NAVILOCK®

NL-120GR Read LOG Data Produce (Create *.tk1)

Version	Release note	Release date
1.0	1.First version release	2007/08/03

Suit device	Device version
NL-120GR	● Hardware Version: 1.100 ● Software Version: 1.305~ ● Log Version: 1.000 ● TimeMachineX: V2.50~

NAVILOCK®

Read LOG produce:

1. Get necessary parameter from NL-120GR such as Flash ID, log version, hardware version, LOG start area, Device Name, Device Information, LOG end area, LOG start address, LOG end address.

2. Calculate total log capacity (byte); total log point = total log capacity/16.

3. Create a empty binary file, the extend file name is “tk1”. Ex: **tmp.tk1**

4. Fill up the tk1 header structure information (refer NL-120GR_technical_document_3_TK1_Header_End) and write into **tmp.tk1.** Note that: The header structure information item “Numbers of track in the tk1” is 0 now and “First track information address for Seek” = 1024+ total log capacity.

5. Start to read log. First **Current_LOG_Point** = LOG start address. If total log capacity smaller than 4096, means that log data not full a section (4096 Bytes) of memory. => **read_buffer_CNT** = total log capacity; else **read_buffer_CNT** = 4096.

6. Send read log command (**Current_LOG_Point**) to NL-120GR.

7. Read log data from comport that connect with NL-120GR (read **read_buffer_CNT** bytes one time); if not get any data more than 5 sec, please check whether login or not. If not please login first. If already login please exit read log produce or re-send read log command (**Current_LOG_Point**) to NL-120GR and repeat step 7.

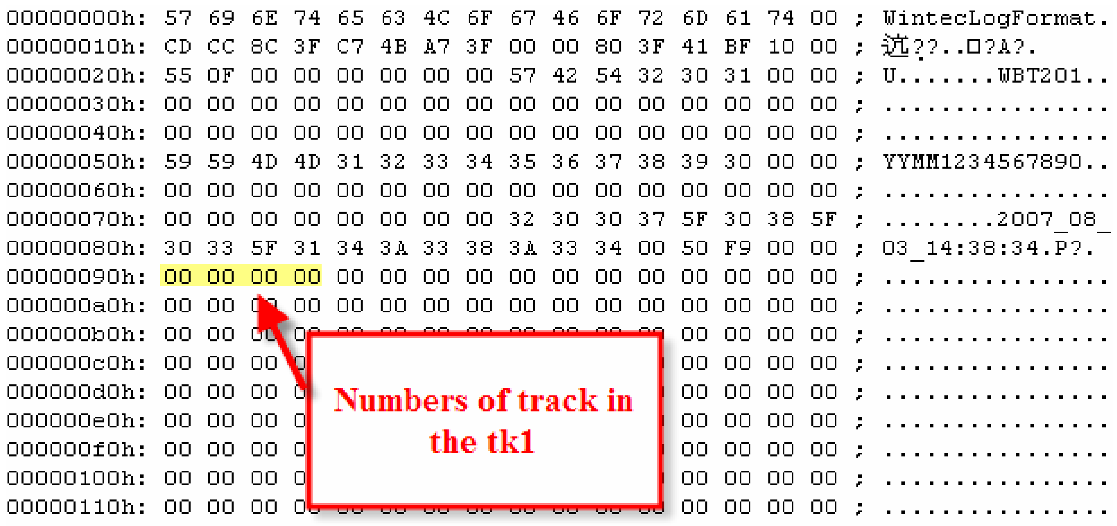
8. If step 7 is finish, continue read check sum (1 unsigned char: **XX**) of this section from comport export data (**@AL,CS,XX, Current_LOG_Point**), then check the value and the check sum that calculate by receiver data (xor each byte of receiver data) whether the same or not. If the same go to step 9; If not the same, repeat step 6(ie. re-send read log command (**Current_LOG_Point**)) and repeat step 7, 8.

9. Finish step 8; Write all receiver data to the **tmp.tk1** file. Then let new **Current_LOG_Point** = last **Current_LOG_Point** +4096; check If **(** new **Current_LOG_Point > LOG end area)**, new **Current_LOG_Point** = **LOG**

start area. And check If **(** new **Current_LOG_Point < LOG end address) & ((**new **Current_LOG_Point +4096) >= LOG end address)** means that is the last section, **read_buffer_CNT** = **(LOG end address -** new **Current_LOG_Point)**; otherwise **read_buffer_CNT** = 4096; send read log command (new **Current_LOG_Point**).

