Make your own software with DasHard2006.dll and our Intelligent Dmx Interface

Overview

Our DasHard2006.dll is a 32 bit Windows DLL (Dynamic Link Librairy). and works on Windows ME, 2000 and XP. It has been tested on Visual C++.

Files

The required files are:

- _DasHard.h
- DasHard2006.dll

Function prototypes

The DasHard2006.dll contains only one function :

int **DasUsbCommand**(int **command**, int **param**, unsigned char ***bloc**);

The first parameter <*command*> defines the thing to do :

command	*explanation*	*param*	*bloc*	*return value*	*Interfaces*
DHC_INIT	Initialisation of the DLL	not used	not used	If the function succeeds (positive values), the return value is the version of the DLL	Siudi5 Siudi6
DHC_EXIT	Closes the dll and free memory prior to application exit	not used	not used		Siudi5 Siudi6
DHC_OPEN	Enables to open the communication with the interface	not used	not used	DHE_OK if the function succeeds.	Siudi5 Siudi6
DHC_CLOSE	Enables to stop the communication with the interface	not used	not used	DHE_OK If the function succeeds. DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the function fails.	Siudi5 Siudi6

DHC_DMXOUT	Enables to send a DMX block to the interface	Specifies the size, in bytes, of the DMX block of memory to send. The normal value is 512	**[out]** Pointer to the DMX block of memory to send	DHE_OK If the function succeeds DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the communication fails.	Siudi5 Siudi6
DHC_DMXOUTOFF	Enables to clean the DMX ouput (force all levels to 0)	not used	not used	DHE_OK If the function succeeds DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the communication fails.	Siudi5 Siudi6
DHC_DMXIN	Enables to read DMX input.	Specifies the size, in bytes, of the DMX block of memory to read. The normal value is 512	**[In]** Pointer to the DMX block of memory to read.	Number of bytes read.	Siudi5
DHC_DMXSCODE	Enables to change the start code	Value of the Start Code	not used	DHE_OK If the function succeeds DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the communication fails.	Siudi5A Siudi5C
DHC_DMX2ENABLE	Enables to change the DMX input in DMX Output	1 for change IN -> OUT 0 for change OUT -> IN	not used	DHE_OK If the function succeeds DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the communication fails.	Siudi5A Siudi5C
DHC_DMX2OUT	Enables to send a DMX block to the second output of the interface	Specifies the size, in bytes, of the DMX block of memory to send. The normal value is 512	**[out]** Pointer to the DMX block of memory to send	DHE_OK If the function succeeds DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the communication fails.	Siudi5A Siudi5C

DHC_PORTREAD	Enables to read the state of the 8 ports and the Next/Previous buttons	not used	not used	DHE_ERROR_NOTOPEN If the interface is not open. If the function succeeds, the return value is from 0 to 1023 (10bits), Bit0=NEXT, Bit1=PREVIOUS, Bit2-9=State of 8ports	Siudi5A Siudi5C
DHC_PORTWRITE	Not yet implemented				
DHC_PORTCONFIG	Not yet implemented				
DHC_WRITEMEMORY	Enables to write the stand alone memory.	Specifies the size, in bytes, of the block of memory to write.	**[out]** Pointer to the block of memory	DHE_OK If the function succeeds. DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the function fails.	Siudi5A Siudi6A
DHC_READMEMORY	Enables to read the stand alone memory.	Specifies the size, in bytes, of the block of memory to read.	**[In]** Pointer to the block of memory	DHE_OK If the function succeeds. DHE_ERROR_NOTOPEN If the interface is not open. DHE_ERROR_COMMAND If the function fails.	Siudi5A Siudi6A
DHC_SIZEMEMORY	Enables to know the size of the stand alone memory.	not used	not used	DHE_ERROR_NOTOPEN If the interface is not open. return the size, in bytes, of the stand alone memory.	Siudi5A Siudi6A
DHC_VERSION	Enables to know the firmware version	not used	not used	return the firmware version	Siudi5 Siudi6
DHC_SERIAL	Enables to know the serial number	not used	not used	return the serial number	Siudi5 Siudi6

Remarks:
- All the constants DHC_OPEN, DHC_CLOSE, DHE_OK .... are defined in the "_DasHard.h" include file.

