

Document	:	Saa7146aDemux
Author	:	M. Majoor
Subject	:	Demux DLL for SAA7146A based budget cards

Revision/Date	Author	Description
20040710	M.Majoor	Initial draft

Saa7146aDemux.dll is a dynamic link library which can be used by application programmers to retrieve satellite data from the SAA7146A based budget cards. These budget cards include the Hauppauge WinTV Nova(-CI), TechnoTrend PC-line budget, Terratec Cinergy 1200 DVB-S.

Since the full source code is included with the dll, programmers should be able to retrieve all relevant information from that. For those which just want to use the dll, without browsing through source code, this document gives the basic information on how to use it.

The 'structure' for most of the functions is based on the LINUX DVB API Version 3. The manual for that can be found at <http://www.linuxtv.org/>.

Basic information:

- All function/procedures in the dll should be accessed using the **stdcall** calling convention.
- When accessing the dll functions by name, be aware that these names are case sensitive!
- Result codes and IOCTL identifiers(*) are those as defined by the Linux DVB API.
- All pointer types require the caller to have allocated the memory for returning the result.

(*) IOCTL identifiers are actually variable by means of settings in the Saa7146a.ini file.

Where type declarations are concerned the following applies:

INTEGER: 32 bit
PCHAR: Pointer to zero terminated string (character based)

IOCTL (index 3, name 'Ioctl')			
function Ioctl(const FileDescriptor: Integer; const Request: Integer; Parameter: Pointer): Integer; stdcall;			
IN	INTEGER	FileDescriptor	NOT USED
	INTEGER	Request	IOCTL identifier The actual link between an IOCTL identifier and the function to call is set in the Saa7146a.ini file (more on this later).
	POINTER	Parameter	Parameter(s) associated with the <i>Request</i> . In most cases a pointer to the parameter, but some functions pass the value instead. Refer to the LINUX DVB API manual for the parameters associated with the different requests if not discussed in this document.
OUT	INTEGER		Result code

The following IOCTL calls have been implemented:

```
DMX_BACKDOOR_START_FEED
DMX_BACKDOOR_STOP_FEED
DMX_REGISTER_CALLBACK
```

In the Saa7146a.ini file the actual IOCTL identifiers the dll responds to are set. This means that an application can use it's own typical identifiers to call specific functions. When for instance an application wants to call the `DMX_BACKDOOR_START_FEED` function, then it should use the identifier as being set in the Saa7146a.ini file. When the .ini file is set as follows:

```
[Ioctl]
DMX_BACKDOOR_START_FEED=10
```

then the application should use '10' as IOCTL identifier (Request).

Thus:

```
Ioctl(FileDescriptor, 10, ParameterPointer)
```

will call the `DMX_BACKDOOR_START_FEED` function.

DMX_BACKDOOR_START_FEED

This command sets the PID for which packets, with that PID, are to be transported. This transport is by means of a callback which is set by the `DMX_REGISTER_CALLBACK` function.

Special case: When using the PID value '65535' then all PIDs (thus all data) will be transported.

This function uses the following parameter structure:

```
PDmxBackDoorFeed = ^TDmxBackDoorFeed;
TDmxBackDoorFeed = packed record
    Pid      : Integer;
    PidType  : Integer;
    PidPestype: Integer;
end;
```

Only the 'Pid' property is used, others are ignored.

DMX_BACKDOOR_STOP_FEED

This command stops the transport of packets with the corresponding PID.

Special case: When using the PID value '65535' then all PIDs will be stopped, thus no data at all will be transported.

This function uses the following parameter structure:

```
PDmxBackDoorFeed = ^TDmxBackDoorFeed;
TDmxBackDoorFeed = packed record
    Pid      : Integer;
    PidType  : Integer;
    PidPestype: Integer;
end;
```

Only the 'Pid' property is used, others are ignored.

DMX_REGISTER_CALLBACK

This command sets the function to call for each packet for which a 'feed' (PID) has been registered through the `DMX_BACKDOOR_START_FEED` function.

A 'nil' value for the parameter disables the callback mechanism.

The parameter for this function is the address to the callback function. This function should be defined as follows:

```
TCallbackProc = function(Buffer: Pointer; DataLength: Integer): Integer; stdcall;
```

or

```
TCallbackProc = function(Buffer: Pointer; DataLength: Dword): Dword; stdcall;
```

Note that the return value from this function is not used. The data size is fixed to 188 bytes in size.

The dll uses the Saa7146a.ini file for some of it's settings. These are the actual IOCTL identifiers the dll uses and what kind of hardware to control. Other settings control the way a log-file is generated (if any).

Here below is a typical Saa7146a.ini file.

```
[Ioctl]
; This is the list of Ioctl identifiers used. These identifiers
; should be identical to the identifiers used by the application
; calling the DLL.
DMX_BACKDOOR_START_FEED= 666000
DMX_BACKDOOR_STOP_FEED= 666001
DMX_REGISTER_CALLBACK= 666999

[Demux]
; LogLevel
; The log file 'Saa7146aDemux.log' is used for logging data
; were the type of messages send to this log file is set by this level
; 0: No log (debug) messages to log file
; 1: -
; 2: Errors
; 3: Main procedures/functions
; 4: Sub procedures/functions
; 5: Intermediate messages
; Note: Only valid if the DLL supports it.
LogLevel=5

; LogClear
; 'Yes' clears the log (only when logging active)
; 'No' appends messages to the log
LogClear=Yes

; Device
; Device to use (0=1st, 1=2nd)
Device=0

; TSBufferSize
; Size of buffer to accomodate (in ms, max 5000)
TSBufferSize=500
```