**DVB-S2 Transport-Streams mit BDA-Geräten anzapfen**

# DVB-S2, HDTV, H.264 & BDA

**Programmierer von TV-Software werden zunehmend mit BDA-Treibern konfrontiert, da BDA die Grundlage dafür ist, dass Windows Media Center die TV-Receiver-Geräte erkennt und einbinden kann. Was BDA bedeutet und wie man sich mit BDA-Geräten (PCI-Einsteckkarten oder USB-Boxen) Zugang zu digitalem SAT-TV beschaffen kann, beantwortet dieser Artikel.** **Peter Thömmes Oktober 2007**

## Digitales Fernsehen

In den 90ern haben Satelliten-Betreiber wie SES ASTRA (Luxemburg) und Eutelsat (Frankreich) in Europa begonnen ihre TV-Übertragungen von ANALOG auf DIGITAL umzustellen. Es gibt immer noch analoge TV-Programme, aber die Tage dieser Technik sind gezählt. Wenn man digitale TV-Programme austrahlt, kann man durch Anwendung effizienter Komprimierverfahren (*MPEG-2* und *MPEG-4*) bis zu 8 Fernsehsendungen auf einem physikalischen Kanal (Transponder) übertragen. Digitales Fernsehen ist in Europa spezifiziert durch den Standard *DVB* (*Digital Video Broadcasting*). Für die verschiedenen Übertragungsmedien gibt es verschiedene Teil-Standards: *DVB-S* und *DVB-S2* (Satellite) für Satelliten-Fernsehen, *DVB-T* (Terrestrial) für terrestrisches Fernsehen und *DVB-C* (Cable) für Kabel-Fernsehen. Für Handheld-Geräte gibt einen gesonderten Standard: *DVB-H*. In den USA hat man neben DVB ein anderes digitales Fernsehen, das durch die Norm *ATSC* (*Advanced Television Systems Committee*) standardisiert ist und den analogen Standard *NTSC* (*National Television Standards Committee*) ablöst.

## Microsoft TV Technologies

Microsoft's Entwickler begannen anfangs des Jahrtausends die kompletten Dekoder für ATSC, DVB-C, DVB-T und DVB-S unter der Überschrift *Microsoft TV Technologies* in ihr *DirectX* aufzunehmen. So wurde ein einheitliches Interface zur Ansteuerung von TV-Adaptern geschaffen, ähnlich wie Anfang der 90er mit ODBC ein einheitliches Interface zur Datenbank-Ansteuerung eingeführt wurde. Das heißt für den Anwendungs-Programmierer: nur einmal die Ansteuerung der Hardware programmieren und dann den Code immer wieder verwenden. Als Name für dieses Interface wählte Microsoft *BDA*, was für *Broadcast Driver Architecture* steht.

*DirectX 1.0* erschien im September 1995 mit Windows 95 als sogenanntes *Windows 95 Game SDK*. Im September 1999 kam mit Windows 2000 (NT5.0) die Version *DirectX 7.0* auf den Markt und *DirectX 8.0* folgte Ende 2000 mit der Xbox. *DirectX 8.1* kam mit XP (NT5.1) im Jahr 2001 und enthielt bereits die BDA Technologie und alles, was man benötigt um digitale TV-Karten anzuzapfen. Eine rundum vollständige Version bezüglich BDA und DVB-S wurde im Dezember 2002 mit Windows 2003 Server (NT5.2) als *DirectX 9.0* herausgebracht. Im September 2003 kam schließlich das letzte unter Windows 2000 einsetzbare *DirectX 9.0 SDK-Update* heraus. Mit Vista (NT6.0) wurde dann im November 2006 auf *DirectX 10.0* hochgezogen.

Es ist also so, dass die Minimalanforderungen an die Entwicklungsumgebung für BDA-kompatible DVB-S2 TV-Software so aussehen:

- HW: DVB-S2 PCI Karte
- DRV: BDA-Treiber
- BS: Windows NT5.0 + sp4
- IDE: Visual C++ 6.0 + sp6
- SDK: NT5.0 SDK Feb 2003
- DX: DirectX 9.0 SDK Sep 2003

Das ist alles. Mit dieser Umgebung kann man nun eine beliebige TV-Software entwickeln, die auf allen NT Betriebssystemen ab Version 5.0 (Windows 2000) läuft. Dazu muss man zwar zuerst einen kleinen Hürdenlauf erfolgreich absolvieren (siehe weiter unten), kann sich dann aber voll auf die H.264/AVC-Dekoder-Programmierung stürzen oder einfach gekaufte Dekoder einbinden. Dann schließlich muss man eine ansprechende Benutzeroberfläche für seine SAT-TV-Software entwickeln.

## DVB-S2, HDTV und H.264/AVC

Da die Begriffe DVB-S2, HDTV und H.264/AVC oft nicht klar getrennt werden, werden sie hier kurz erläutert:

*DVB-S2* beschreibt eine Übertragungstechnik für digitale TV-Signale im Satellitenbereich. Es beschreibt im wesentlichen Modulation (*QPSK*/*8PSK*), *Forward Error Correction* (*FEC*) und die zusätzlich eingefügte Service-Information (*SI*). Multiplexing und Kompression werden von *MPEG-2* übernommen. Im Gegensatz zu DVB-S erlaubt DVB-S2 nicht nur QPSK-Modulation, sondern auch 8PSK-Modulation. *QPSK* ist eine Phasen-Modulation mit 4 Verschiebungswinkeln und somit vierwertigen Übertragungssymbolen. Das heißt, man kann 2 Bit mit einem Symbol (also mit einer phasenverschobenen Schwingung) übertragen. Bei *8PSK* werden 8 Verschiebungswinkel benutzt, also achtwertige Übertragungssymbole bzw. 3 Bit pro Symbol. Das sieht zunächst nach doppelter Übertragungsgeschwindigkeit aus, jedoch muss wegen der höheren Fehlerrate mehr Redundanz zur Fehlerkorrektur eingebaut werden. Außerdem werden auf unterster Schicht vor dem Scrambling noch PLHEADER und Pilot-Blocks eingefügt, was aber maximal 3% wegnimmt. So gewinnt man ungefähr 30% beim Datendurchsatz.