You can use up to 10 nterfaces simultaneously.
To do this, just add a value in the <command> parameter :
- add 100 (DHC_SIUDI1) if you want to use the interface #2
- add 200 (2 * DHC_SIUDI1) if you want to use the interface #3 ...
Example: DasUsbCommand(DHC_SIUDI1+DHC_OPEN, 0, 0) opens the interface #2

Example of code using our DLL - C++ style

Opening the interface when your application is starting:

```
int interfaceOpen;
int numberOfInterface;
unsigned char dmxBlock[512];

DasUsbCommand(DHC_INIT,0, NULL);
interfaceOpen = DasUsbCommand(DHC_OPEN,0,0);
if (interface_open>0){
    for(int i=0;i<512;i++)
      dmxblock[i] = 0;
  }
}
```

Sending the DMX signal :

```
if (interface_open>0){
  DasUsbCommand(DHC_DMXOUT, 512, dmxblock);
}
```

Note :
- After 5 seconds without communication, the interface go in stand alone mode. This is why we propose to **write the dmx signal all the time** to force a communication.

Closing the interface when your application is stopping:

```
if (interface_open>0)
  DasUsbCommand(DHC_CLOSE,0,0);
DasUsbCommand(DHC_EXIT,0, NULL);
```

Data format of the stand alone memory

```
8bits       set to 2
8bits       set to 5
8bits       first channel                0=1 1=3 ... 255=511
8bits       [c]: number of channels      0=2 1=4 ... 255=512
8bits       set to 0
8bits       set to 0
16bits      [s]: number of scenes
8bits       [p]: number of ports          (to trigger scenes with external ports)
8bits       [n]: number of time trigger   (to trigger scenes with internal clock)
16bits      [t]: size of time trigger bloc data
[p]x 16bits port trigger bloc data: each 16bits contains scene number(0 for nothing)
            ex: port1 (=address1) for scene2 [0]->2;
                port4 (=address8) for scene3 [8]->3.
[t]x 8bits  time trigger bloc data: contains the trigger data, the scene number
            ([t] = [n] x XXbits, XX = [32bits..128bits], [n] = [0..20])
[c]x 8bits  channels settings: bit8 (0 for CUT,1 for FADE), bit7 (1 for DIMMER on)
[s]x 16bits Address/2 of each scene: [0]-> address/2 of scene1..., [1]-> address/2
of scene2


SCENE1
      16bits      <number of steps> = [p]
      8bits       <number of loops, set 0 to loop always>
      8bits       <scene settings, bit0=AUTONEXT, bit1=JUMP, bit2=FADE>
      16bits      <index of next scene if AUTONEXT>
      STEP1 16bits      <fade time step1>
            16bits      <wait time step1>
            [c] x 8bits <DMX levels step1>
      STEP2 16bits      <fade time step2>
            16bits      <wait time step2>
            [c] x 8bits <DMX levels step2>
      STEP3       . . . . . .

SCENE2
      ...


Note :
For 16 bits number, high byte is the first.
JUMP parameter for a scene means that this scene cannot be called with
next and previous buttons.
```

Time trigger bloc data (Only for Siudi5A interfaces) :

20 scenes can be triggered by the internal clock.

There are 3 types of trigger :

- Appointed time
- Repeating time slot
- Unsettled time (not yet implemented)

Each trigger can have different options:

- triggering everyday
- triggering only one day (dd/mm)
- triggering several days (from dd/mm to dd/mm)

Data format of each type of triggering

The first 8 bytes define the type of trigger and the options :

- ED: triggering everyday . Parameters « day 1 » and « day 2 » are not used.
- OD: triggering only the « day 1 » . Parameter « day 2 » is not used.
- FTD: triggering from « day 1 » to « day 2 » .

- SS: Unsettled time (not yet implemented)
- OH: triggering at « hour 1 ». Parameter « hour 2 » is not used.
- FTT: triggering from « hour 1 » to « hour 2 » every « hour 3 » .

HOUR = *hour* * 60 + *minute* (16 bits)
DAY = *month* * 100 + *day* (16 bits)

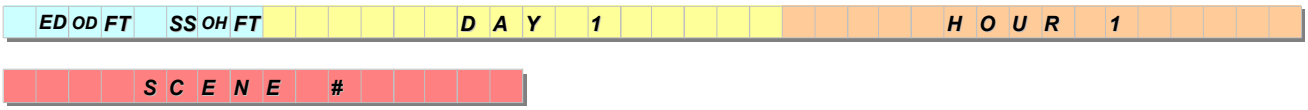
If *month* is set to 0, it means all month.
If *day* is set to 32, it means sunday.
If *day* is set to 33, it means monday.
If *day* is set to 34, it means tuesday