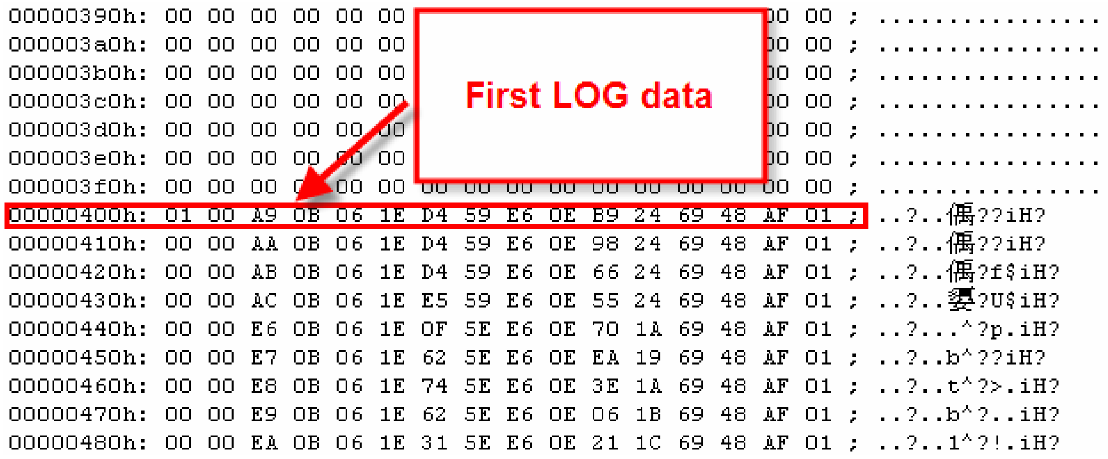
10. Repeat step 6~9, until read down all log data.

Tmp.tk1 header structure data.



```
00000000h: 57 69 6E 74 65 63 4C 6F 67 46 6F 72 6D 61 74 00 ; WintecLogFormat.
00000010h: CD CC 8C 3F C7 4B A7 3F 00 00 80 3F 41 BF 10 00 ; 迒??..□?A?.
00000020h: 55 0F 00 00 00 00 00 00 57 42 54 32 30 31 00 00 ; U.......WBT201..
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 59 59 4D 4D 31 32 33 34 35 36 37 38 39 30 00 00 ; YYMM1234567890..
00000060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000070h: 00 00 00 00 00 00 00 00 32 30 30 37 5F 30 38 5F ; ........2007_08_
00000080h: 30 33 5F 31 34 3A 33 38 3A 33 34 00 50 F9 00 00 ; 03_14:38:34.P?.
00000090h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000a0h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000b0h: 00 00 00                         00 00 00 00 ; ................
000000c0h: 00 00 00                         00 00 00 00 ; ................
000000d0h: 00 00 00                         00 00 00 00 ; ................
000000e0h: 00 00 00                         00 00 00 00 ; ................
000000f0h: 00 00 00                         00 00 00 00 ; ................
00000100h: 00 00 00                         00 00 00 00 ; ................
00000110h: 00 00 00                         00 00 00 00 ; ................
```

Tmp.tk1 LOG data.



```
00000390h: 00 00 00 00 00 00                      00 00 ; ................
000003a0h: 00 00 00 00 00 00                      00 00 ; ................
000003b0h: 00 00 00 00 00 00                      00 00 ; ................
000003c0h: 00 00 00 00 00 00                      00 00 ; ................
000003d0h: 00 00 00 00 00 00                      00 00 ; ................
000003e0h: 00 00 00 00 00 00                      00 00 ; ................
000003f0h: 00 00 00    00 00                      00 00 ; ................
00000400h: 01 00 A9 0B 06 1E D4 59 E6 0E B9 24 69 48 AF 01 ; ..?..偶??iH?
00000410h: 00 00 AA 0B 06 1E D4 59 E6 0E 98 24 69 48 AF 01 ; ..?..偶??iH?
00000420h: 00 00 AB 0B 06 1E D4 59 E6 0E 66 24 69 48 AF 01 ; ..?..偶?f$iH?
00000430h: 00 00 AC 0B 06 1E E5 59 E6 0E 55 24 69 48 AF 01 ; ..?..褭?U$iH?
00000440h: 00 00 E6 0B 06 1E 0F 5E E6 0E 70 1A 69 48 AF 01 ; ..?...^?p.iH?
00000450h: 00 00 E7 0B 06 1E 62 5E E6 0E EA 19 69 48 AF 01 ; ..?..b^??iH?
00000460h: 00 00 E8 0B 06 1E 74 5E E6 0E 3E 1A 69 48 AF 01 ; ..?..t^?>.iH?
00000470h: 00 00 E9 0B 06 1E 62 5E E6 0E 06 1B 69 48 AF 01 ; ..?..b^?..iH?
00000480h: 00 00 EA 0B 06 1E 31 5E E6 0E 21 1C 69 48 AF 01 ; ..?..1^?!.iH?
```

11. Verification the **tmp.tk1.** Verification produces as below:

   11.1 Read (Copy) tk1 header structure from **tmp.tk1** to a new tk1 header structure**.**

   11.2 Read each LOG data from **tmp.tk1** and check its state flag to calculate **n**umbers of track in the tk1, and then fill up into new tk1 header structure item "Numbers of track in the tk1".