Normalerweise werden DVB-S/S2-Signale mit einer Symbolrate

von 22,0 oder 27,5 MSymbols/s übertragen. Bei QPSK benutzt man eine äußere Fehlerkorrektur (Reed Solomon) mit einem Nutzdaten/ Bruttodaten-Verhältnis von 188/204 und eine innere Fehlerkorrektur mit einem 5/6 Verhältnis bei 22,0 MSymbols/s und einem 3/4 Verhältnis bei 27,5 MSymbols/s. Bei kleiner *QPSK* Übertragungsgeschwindigkeit arbeitet man also mit 2 x 22,0 MBit/s = 44 MBit/s brutto und (188/204) x (5/6) x 44 MBit/s = *33,79 MBit/s* netto. Bei großer Übertragungsrate sind es 2 x 27,5 MBit/s = 55 MBit/s brutto und (188/204) x (3/4) x 55 MBit/s = *38,015 MBit/s* netto. Bei einer *8PSK* Übertragung mit 22,0 MSymbols/s und 2/3 Verhältnis für die gesamte Fehlerkorrektur und etwas Abzug, wegen PLHEADER und Pilot-Blocks, werden aus den 3 x 22,0 MBit/s = 66 MBit/s brutto noch etwa *43 MBit/s* netto.

Auf Empfängerseite übernimmt ein Tuner-Chip (z.B. stb6100 oder cx24118) das Tunen auf die gewünschte Frequenz, wobei der LNB die Ku-Band-Frequenz (10,70 - 12,75 GHz) für den Tuner auf das L-Band (0,95 - 2,15 GHz) umsetzt. Unterhalb 11,725 GHz benutzt der LNB dazu eine LOF (*Local Oscillator Frequency*) von 9,75 GHz (Low) und darüber 10,60 GHz (High). Der Tuner-Chip stellt die LOF und die gewünschte Polarisation (H/V linear) über Gleich- oder Wechselspannung am LNB ein

- 13V/= → V, Low
- 18V/= → H, Low
- 13V/22kHz → V, High
- 18V/22kHz → H, High

und lockt seinen PLL auf die entsprechende Zwischenfrequenz im L-Band, die sogenannte SAT-ZF ($f_{ZF}$ = f - LOF). Frequenz, zusammen mit LOF und Polarisation, adressieren also eindeutig einen Empfangskanal (Transponder). Der Tuner-Chip gibt das herausgefilterte Signal weiter an den Demodulator-Chip (z.B. stb0899 oder cx24116), der aus den 4-wertigen QPSK-Symbolen bzw. 8-wertigen 8PSK-Symbolen 2-wertige Bits macht, die den Demodulator mit ca. 33 bis 43 MBit/s in Form von 188 Byte großen MPEG-2 Packeten (fertig fehlerkorrigiert, also frei von FEC) verlassen.

DVB-S2 ist also für den PC reine *Hardwaresache*, d.h. dieser Part wird voll und ganz von der DVB-S2 Empfänger-Karte erledigt.

*HDTV* (*High Definition TV*) beschreibt den Bildaufbau der Videodaten. HDTV-Filme haben immer 16:9 Format (also nicht wahlweise 4:3 oder 16:9). Die Darstellung geschieht entweder progressiv (p) oder interlaced (i). Dabei gibt es die Möglichkeit 720 oder 1080 Zeilen darzustellen. Abkürzend bezeichnet man die Bildformate der HDTV-Geräte so:

- 720p → 1280 x 720, progressiv
- 1080i → 1920 x 1080, interlaced

Die Bilder werden mit bis zu 60 Hz Bildwiederholfrequenz (Vertikalfrequenz) angezeigt. Die physikalische Auflösung vieler Bildschirme ist jedoch an die Computer-Industrie angelehnt und benutzt *WXGA* mit 1366 x 768 Pixeln, was man auch schon mal mit *768p* abkürzt. HDTV kann durchaus über DVB-S, also mit QPSK übertragen werden. Es ist nicht zwingend notwendig dazu ein DVB-S2 Übertragungsmedium zu haben. Zum Beispiel wird *ASTRA HD 01* auf dem DVB-S Transponder 119 ausgestrahlt.

In der DirectShow-Welt ist HDTV *Sache des Video-Renderers*, also dem letzten Filter in der Kette der Videodaten-Verarbeitung.

Bemerkung: HDTV Bildschirme werden normalerweise mit HDMI Schnittstelle ausgeliefert. *HDMI* (High Definition Multimedia Interface) ist eine digitale Schnittstelle zur Übertragung von Audio- und Video-Daten in der Unterhaltungselektronik. HDMI spezifiziert *HDCP* (High-bandwidth Digital Content Protection) als Kopierschutz. HDCP ist ein von Intel entwickeltes kryptographisches Verschlüsselungssystem für die digitalen Schnittstellen *DVI* und *HDMI*. Es dient zur geschützten Übertragung von Audio- und Video-Daten. Mit HDCP soll das Abgreifen des Video- und Audio-Materials innerhalb der Verbindung zwischen Sender und Empfänger verhindert werden.

