Make your own software with DasNet.dll and our Intelligent DMX Interface

Overview

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

Files

The required files are:

- _DasNet.h
- DasNet.dll

Function prototypes

The DasNet.dll contains only one function :

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

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

command	*explanation*	*param*	*bloc*	*return value*
DNC_INIT	Initialisation of the DLL	not used	not used	If the function succeeds (positive values), the return value is the version of the DLL
DNC_EXIT	Closes the dll and free memory prior to application exit	not used	not used	
DNC_POLL	Enables to look for connected interface on the nerwork	Specifies the number of interface	**[in]** Pointer to the structure which contains the information about the interface (INTERFACEIP*)	DNE_ERROR If the function fails. If the function succeeds, the return value is the number of interface discover.
DNC_OPEN	Enables to start the communication with the interface	not used	not used	Positive value if the function succeeds.
DNC_CLOSE	Enables to stop the communication with the interface	not used	not used	DNE_OK If the function succeeds. DNE_ERROR_NOTOPEN If the interface is not open. DNE_ERROR_COMMAND If the function fails.
DNC_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	DNE_OK If the function succeeds DNE_ERROR_NOTOPEN If the interface is not open. DNE_ERROR_NETWORK If the communication fails.

command	***explanation***	***param***	***bloc***	***return value***
DNC_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	DNE_OK If the function succeeds. DNE_ERROR_NOTOPEN If the interface is not open. DNE_ERROR If the function fails.
DNC_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.
DNC_PLAYSHOW	Enables to start or stop the stand alone mode	not used	not used	DNE_OK If the function succeeds. DNE_ERROR_NOTOPEN If the interface is not open.
DNC_PORTREAD	Enables to read the state of the 8 ports and the Next/Previous buttons	not used	not used	DNE_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
DNC_CONFIG	Enables to open a dialog to configure the parameter	not used	not used	DNE_OK If the function succeeds. DNE_ERROR_NOTOPEN If the interface is not open. DNE_ERROR_COMMAND If the function fails.
DNC_GETNAME	Enables to know the name of the interface	Specifies the size in bytes, of the block. It must be equal to 10	**[in]** Pointer to the block that receives the name	DNE_OK If the function succeeds. DNE_ERROR If the function fails.
DNC_GETVERSION	Enables to know the firmware version	Not used	Not used	Return the firmware version
DNC_GETIPADDRESS	Enables to know the IP address	Specifies the size in bytes, of the block. It must be equal to 4	**[in]** Pointer to the block that receives the Ip address	DNE_OK If the function succeeds. DNE_ERROR If the function fails.
DNC_GETSERIAL	Enables to know the serial number	Not used	Not used	Return the serial number
DNC_GETSIZEMEMORY	Enables to know the size of the stand alone memory	Not used	Not used	Return the size of the stand alone memory
DNC_SYNCHROCLOCK	Synchronise clock of all connected interface	1 -> Open a dialog to set up the time 0 -> Use Windows Time		

Remarks:

- All the constants DNC_OPEN, DNC_CLOSE, DNE_OK .... are defined in the "_DasNet.h" include file.

You can use up to 40 IP interfaces simultaneously.
To do this, just add a value in the <command> parameter :

- add 100 (DNC_SIUDI1) if you want to use the interface #2
- add 200 (2 * DNC_SIUDI1) if you want to use the interface #3 ...

Example: DasNetCommand(DNC_SIUDI1+DNC_OPEN, 0, 0) opens the interface #2

When several interfaces are connected, the interface #1 is the one with the smallest IP address.

Example of code using our DLL - C++ style

Opening the interface when your application is starting:

```
int interfaceOpen;
INTERFACEIP interface[10];
int numberOfInterface;
unsigned char dmxBlock[512];

NetDllCommand(DNC_INIT,0, NULL);
numberOfInterface = NetDllCommand(DNC_POLL,10, &interface);
if(numberOfInterface>0){
  interfaceOpen = NetDllCommand(DNC_OPEN,0,0);
  if (interface_open>0){
    for(int i=0;i<512;i++)
      dmxblock[i] = 0;
  }
}
```

Sending the DMX signal and reading the PORT:

```
int v,ports;
if (interface_open>0){
  ports = NetDllCommand(DNC_PORTREAD,0,0);
  NetDllCommand(DNC_DMXOUT, 512, dmxblock);
}
```

Note :

- After 20 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:

```
int v;
if (interface_open>0)
  v = NetDllCommand(DNC_CLOSE,0,0);
NetDllCommand(DNC_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 each 16bits contains: scene number (16bits) 0 -> Nothing
[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>
      16bits      <index of JUMP scene if JUMP>
      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
      ...
```

<u>Note :</u>

`For 16 bits number, low byte is the first.`

Time trigger bloc data (`[t]x 8bits`):

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 » .