   11.3 Create a empty binary file, the extend file name is "tk1". Ex: **Final.tk1**

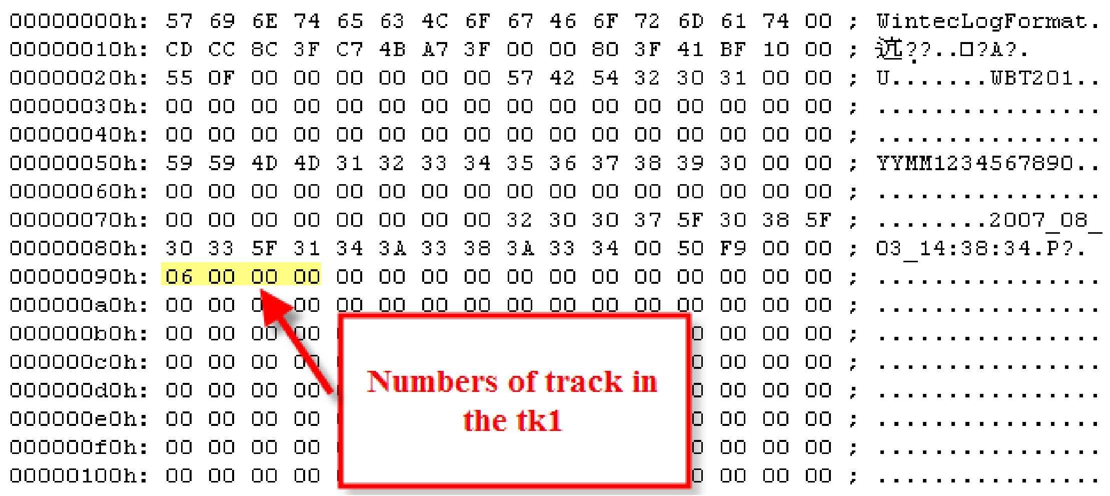
11.4 Write new tk1 header structure into **Final.tk1.**

11.5 Read each LOG data from **tmp.tk1** and check its data correct or not ,(ie month>13, date>31, hour>23, minute>60, second>60, absolute value of latitude >90, absolute value of longitude >180) then fill up into **Final.tk1** and get each track information such as track ID number, This track start Seek from the file start, Numbers of Point in this track, Total spend time in this track (seconds), Total distance in this track (Km) to fill up tk1 end structure (refer NL-120GR_technical_document_3_TK1_Header_End).

11.6 Until this track is finish (ie. The states flag of next log data is start flag), according tk1 end structure item “This track start Seek from the file start” get its address, write this tk1 end structure into **Final.tk1.**

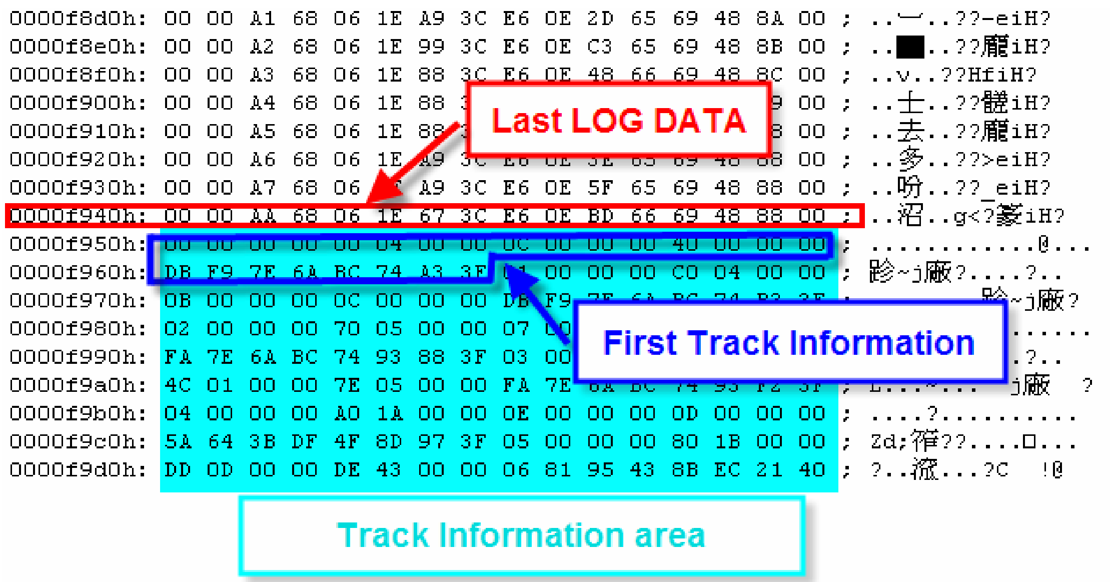
11.7 Repeat step 11.5~11.6 until finish all log data.

Final.tk1 header structure data.



```
00000000h: 57 69 6E 74 65 63 4C 6F 67 46 6F 72 6D 61 74 00 ; WintecLogFormat.
00000010h: CD CC 8C 3F C7 4B A7 3F 00 00 80 3F 41 BF 10 00 ; 烫??..□?A?.
00000020h: 55 0F 00 00 00 00 00 00 57 42 54 32 30 31 00 00 ; U.......WBT201..
00000030h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000040h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000050h: 59 59 4D 4D 31 32 33 34 35 36 37 38 39 30 00 00 ; YYMM1234567890..
00000060h: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
00000070h: 00 00 00 00 00 00 00 00 32 30 30 37 5F 30 38 5F ; ........2007_08_
00000080h: 30 33 5F 31 34 3A 33 38 3A 33 34 00 50 F9 00 00 ; 03_14:38:34.P?.
00000090h: 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000a0h: 00 00    00 00 00 00 00 00 00 00 00 00 00 00 00 ; ................
000000b0h: 00 00 00                               00 00 00 ; ................
000000c0h: 00 00 00                               00 00 00 ; ................
000000d0h: 00 00 00 00                            00 00 00 ; ................
000000e0h: 00 00 00 00                            00 00 00 ; ................
000000f0h: 00 00 00 00                            00 00 00 ; ................
00000100h: 00 00 00 00                            00 00 00 ; ................
```

Final.tk1 end structure data.