*H.264/AVC* beschreibt die Videodaten-Komprimierung. *H.264/ AVC* (*Advanced Video Coding*) ist eine Empfehlung bzw. ein de facto Standard der *VCEG* (*Video Coding Experts Group*) von *ITU-T* bzw. der *MPEG* (*Moving Pictures Expert Group*) von *ISO/IEC*. Beide zusammen starteten 2001 die Entwicklung dieses Standards und gründeten dazu das Team *JVT* (*Joint Video Team*). ISO/IEC benennt den Standard *ISO/IEC 14496-10* (*MPEG-4 Part 10*). Herkömmliche digitale TV-Übertragungen benutzen *H.262* bzw. *ISO/IEC 13818-2* (*MPEG-2 Part 2*) zur Video-Komprimierung, während HDTV in der Regel mit H.264/AVC komprimiert wird. DVB-S/S2 packt jedoch das Ergebnis dieser MPEG-4 Komprimierung für die Übertragung wieder in einen MPEG-2 Transport-Stream ein.

Man kann nicht zwingend sagen, dass H.264/AVC-komprimierte Videodaten unbedingt ein DVB-S2-Übertragungsmedium voraussetzen. Eine solche Komprimierung kann ebenso über das Medium DVB-S übertragen werden. Auch ist es nicht zwingend so, dass man HDTV-Videodaten mit H.264/AVC komprimieren muss, aber bei einer verfügbaren Gesamtbitrate von ca. 40 MBit/s, die netto auf einem DVB-S2 Transponder mit einer Übertragungsgeschwindigkeit von 22 MSymbols/s rest bleibt, kommt man auf keinen Fall ohne Komprimierung aus. Denn die 1920 x 1080 Bildpunkte von HDTV mit einer Farbtiefe von 24 Bit würden bei einer Bildwiederholfrequenz von nur 30 Hz bereits 1920 x 1080 x 24 x 30 Bit/s = 1,49 GBit/s, also die 37-fache Geschwindigkeit benötigen. Ohne Zweifel benötigt man also Komprimierung. Um viele Endnutzer zu erreichen, muss man einen gut definierten Standard zugrunde legen, damit auch möglichst viele Hersteller Dekoder entwickeln und somit ihren Kunden Zugang zu den Videodaten verschaffen können.

H.264/AVC ist also *Sache des Video-Dekoders* (auch *Codec* genannt), welcher unter DirectX durch einen weiteren DirectShow-Filter realisiert wird. Die ersten H.264/ AVC Video-Dekoder für den PC Bereich stellten *Cyberlink* mit ihrem *PowerDVD* und *InterVideo* mit ihrem *WinDVD* bereit.

## DVB-S2 PCI Karten

Es ist schwierig eine vollständige Auflistung verfügbarer PCI-Karten für DVB-S2 zu machen. Vor allem ist die Verfügbarkeit von Hardware und Treiber nur überprüfbar, wenn man die Dinge wirklich bestellt und testet. Dabei stellt sich dann oft heraus, dass es sich bei der Werbung nur um eine Luftblase handelt, das Gerät aber weder verfügbar noch entwickelt ist. Für diesen Erfahrungsbericht wurden zwei, fast überall verfügbare, Karten benutzt:

- SkyStar HD von TechniSat
- TT S2-3200 von TechnoTrend

Die *SkyStar HD* und die *TT S2-3200* sind Baugleich und benutzen als Tuner den *stb6100* und als Demodulator den *stb0899* von *ST Microelectronics* (ein anderes bekanntes Paar sind der Tuner *cx24118* und der Demodulator *cx24116* von *Conexant*). Beide PCI Karten werden mit TechnoTrend Treibern, aber leicht unterschiedlicher TV-Software ausgeliefert (Skin ist etwas anders). Hauptunterschied der Software ist der Bundle: TechniSat legt MainConcept's EVE (Easy Video Editor) mit bei, während TechnoTrend eine Vollversion von CyberLink's PowerDVD7 drauf legt, also nicht nur deren H.264/AVC Dekoder benutzt.

## Windows Media Center

Mit *Windows XP Media Center Edition 2004* brachte Microsoft erstmals eine vollständige graphische Benutzeroberfläche zum Anschauen und Editieren von Bildern, Musik und Filmen heraus. Wenn dabei als Quelle für einen Film nicht eine DVD oder eine Datei verwendet werden soll, dann kann man sich eine TV-Karte aussuchen und TV-Programme anschauen. Großer Nachteil bei der Sache: nur terrestrisches Fernsehen wird unterstützt - und das wurde bis heute nicht geändert. Möchte man also SAT-TV schauen (DVB-S/S2), dann muss dieser Media Center Software eine *DVB-T* Karte vorgegaukelt werden, was nur mit speziellen *MCE Tools* möglich ist, die von Drittanbietern angeboten werden. Für diejenigen, die noch auf Windows 2000 arbeiten oder lieber andere TV-Software benutzen, gibt es aber immer noch die mitgelieferte Software des Karten-Herstellers. Zudem findet man auch Freeware- und Shareware-Programme, wie ProgDVB 4.85.1

http://www.progdvb.com/

und DVBViewer 3.6.1.20

http://www.dvbviewer.com/.