SCENE is 8 bits
For DAY and *HOUR* is coded *high byte is the first*

Case 1 or trigger everyday at a specified time:

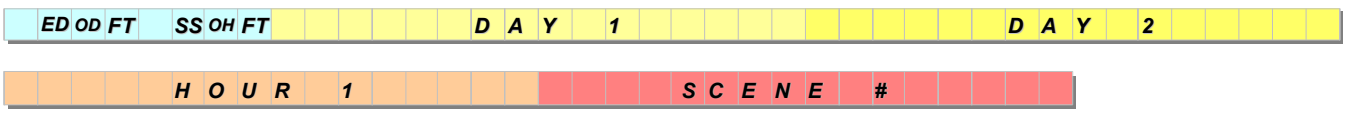
ED = 1, OD = 0, FTD = 0, SS = 0, OH = 1, FTT = 0 (0x42).

Case 2 or trigger the « day 1 » at « hour 1 »:



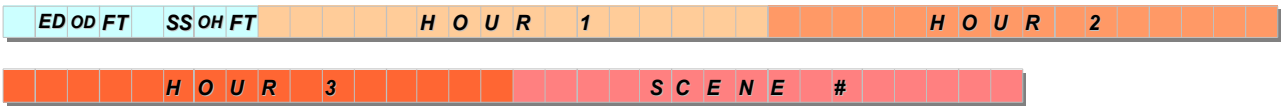
ED = 0, OD = 1, FTD = 0, SS = 0, OH = 1, FTT = 0 (0x22).

Case 3 or triggger from « day 1 » to « day 2 » at « hour 1 »:



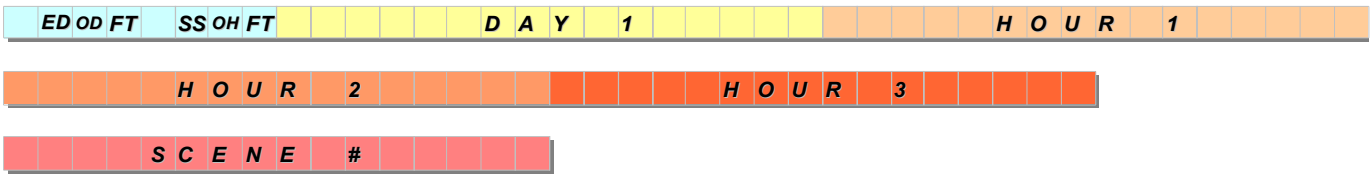
ED = 0, OD = 0, FTD = 1, SS = 0, OH = 1, FTT = 0 (0x12)

Case 4 or trigger everyday from « hour 1 » to « hour 2 » every « hour 3 »:



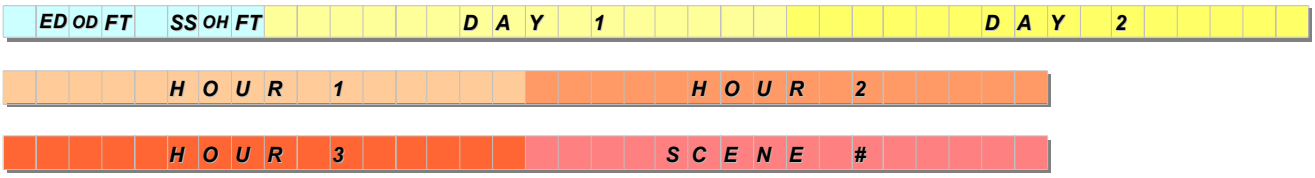
ED = 1, OD = 0, FTD = 0, SS = 0, OH = 0, FTT = 1 (0x41)

Case 5 or trigger the « day 1 », from « hour 1 » to « hour 2 » every « hour 3 »:



ED = 0, OD = 1, FTD = 0, SS = 0, OH = 0, FTT = 1 (0x21)

Case 6 or trigger from « day 1 » to « day 2 », from « hour 1 » to « hour 2 » every « hour 3 »:



ED = 0, OD = 0, FTD = 1, SS = 0, OH = 0, = 1 (0x11)

Case 7:

Not yet implemented

Case 8:

Not yet implemented

Case 9:

Not yet implemented

Comments on version 110:

First official release

Comments on version 130:

Comments on version 131:

Siudi6 management

New functions added and some minor bugs corrected.

Comments on version 136:

Firmware Siudi6A v1.15 and Siudi6C v1.14 supported.

Some minor bugs corrected.

Please report any problems to julien@nicolaudie.com

www.nicolaudie.com