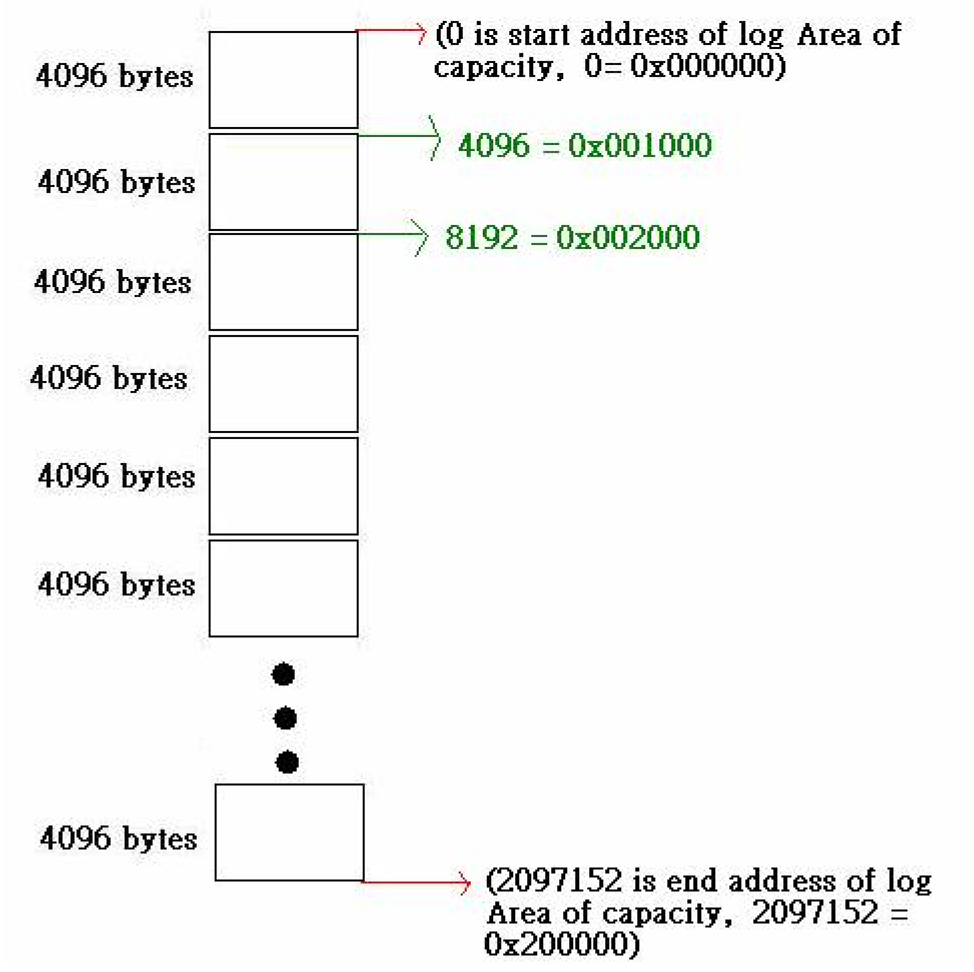


```
0000f8d0h: 00 00 A1 68 06 1E A9 3C E6 0E 2D 65 69 48 8A 00 ; ..ㄧ..??-eiH?
0000f8e0h: 00 00 A2 68 06 1E 99 3C E6 0E C3 65 69 48 8B 00 ; ..█..??龐iH?
0000f8f0h: 00 00 A3 68 06 1E 88 3C E6 0E 48 66 69 48 8C 00 ; ..v..??HfiH?
0000f900h: 00 00 A4 68 06 1E 88                           00 ; ..士..??髓iH?
0000f910h: 00 00 A5 68 06 1E 88                           00 ; ..去..??龐iH?
0000f920h: 00 00 A6 68 06 1E A9 3C E6 0E 3E 65 69 48 88 00 ; ..多..??>eiH?
0000f930h: 00 00 A7 68 06    A9 3C E6 0E 5F 65 69 48 88 00 ; ..吩..??_eiH?
0000f940h: 00 00 AA 68 06 1E 67 3C E6 0E BD 66 69 48 88 00 ; ..沼..g<?篆iH?
0000f950h: 00 00 00 00 00 04 00 00 0C 00 00 00 40 00 00 00 ; ............@...
0000f960h: DB F9 7E 6A BC 74 A3 3F 01 00 00 00 C0 04 00 00 ; 聆~j廠?....?..
0000f970h: 0B 00 00 00 0C 00 00 00 DB F9                      聆~j廠?
0000f980h: 02 00 00 00 70 05 00 00 07 00                      ......
0000f990h: FA 7E 6A BC 74 93 88 3F 03 00                      .?..
0000f9a0h: 4C 01 00 00 7E 05 00 00 FA 7E                      j廠  ?
0000f9b0h: 04 00 00 00 A0 1A 00 00 0E 00 00 00 0D 00 00 00 ; ....?.........
0000f9c0h: 5A 64 3B DF 4F 8D 97 3F 05 00 00 00 80 1B 00 00 ; Zd;徉??....□...
0000f9d0h: DD 0D 00 00 DE 43 00 00 06 81 95 43 8B EC 21 40 ; ?..瀮...?C  !@
```