Beim ersten Test einer TV-Software sollte man unbedingt den Task-Manager mitlaufen lassen und die CPU beobachten. Erstens wegen der Effizienz des H.264/AVC-Dekoders, und zweitens wegen der Effizienz der Treiber-Ansteuerung. Es sieht so aus, als ob ältere BDA-Treiber noch teilweise herstellerspezifische WDM-Schnittstellen unterstützen, da der Hersteller nicht von jetzt auf gleich alles auf BDA umstellen kann. So kann es sein, dass die Software des Gerätebauers effizienter läuft als die von Drittanbietern, da Drittanbieter nur die offene BDA-Schnittstelle nutzen können.

## BDA vs. WDM

Der Unterschied zwischen *BDA* (*Broadcast Driver Architecture*) und *WDM* (*Windows Driver Model*) ist sehr groß. In diesem Artikel wird aber lediglich der Unterschied aus der Sicht des Anwendungs-Programmierers herausgearbeitet.

Bei *WDM* wird ein Gerät mit *CreateFile()* geöffnet und mit *CloseHandle()* geschlossen. Das Setzen und Lesen der Steuer-Parameter geschieht mit der SDK-Funktion *DeviceIoControl()*. Diese wird mit einem sogenannten IOCTL-Code als Parameter aufgerufen. Ein *IOCTL-Code* entspricht einer bestimmten Funktion, die das Gerät (der Treiber) ausführen soll und es liegt am Hersteller des Gerätes für jede Funktion einen IOCTL-Code zu definieren. Abhängig von diesem Code sind [IN]-Parameter als Steuerdaten für den Treiber und/oder [OUT]-Parameter als Rückgabe-Container für Statusdaten vom Treiber mitzugeben. Die Nutzdaten-Übertragung geschieht bei WDM über *ReadFile()* und *WriteFile()* und bei asynchroner Übertragung über *ReadFileEx()* und *WriteFileEx()* und den dazugehörigen IoCompletion-Callback-Funktionen. Dies alles ist von Anwendungssicht ganz so, wie es auch unter UNIX ist, dort sind lediglich die SDK-Funktionen etwas anders benannt: *open()*, *close()*, *ioctl()*, *read()*, *write()*, *aio_read()*, und *aio_write()*. Beim Ausliefern von Software, die auf WDM-Treibern basiert, müssen also lediglich die Treiber und die DLLs der verwendeten Laufzeitumgebung dazu kopiert werden (falls nicht schon in aktuellerer Version auf dem Zielsystem vorhanden).

Bei *BDA* sieht das komplett anders aus. Hier muss man tief in die Welt von *DirectX* abtauchen um zu verstehen, worum es geht. Daneben benötigt man zusätzlich zur normalen Entwicklungsumgebung noch das *DirectX SDK*. Das letztes SDK, welches noch einwandfrei unter

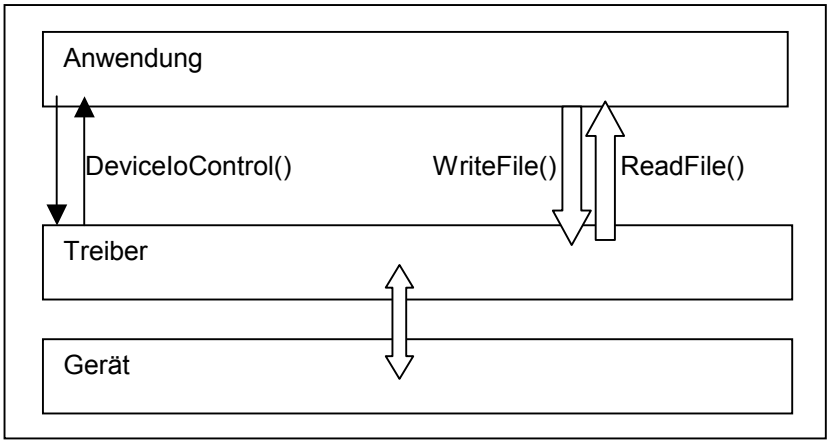


**Bild1: Kommunikation mit Geräten über WDM-Treiber**

Windows 2000 mit Visual C++ 6.0 arbeitet ist das vom September 2003 (9.00.1126). Die Versionen vom Oktober 2004 und Dezember 2004 installieren zwar noch, funktionieren aber nur noch teilweise. Obwohl die Version vom September 2003 DVB-S-mäßig voll implementiert ist, hat man beim bauen einen Fehler im Inline-Assembler von Visual C++ 6.0, da dieser den uralten x86 Befehl *STD* (*Set Direction*) nicht kennt. Das ist seltsam, vor allem da er schließlich *CLD* (*Clear Direction*) kennt. Wie man das Problem fixt wird weiter unten gezeigt. Die Ansteuerung von BDA-Geräten funktioniert also noch mit Visual C++ 6.0 unter Windows 2000, solange der Gerätehersteller noch BDA-Treiber für dieses Betriebssystem bereit stellt.

DirectX ist im Prinzip weiter nichts als COM-Objekte (COM zieht sich seit Mitte der 90er im Hintergrund durch nahezu alles, was Microsoft so macht, auch wenn man es oft nicht sieht). Diese DirectX-COM-Objekte müssen lediglich in das DirectX Framework passen, also bestimmte Einsprungstellen in den Binärdateien aufweisen und die Interface-Spezifikationen für die Anknüpfpunkte (Pins) erfüllen. So wird jedes dieser Objekte zu einem sogenannten *Filter*. Die Binär-Dateien dieser Filter haben Namen mit der Endung *.ax*. Filter, die zur Kodierung/Dekodierung dienen nennt man auch *Codec* (Coder/Decoder).