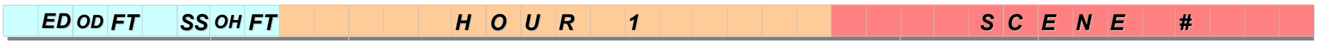
- SS: Unsettled time (not yet implemented)
- OH: triggering at « hour 1 ». Parameter « hour 2 » is not used.
- FTT: triggering from « hour 1 » to « hour 2 » every « hour 3 » .

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

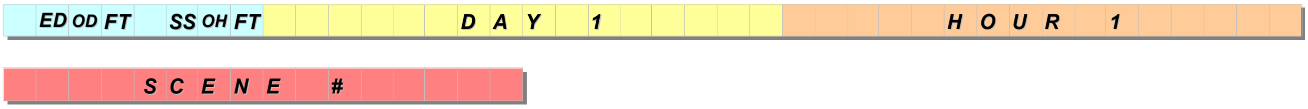
SCENE is 8 bits
DAY and *HOUR* is coded *high byte first – low byte*

Case 1 or trigger everyday at a specified time:



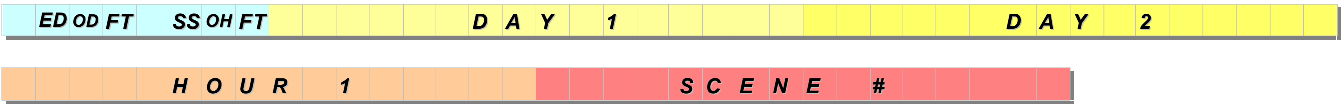
ED = 1, OD = 0, FTD = 0, SS = 0, OH = 1, FTT = 0 (0x42).

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



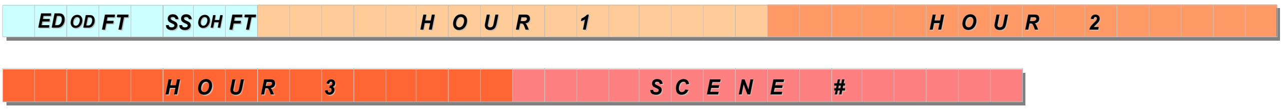
ED = 0, OD = 1, FTD = 0, SS = 0, OH = 1, FTT = 0 (0x22).

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 »:

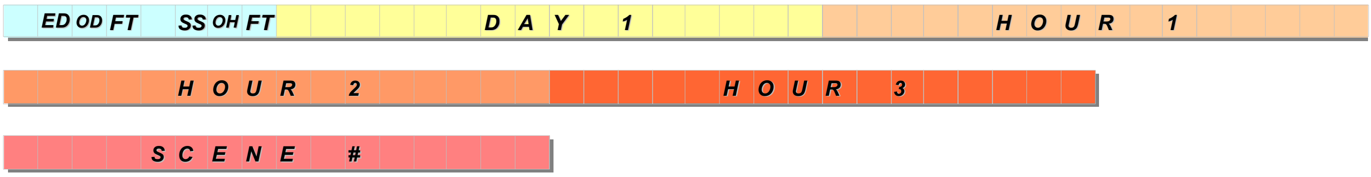


ED OD FT SS OH FT HOUR 1 HOUR 2

HOUR 3 SCENE #

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 »:



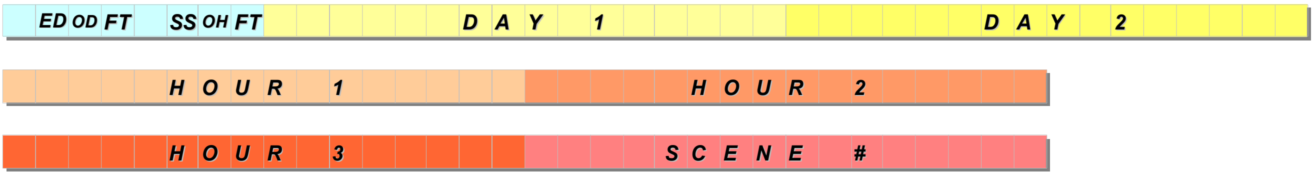
ED OD FT SS OH FT DAY 1 HOUR 1

HOUR 2 HOUR 3

SCENE #

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 OD FT SS OH FT DAY 1 DAY 2

HOUR 1 HOUR 2

HOUR 3 SCENE #

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

Please report any problems to support@soundlight.de

www.pcdmx512.com