Read LOG produce example:

TimeMachineX send command to NL-120GR.
TimeMachineX received data from NL-120GR.
Ex:
@AL
@AL
@AL,LoginOK

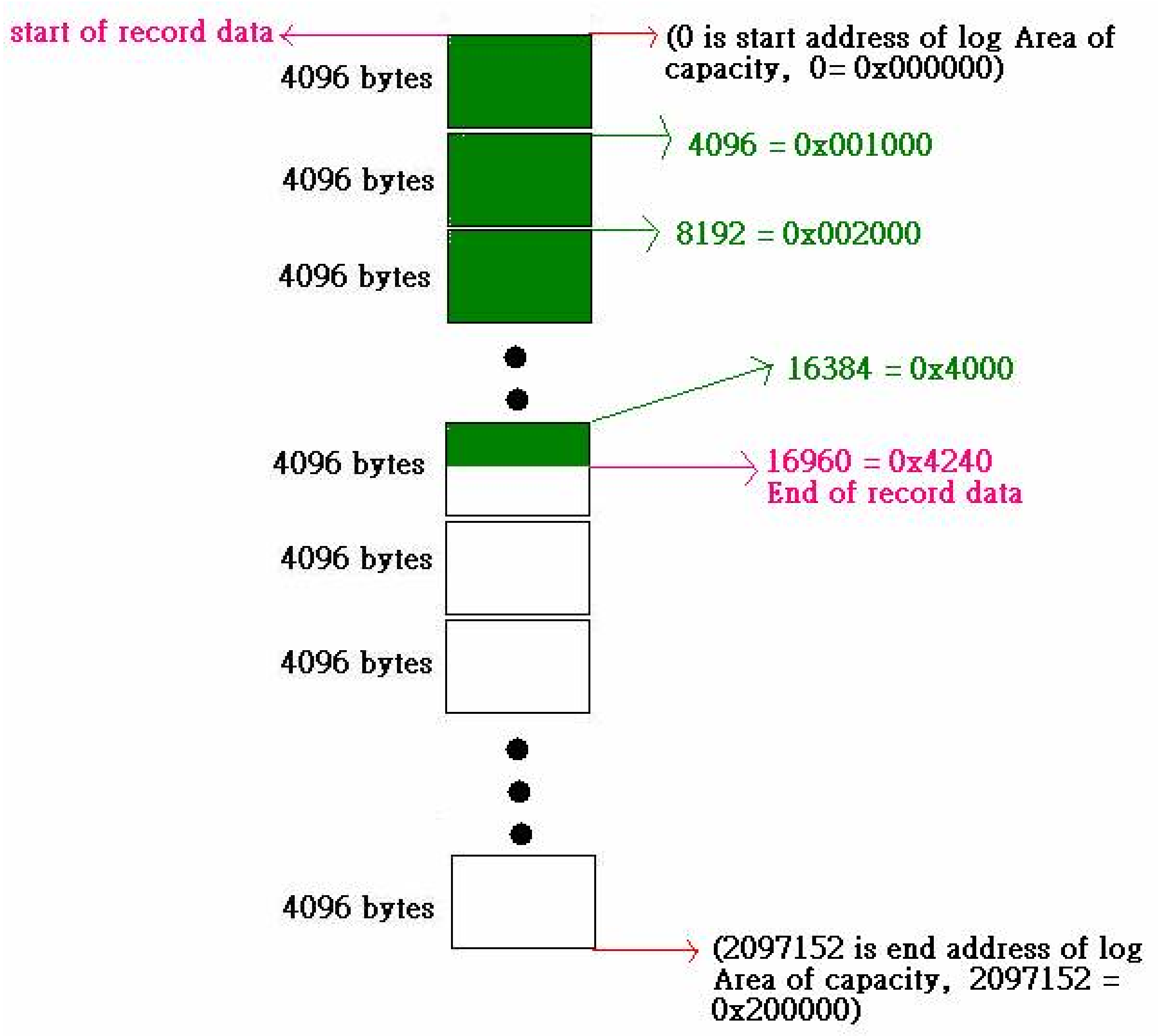
//***
1. Get **LOG start area, LOG end area, LOG start address, LOG end address.**
//***
@AL,5,9
@AL,5,9,0 (0 is start address of log Area of capacity, 0= 0x000000)
=> **LOG start area = 0x000000**
@AL,5,10
@AL,5,10,2097152 (2097152 is end address of log Area of capacity, 2097152 = 0x200000)
=> **LOG end area = 0x200000**



4096 bytes
4096 bytes
4096 bytes
4096 bytes
4096 bytes
4096 bytes
4096 bytes
(0 is start address of log Area of capacity, 0= 0x000000)
4096 = 0x001000
8192 = 0x002000
(2097152 is end address of log Area of capacity, 2097152 = 0x200000)

@AL,5,1
@AL,5,1,0   (0 is start address of record data,   0= 0x000000)
=>LOG start address=0x000000

@AL,5,2
@AL,5,2,16960   (16960 is end address of record data,   16960= 0x004240)
=>LOG end address=0x004240



NAVILOCK®

//***

2. Calculate total log capacity (byte); total log point = total log capacity/16.
3. First **Current_LOG_Point** = LOG start address. If total log capacity smaller than 4096, means that log data not full a section (4096 B) of memory. => **read_buffer_CNT** = total log capacity; else **read_buffer_CNT** = 4096. Create a file to store read log data.