Bei DirectX spricht man also von Filtern, wie in der Systemtheorie oder Nachrichtentechnik. Jeder Filter hat mindestens einen Pin mit dem er an mindestens einen anderen Filter angeschlossen werden kann. Man benötigt ein sogenanntes *Graphbuilder*-Objekt um die Filter zu laden und zu verknüpfen. Das bedeuted in letzter Konsequenz, dass COM-Objekte (.ax-Dateien) per 128-Bit Schlüsssel (GUID) über die Registry aufgefunden und in den Speicher geladen und dann ihre Pins verknüpft werden. Pins verknüpfen heißt, dass ein Filter, der Ausgangsdaten an seinem Ausgangs-Pin produziert, mitgeteilt bekommt, wo er diese abzuliefern hat (Eingangs-Pin des nächsten Filters). Der Filter wird dann das COM-

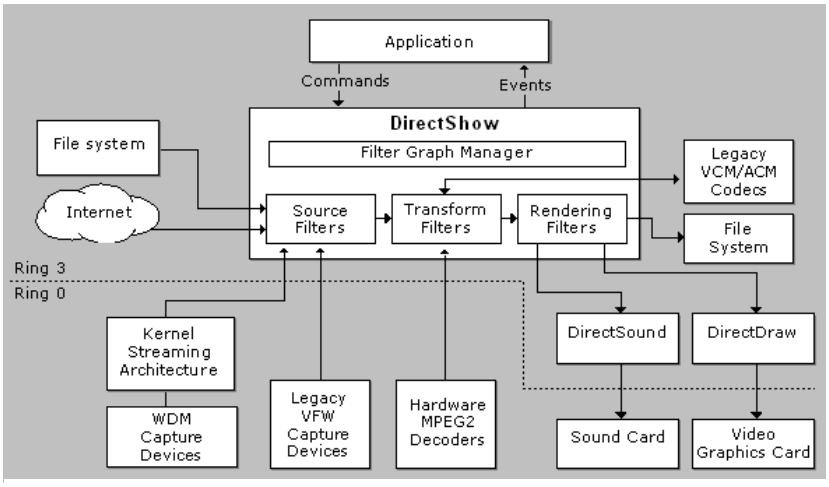


**Bild2: DirectShow (Quelle: Microsoft, MSDN-Library)**

Interface des nächsten Filters benutzen um dort eine Datenablieferungs-Funktion am Eingangs-Pin aufzurufen. Sind alle Filter verknüpft, dann spricht man von einem Filter-Graphen.

DirectX TV-Software folgt den Spezifikationen für *DirectShow* (Bild 2), was ein Teilbereich von DirectX ist und früher *ActiveMovie* genannt wurde (auch unter dem Codenamen *Quartz* bekannt). Jeder DirectShow Filter-Graph besteht aus einer Quelle (*Source*) und ein oder mehreren Umwandlern (*Transformer*) und einem oder mehreren Bild/Ton-Darstellern (*Renderer*). Details dazu weiter unten. Hier nur soviel: es muss ein Graph aufgebaut werden, dann muss dieser initialisiert werden und dann muss er gestartet werden. Erst dann kann die SAT-Karte Informationen über die Signalqualität und/oder Daten abliefern, die dann im weiteren dekodiert und zur Ausgabe gebracht werden. Alles in allem führt dies dazu, dass man nach dem Entwickeln der Software nicht nur (wie gewohnt) die Treiber und die Laufzeit-DLLs mit ausliefern muss, sondern auch die wiedervertreibbare (redistributable) DirectX Laufzeitumgebung (min. 37 MB) und alle selbst entwickelten oder zugekauften DirectX-Filter (.ax-Dateien). Nach dem kopieren müssen die Filter dann auf jeden Fall mit ihrer GUID registriert werden, da sie als COM Objekte darüber geladen werden. Da Microsoft keinen DVB-S Tuning-Space definert hat, muss dieser dann noch in die Registry eingetragen werden, da dies nicht automatisch mit der Treiberinstallation einhergeht. Dazu legt der Gerätehersteller normalerweise eine *.reg*-Datei zum Treiber hinzu, die dann *REGEDIT.EXE* anweist die Einträge unter

HKEY_LOCAL_MACHINE\
SOFTWARE\Microsoft\
Tuning Spaces

anzulegen. Dort wird dann ein DVB-S Tuning Space mittels ID (z.B. 42) angelegt und als Unterzweig auch noch ein Default Locator.

