105	1101001	69
:	58	0111010	3A	j	106	1101010	6A
;	59	0111011	3B	k	107	1101011	6B
<	60	0111100	3C	l	108	1101100	6C
=	61	0111101	3D	m	109	1101101	6D
>	62	0111110	3E	n	110	1101110	6E
?	63	0111111	3F	o	111	1101111	6F
@	64	1000000	40	p	112	1110000	70
A	65	1000001	41	q	113	1110001	71
B	66	1000010	42	r	114	1110010	72
C	67	1000011	43	s	115	1110011	73
D	68	1000100	44	t	116	1110100	74
E	69	1000101	45	u	117	1110101	75
F	70	1000110	46	v	118	1110110	76
G	71	1000111	47	w	119	1110111	77
H	72	1001000	48	x	120	1111000	78
I	73	1001001	49	y	121	1111001	79
J	74	1001010	4A	z	122	1111010	7A
K	75	1001011	4B	{	123	1111011	7B
L	76	1001100	4C	}	125	1111101	7D
Character	ASCII Value			Character	ASCII Value		
	Decimal	Binary	HEX		Decimal	Binary	HEX

Other supplemental commands:

CRC check computation

```
//Crc16 computing fuction
const unsigned short  crc_ta[256]={ /* CRC Residue Table */

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, 0xcbdc, 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, 0xa1eb, 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

};

//CRC check , applies for checking continuous data
unsigned short  Crc16(unsigned char *ptr, unsigned short len)
```

```
{
    unsigned short crc;
    unsigned char  da;

    crc=0;
    while(len--!=0)
     {
      da=(unsigned char) (crc/256);
      crc<<=8;
      crc^=crc_ta[da^*ptr];
      ptr++;
    }
    return(crc);
}


//applies for checking non-continuous data
unsigned short  Crc16_New(unsigned short old_crc,unsigned char *ptr, unsigned
short len)
{
    unsigned short crc;
    unsigned char  da;

    crc=old_crc;
    while(len--!=0)
     {
      da=(unsigned char) (crc/256);
      crc<<=8;
      crc^=crc_ta[da^*ptr];
      ptr++;
    }
    return(crc);
}
```