//***

Now you could calculate how many log data was record.
=>Total log capacity: **16960 – 0 = 16960 (Record Data Length <unit is byte>)**
=>Total log point **16960 / 16 = 1060 (number of log data, 16 <byte> is the length of each log data)**
=> **read_buffer_CNT =4096** (16960 > 4096)
=> First **Current_LOG_Point** =0;

//***

4. Send read log command (**Current_LOG_Point**) to NL-120GR.
5. Read log data from comport that connect with NL-120GR (read **read_buffer_CNT** bytes one time); if not get any data more than 5 sec, please check whether login or not. If not please login first. If already login please exit read log produce or re-send read log command (**Current_LOG_Point**) to NL-120GR and repeat step 5.

//***

Now you can send command to read log data.
@AL,5,3,0

Below data is from WBT201 start record address (0x000000)
Total data length is 4096 bytes in this sector (0x000000 ~ 0x000FFF)

```
01 00 A3 BE 62 1D 79 44 E6 0E 43 40 69 48 47 00
00 00 A4 BE 62 1D 04 44 E6 0E 4E 41 69 48 49 00
00 00 A5 BE 62 1D 7F 43 E6 0E 47 42 69 48 4B 00
…………………………………………………………
…………………………………………………………

                Omission

…………………………………………………………
…………………………………………………………
…………………………………………………………
00 00 00 C4 62 1D 40 48 E6 0E 43 F5 68 48 F3 FF
00 00 01 C4 62 1D 40 48 E6 0E 43 F5 68 48 F3 FF
00 00 02 C4 62 1D 40 48 E6 0E 43 F5 68 48 F3 FF
```

NAVILOCK®

//***

6. If step 5 is finish, continue read check sum (1 unsigned char: **XX**) of this section from comport export data (**@AL,CS,XX, Current_LOG_Point**), then check the value and the check sum that calculate by receiver data (xor all receiver data) whether the same or not. If the same go to step 7; If not the same, repeat step 4(ie. re-send read log command (**Current_LOG_Point**)) and repeat step 5, 6.

//***

@AL,CS,23,0 (23 (Hex) is check sum from WBT201, 23 = 0x23)(0 is NL-120GR repeat this sector start address)
@AL,5,3,0 (repeat last command from TimeMachineX to NL-120GR)

```
Calculate Check Sum:
Unsigned int DataLength=4096; //each full sector data length is 4096
Unsigned char CS = 0x23;   //0x23 is Check sum from NL-120GR as above
Unsigned char ChecksumValue = 0;

for(unsigned int i=0; i < DataLength; i++)
{
        ChecksumValue ^=Data[i];
}

If(ChecksumValue == CS)
{
      // check sum is correct. You should store Data.
     //Send command of read next sector to NL-120GR
    //=>New Current_LOG_Point = 4096; this value desired by step 7.
      @AL,5,3,4096

 }
Else
{
    //Check sum error resend read this sector command to NL-120GR
      @AL,5,3,0
}
```

//***

7. Finish step 6; Write all receiver data to the .tk1 file. Then let new **Current_LOG_Point** = last **Current_LOG_Point** +4096; check If **(** new **Current_LOG_Point > LOG end area)**, new **Current_LOG_Point** = **LOG start area**. And check If **(** new **Current_LOG_Point < LOG end address) & ((**new **Current_LOG_Point +4096) >= LOG end address)** means that is the last section, **read_buffer_CNT** = **(LOG end address -** new **Current_LOG_Point)**; otherwise **read_buffer_CNT** = 4096; send read log command (new **Current_LOG_Point**).
8. Repeat step 5~7, until read down all log data.

//***

After Send command of read next sector to NL-120GR (@AL,5,3,4096)
Below data is from NL-120GR start record address (0x001000)
Total data length is 4096 byte in this sector (0x001000 ~ 0x001FFF)

```
01 00 A3 BE 62 1D 79 44 E6 0E 43 40 69 48 47 00
00 00 A4 BE 62 1D 04 44 E6 0E 4E 41 69 48 49 00
00 00 A5 BE 62 1D 7F 43 E6 0E 47 42 69 48 4B 00
…………………………………………………………
…………………………………………………………
```

Omission

```
…………………………………………………………
…………………………………………………………
…………………………………………………………
00 00 00 C4 62 1D 40 48 E6 0E 43 F5 68 48 F3 FF
00 00 01 C4 62 1D 40 48 E6 0E 43 F5 68 48 F3 FF
00 00 02 C4 62 1D 40 48 E6 0E 43 F5 68 48 F3 FF
```