Tabelle 1 und 2 zeigen eine kurze Zusammenfassung der Vor- und Nachteile der BDA- und WDM-Treiber-Technologie.

| |
|---|
| Vorteile |
| Erforderliches Wissen zur Applikations-Programmierung kann man sich in sehr kurzer Zeit aneignen |
| Robustheit hängt nur von wenigen Komponenten ab: Hardware, Betriebssystem, Treiber und Anwendung |
| Alle Hardware Features sind nutzbar |
| Nachteile |
| Kein einheitliches Interface für gleichartige Hardware |
| Jedes Gerät erfordert eine Neuentwicklung (zumindest eines Wrappers) |
| Geräte funktionieren nicht unter Windows Media Center |

**Tabelle 2: WDM-Treiber**

| BDA-Treiber |
|---|
| Vorteile |
| Einheitliches Interface und daher einmaliger Entwicklungsaufwand (bis auf kleinere Anpassungen) |
| Geräte können unter Windows Media Center eingebunden werden |
| Man kann Geräte ansteuern, ohne vom Hersteller eine offene Spezifikation zu haben. |
| Nachteile |
| Erforderliches Training zur Applikations-Programmierung ist mit intensivem Lernen verbunden |
| Spezielle Hardware Features werden nicht unterstützt |
| DirectX Laufzeitumgebung muss mit ausgeliefert und installiert werden (min. 37 MB) |
| Eigene und/oder zugekaufte DirectX Filter müssen mit ausgeliefert und installiert werden |
| Mögliche Anfälligkeit gegenüber Upgrades von DirectX Laufzeitumgebung und Filtern |
| Robustheit hängt von vielen Komponenten und deren Version ab: Hardware, Betriebssystem, Treiber, DirectX Laufzeitumgebung / Filter & Anwendung |

**Tabelle 1: BDA-Treiber**

## BDA - Die Hürden

Da bei BDA viele Technologien ineinander greifen, es also viele Schnittstellen und viele Beteiligte gibt, gilt auch hier, was sonstwo in dem Fall gilt: viel Köche verderben den Brei. Das führte in der Vergangenheit dazu, dass der Prozeß der Einführung einer vereinheitlichten TV-Software sich lange hinzog und sich letzendlich immer noch als eine Bastellösung darstellt.

**Hürde Nummer 1**: Fangen wir unten an, nämlich bei der Hardware. Da die DVB-S2 Demodulator-Chips für den Konsum-Elektronik Bereich spezielle Neuentwicklungen sind, sind noch die ein oder anderen Fehler nicht bereinigt bzw. nicht bekannt. So zeigte sich beim *stb0899*, dass die *Spektrale Inversion* falsch herum gehandhabt wird. Zum Beispiel lockt der stb0899 nicht auf SES ASTRA's DVB-S2 Transponder 119, wenn man dem Chip mitteilt, dass dort keine spektrale Inversion vorliegt, was der richtigen Einstellung entspricht, denn im Ku-Band wird keine spektrale Inversion vorgenommen. Spektrale Inversion macht man im C-Band (3,6 - 4,2 GHz), da dort die LOF (5,15 GHz) höher als die Empfangsfrequenzen ist. Der stb0899 vertauscht also konsequent invertiertes und nichtinvertiertes Spektrum. Die Lösung für dieses Problem: Spektrale Inversion IMMER auf AUTOMATIC stellen und den Benutzer der Software gar nicht erst wählen lassen, denn es könnte ja sein, dass der Treiber-Programmierer die Sache in Zukunft dreht und durch ein Treiber-Upgrade eine Fehlerbehebung auf Anwendungsebene wieder aufhebt.

**Hürde Nummer 2**: Etwas höher im Schichtenmodell finden wir dann den Treiber und darüber DirectX. Microsoft's DirectX/BDA Spezifikation kennt kein DiSEqC. *DiSEqC* (*Digital Satellite Equipment Control*) ist ein einfaches serielles Protokoll zur Umschaltung auf einen anderen LNB bei Multifeed-SAT-Anlagen (das Signal geht, wie die 13V/18V zur Umstellung der Polarisation V/H bzw. 0Hz/22kHz zur Umschaltung des Bandes Low/High, raus auf das Antennenkabel und hin zur DiSEqC-Umschaltbox). Da nun für die Übergabe eines DiSEqC-Kommandos kein Interface in DirectX/BDA existiert, muss der Gerätehersteller seinen Treiber-Programmierer anweisen eine Ersatzlösung zu finden. Das sieht dann schon mal so aus, dass

*IBDA_FrequencyFilter::put_Range()*

zur LNB-Umschaltung per DiSEqC benutzt wird. Dazu wird das *IBDA_FrequencyFilter* Interface benutzt, welches man über den TUNER-Proxy bekommt, wenn man den Control-Node vom Type 0 (Tuner-Chip) über das COM-Interface IBDA_Topology kontaktiert

*IBDA_Topology::*
*GetControlNode(0,1,0,&pINode)*

und mit *QueryInterface()* nach

*IBDA_FrequencyFilter*

sucht, welches das Interface zum stb6100 ist. Danach ruft man die Funktion *put_Range()* auf und aktiviert per *unsigned long* Parameter den gewünschten LNB. Man muss den Gerätehersteller kontaktieren, um zu erfahren, wie der *unsigned long* Parameter zu belegen ist. Achtung: *IBDA_FrequencyFilter::put_Range()* darf erst aufgerufen werden, nachdem der Filter-Graph fertig aufgebaut wurde, sonst funktioniert die Funktion nicht und liefert ein *E_HANDLE* zurück.

**Hürde Nummer 3**: Bleiben wir beim Zusammenspiel Treiber und DirectX. Ebenfalls nicht in der BDA Spezifikation enthalten ist ein gültiger Parameter für die Modulationsart *8PSK*. Microsoft hat viele Modulationsarten vorgesehen, aber eben nicht 8PSK. Da man den Wert aber irgendwie durch die DirectX Laufzeitumgebung an den Treiber bzw. den Demodulator-Chip senden muss, empfiehlt es sich, einen anderen Wert aus dem gültigen Wertebereich zu nehmen. Hier hat wieder mal der Treiber-Programmierer die Wahl und er wird sich normalerweise für *8VSB* (*BDA_MOD_8VSB*) entscheiden, da dies die Modulation für das amerikanische digitale ATSC ist und dies garantiert nicht in Konflikt gerät mit DVB-S2. Man muss also wieder den Gerätehersteller kontaktieren und erfragen, welchen Wert sein Treiber als *8PSK* Modulation interpretiert.