@AL,CS,48,4096 (48 (Hex) is check sum from NL-120GR, 48 = 0x48)
@AL,5,3,4096 (repeat last command from TimeMachineX to NL-120GR)

Calculate Check Sum:

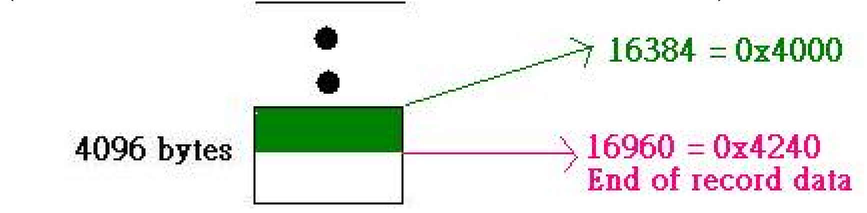
```
Unsigned int DataLength=4096; // each full sector data length is 4096
Unsigned char CS = 0x48;   //0x48 is Check sum from NL-120GR as above
Unsigned char ChecksumValue = 0;

for(unsigned int i=0; i < DataLength ; i++)
{
        ChecksumValue ^=Data[i];
}

If(ChecksumValue == CS)
{
      // check sum is correct. You should store Data.
      //Send command of read next sector to NL-120GR
     //=>New Current_LOG_Point = 8192; this value desired by step 7.
      @AL,5,3,8192

}
Else
{
      //Check sum error resend read this sector command to NL-120GR
      @AL,5,3,4096
}
```

//***

Repeat until last sector. The last sector is not full sector in this example.

//***

(16960 is end address of record data, 16960= 0x004240)



Now read last sector.

Note: This sector data length only 576 bytes (576 bytes = 0x240 bytes , ie only 36 (576/16=36) Log data in this sector)

=> read_buffer_CNT = 576

@AL,5,3,16384

```
00 00 39 06 7E 1D 88 47 E6 0E 63 5C 69 48 79 00
00 00 43 06 7E 1D 88 47 E6 0E 4C 5D 69 48 79 00
00 00 50 06 7E 1D 45 47 E6 0E 4C 5D 69 48 79 00
00 00 58 06 7E 1D 24 47 E6 0E C1 5D 69 48 79 00
00 00 5D 06 7E 1D C0 46 E6 0E 9F 5D 69 48 79 00
00 00 63 06 7E 1D 6D 46 E6 0E 8F 5D 69 48 79 00
00 00 83 06 7E 1D 6D 46 E6 0E D1 5D 69 48 79 00
00 00 88 06 7E 1D 9F 46 E6 0E D8 5C 69 48 7A 00
00 00 A8 06 7E 1D 8E 46 E6 0E B0 5D 69 48 7B 00
00 00 AD 06 7E 1D 8E 46 E6 0E 25 5E 69 48 7C 00
00 00 E6 06 7E 1D 9F 46 E6 0E BB 5E 69 48 7C 00
00 00 F7 06 7E 1D AF 46 E6 0E 40 5F 69 48 7D 00
00 00 03 07 7E 1D 03 47 E6 0E F8 5F 69 48 7D 00
00 00 08 07 7E 1D 45 47 E6 0E 2F 5F 69 48 7E 00
00 00 17 07 7E 1D 1E 48 E6 0E F2 60 69 48 7F 00
00 00 1C 07 7E 1D CB 47 E6 0E DB 61 69 48 7F 00
00 00 21 07 7E 1D AA 47 E6 0E B2 65 69 48 80 00
00 00 26 07 7E 1D 0E 48 E6 0E 21 67 69 48 82 00
00 00 52 07 7E 1D 82 48 E6 0E 31 67 69 48 84 00
00 00 58 07 7E 1D B4 48 E6 0E 74 67 69 48 84 00
00 00 5D 07 7E 1D 8D 49 E6 0E 4D 68 69 48 86 00
00 00 62 07 7E 1D 12 4A E6 0E A0 68 69 48 87 00
00 00 88 07 7E 1D 12 4A E6 0E A0 68 69 48 88 00
00 00 A3 07 7E 1D 01 4A E6 0E A0 68 69 48 89 00
00 00 B0 07 7E 1D D0 49 E6 0E 2C 68 69 48 8A 00
00 00 B8 07 7E 1D 3A 49 E6 0E 31 67 69 48 8A 00
00 00 49 08 7E 1D BA 47 E6 0E 6A 66 69 48 8B 00
00 00 4E 08 7E 1D CB 47 E6 0E 5F 65 69 48 8B 00
00 00 53 08 7E 1D E6 48 E6 0E 49 63 69 48 8B 00
00 00 58 08 7E 1D 71 48 E6 0E 9E 60 69 48 8B 00
```

```
00 00 82 08 7E 1D 09 46 E6 0E 3A 60 69 48 8B 00
01 00 69 0E 7E 1D 77 3C E6 0E F6 62 69 48 84 00
00 00 6A 0E 7E 1D E1 3B E6 0E 7C 63 69 48 81 00
01 00 8A 10 7E 1D 3F 3D E6 0E 6E 68 69 48 7A 00
01 00 2A 12 7E 1D C5 3A E6 0E 54 4B 69 48 33 00
00 00 2B 12 7E 1D 51 3A E6 0E 9D 4A 69 48 37 00
```

Note: This sector data length only 576 bytes, so you should stop receive data to your buffer. And waiting Check sum (@AL,CS..) and repeat last command.

@AL,CS,93,16384 (93 (Hex) is check sum from NL-120GR, 93 = 0x93)
@AL,5,3,16384 (repeat last command from TimeMachineX to NL-120GR)

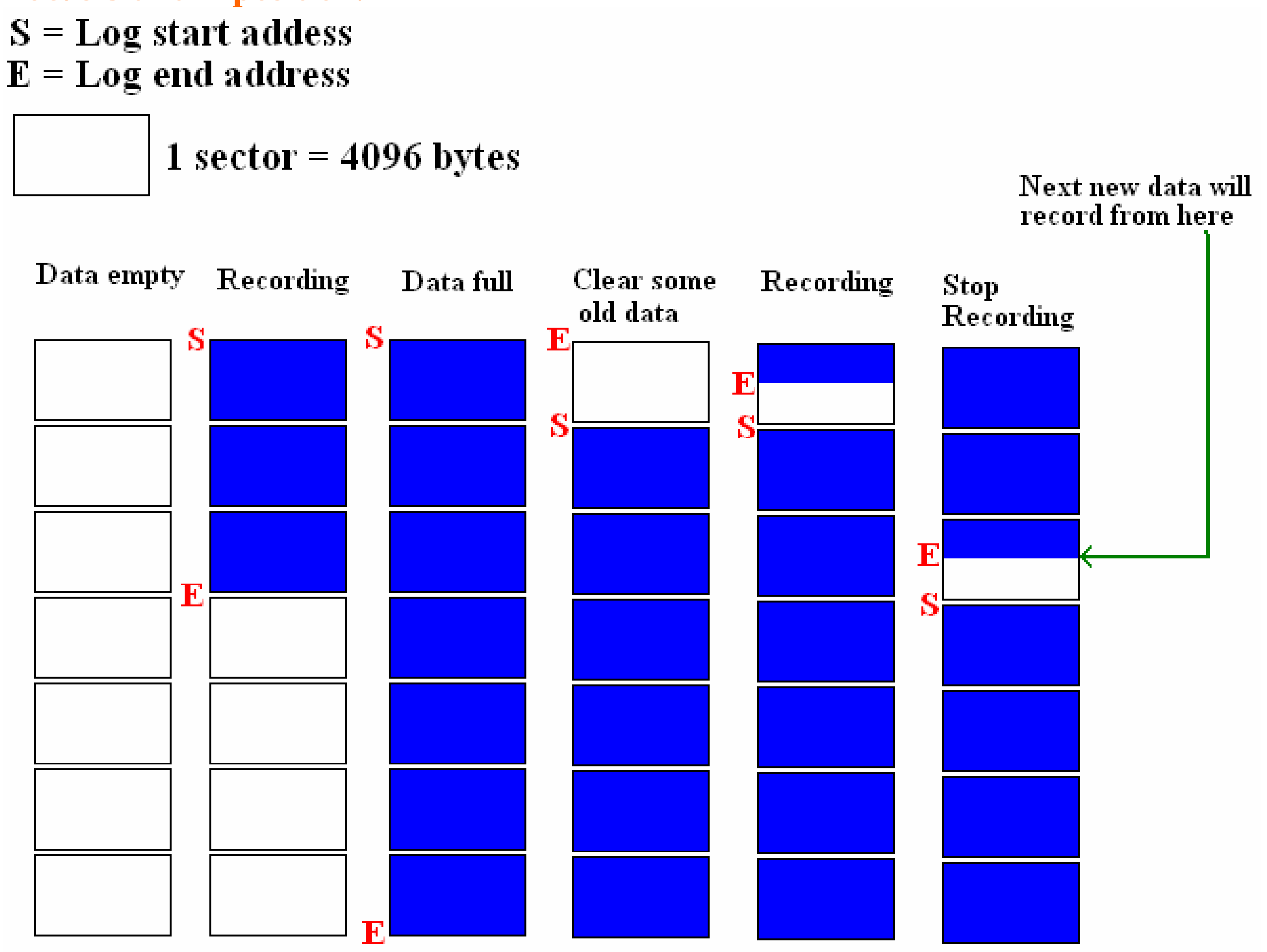
Calculate Check Sum:

```
Unsigned int DataLength=576;// (This sector record data length is 576)
Unsigned char CS = 0x93;   //0x93 is Check sum from NL-120GR as above.
Unsigned char ChecksumValue = 0;

for(unsigned int i=0; i < DataLength ; i++)
{
        ChecksumValue ^=Data[i];
}

If(ChecksumValue == CS)
{
    // check sum is correct. You should store Data.
    //All Data correct received. Finish read log produce.
}
Else
{
      //Check sum error resend read this sector command to NL-120GR
      @AL,5,3,16384
}
```

LOG Data store area distribution:

This is show when record data over log area capacity active and indicate how to read log.

Focus S and E position.


S = Log start addess
E = Log end address
1 sector = 4096 bytes
Next new data will record from here
Data empty
Recording
Data full
Clear some old data
Recording
Stop Recording
S
E
S
E
E
S
E
S
E
S