**Hürde Nummer 4**: Nochmal die Kommunikation zwischen Treiber und DirectX. Der Satelliten-Betreiber kann frei entscheiden, ob er *Pilot-Blocks* mitsendet oder nicht. Pilot-Blocks sind 36 unmodulierte Schwingungen, die an festen Stellen des DVB-S2 Übertragungsschemas eingefügt werden um die Demodulatoren beim synchronisieren zu unterstützen. Bei manchen Demodulator-Chips muss eingestellt werden, ob das Signal Pilot-Blocks beinhaltet oder nicht. Stimmt diese Einstellung nicht, dann lockt der Demodulator sich nicht in das Signal. Auch hier fehlt die BDA Spezifikation und nur der Gerätehersteller kann Auskunft geben, wie er das Problem umgeht. Beim Chip stb0899 jedoch wird dies automatisch erkannt, wodurch sich das Problem erübrigt. Beim cx24116 hingegen muss es richtig eingestellt werden.

**Hürde Nummer 5**: Schaffen wir uns ein bisschen höher. Diesmal ist es nicht die DirectX/BDA Spezifikation, die uns zu schaffen macht, sondern die Implementierung von DirectX. Der Wert für die Modulation wird nicht auf allen Plattformen bis zum Treiber durchgereicht. Dies ist z.B. unter Windows 2000 so, also liegt der Fahler wahrscheinlich in der Datei *BDANT.cab*. Die Interface-Funktion

*IDVBSLocator::put_Modulation()*

soll den BDA Network Provider anweisen, über den TUNER-Proxy die Modulation einzustellen. Beim Treiber kommt aber nur der Wert 0 an, was immer man auch hinschickt. Dieser Wert ist gleichbedeutend mit *BDA_MOD_NOT_DEFINED*. So wird dann normalerweise auf QPSK getippt, was den Dienst für 8PSK aber nicht tut. Man muss hier einen anderen Weg benutzen, nämlich den über die Funktion

*IBDA_DigitalDemodulator::*
*put_ModulationType()*.

Das *IBDA_DigitalDemodulator* Interface bekommt man über den TUNER-Proxy, wenn man den Control-Node vom Type 1 (Demodulator-Chip) über das COM-Interface IBDA_Topology kontaktiert

*IBDA_Topology::*
*GetControlNode(0,1,1,&pINode)*

und mit *QueryInterface()* nach

*IBDA_DigitalDemodulator*

sucht, welches das Interface zum stb0899 ist. Danach ruft man die Funktion *put_ModulationType()* auf und setzt die Modulation. Achtung: Auch hier gilt, dass man die Funktion erst aufrufen kann, nachdem der Filter-Graph fertig aufgebaut wurde, sonst gibt's ein *E_HANDLE* zurück.

**Hürde Nummer 6**: Noch eine Schicht höher finden wir die DirectShow-*BaseClasses*, die Klassen-Bibliothek von DirectShow. Möchte man rückwärts-kompatibel zu Windows 2000 programmieren (es gibt noch einige Windows 2000 Systeme) und auch Visual C++ 6.0 benutzen (Service Pack 6 sollte installert sein), dann bekommt man Schwierigkeiten bei der Kompilierung der BaseClasses: Der *Inline Assembler* kennt die x86 Instruction *STD* (*Set Direction*) nicht, die das DF-Flag (Bit 10) im FLAGS Register der CPU setzt. Dieses Flag steuert die Behandlung der Index-Register ESI und/oder EDI während String-Operationen: ist DF gesetzt (also 1), dann werden die Index-Register dekrementiert, andernfalls inkrementiert. Der Befehl STD hat den OpCode 0xFD, also hat man folgende Möglichkeit zum Fixen des Problems: in der Datei *wxutil.cpp* in der Funktion *memmoveInternal()* wird *std* durch *_emit 0xFD* ersetzt:

```
_asm{
      mov     esi,src
      mov     edi,dst
      mov     ecx,count
      _emit   0xFD
      add     esi,ecx
      add     edi,ecx
      dec     esi
      dec     edi
      rep     movsb
      cld
}
```

Mit dieser Änderung baut der Code und wir haben die DirectShow BaseClasses, wie wir sie brauchen.

**Hürde Nummer 7**: Nochmals BaseClasses. Möchte man einen Grabber Filter programmieren und folgt dem Beispiel von Microsoft in

*...\Samples\C++\DirectShow*
*Filters\Grabber*

dann wird man beim RELEASE-Bau ordentlich gegen die Wand laufen: CRASH. Der Grund ist die Callback Funktion, die nicht als *CALLBACK* definiert wurde. Entgegen dem Beispiel muss der *typedef* wie folgt aussehen:

```
typedef HRESULT (CALLBACK
         *SAMPLECALLBACK)
(
IMediaSample* pSample,
REFERENCE_TIME* StartTime,
REFERENCE_TIME* StopTime,
BOOL TypeChanged
);
```

Ohne diese Änderung wird der Filter (COM-Objekt) im DirectShow-Graph nie auf die richtige Adresse in der Applikation zugreifen, denn beide laufen in verschiedenen Adreßräumen.

## BDA Software Entwicklung

Nachdem nun alle Hürden überwunden wurden, geht es an die Software-Entwicklung. Wenn man das BDA-Gerät lediglich als Transport-Stream-Lieferant benutzen will, also so, wie man die herkömmlichen Geräte mit WDM-Treiber benutzt hat, dann geht man wie folgt vor:

- Grabber-Filter entwickeln
- BDA-Wrapper entwicklen
- SAT-TV-Software entwickeln

*Grabber-Filter*: Wie oben bereits erwähnt, entwickelt man auf der Basis der Beispiel-Software *Grabber* seinen, an die eigenen Bedürfnisse angepaßten Filter, der später die Daten hoch in die Applikation pumpt und beachtet das oben gesagte bezüglich der *CALLBACK*-Funktion. Als Vorbereitung dazu muss das *DirectX 9.0 SDK* installiert werden. Der Übersicht wegen sind folgende Installations-Pfade empfehlenswert:

DirectX9-SDK Dezember 2002
(9.00.0900 2002-12):
*C:\DX90ASDK*

DirectX9-SDK Update Sept. 2003
(9.00.1126 2003-09):
*C:\DX90BSDK*

Dann ergänzt man die Such-Pfade in Visual C++ 6.0:

Tools...Options...Directories...
[Include files]...
*C:\DX90BSDK\INCLUDE*
[Library files]...
*C:\DX90BSDK\LIB*

und achtet darauf, dass sie vor die anderen Einträge in der jeweiligen Liste hochgeschoben werden.

*BDA-Wrapper*: Man schreibt sich eine Klasse, die sich im wesentlichen nur 2 Funktionen hat, nämlich eine zum Initialisieren (hier wird unter anderem der BDA-Filter-Graph auf-

gebaut) und eine um den Datenempfang von einem bestimmten Transponder zu starten: *Initialize()* und *StartRx()* (siehe weiter unten).

*SAT-TV-Software*: Hier hat man grundsätzlich 2 Möglichkeiten: Entweder man bewegt sich komplett in der DirectX-Welt und implementiert die ganze Video-Dekodierung/ Ausgabe als DirectShow-Filter, oder man schreibt den Code plattformübergreifend in reinem C++ und zieht nur den Transport-Stream mittels BDA-Wrapper aus dem Filter-Graph raus in die eigentliche Anwendung. Als Beispiel sollte man sich auf jeden Fall das Open-Source-Projekt *DigitalWatch* anschauen, welches für DVB-T geschrieben wurde (wie Windows Media Center) und hier zu finden ist:

http://nate.dynalias.net/
DigitalWatch/DW2.html

## BDA-Wrapper: Filter-Graph

Wie bereits erwähnt, benötigt man zum Betreiben eines BDA SAT-Receiver-Gerätes einen DirectShow Filter-Graphen, hier BDA-Filter-Graph genannt. Als Quelle (*Source*) ist ein sogenannter *BDA Network Provider* zu verwenden, also sozusagen die TV-Antenne oder SAT-Schüssel mit dem LNB. Die Quelle muss mit Default-Parametern initialisiert werden. Diese kann man aus dem Default Locator im Tuning-Space-Eintrag

HKEY_LOCAL_MACHINE\
SOFTWARE\Microsoft\
Tuning Spaces

der Registry lesen oder einfach so setzen, wie man will. Der Ausgang der Quelle führt dann zum *TUNER-Proxy*, was der zweite Filter in der Kette ist. Im DirectShow Filter-Graph spricht man zwar vom TUNER, meint aber genau genommen:

TUNER :=
Tuner + Demodulator

was also zum Beispiel ein *stb6100* (Tuner) mit einem *stb0899* (Demodulator) sein könnte. Dieser TUNER-Proxy wird mit dem *CAPTURE-Proxy* verbunden, was in Wirklichkeit der Multimedia Bridge zum PCI Bus entspricht, also zum Beispiel dem Chip *SAA7146AH*:

CAPTURE :=
MM-Bridge zum PCI Bus

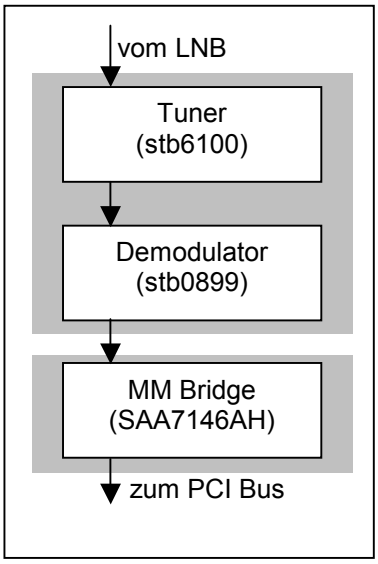


**Bild 4: DVB-S2 Karte**

Beides sind *Kernel-Streaming-Proxies* und im Filter *ksproxy.ax* implementiert. Von dort geht es über den Grabber-Filter zum *MPEG-2 Demultiplexer* (H.264/AVC, also MPEG-4 Part 10, wird bei der DVB-S2 Übertragung ebenfalls in MPEG-2 Packete eingepackt), der Anhand der, im Transport-Stream enthaltenen, Metadaten die Video-, Audio- und Daten-Streams herausfiltern kann. Will man einen BDA-Wrapper entwickeln, dann muss jetzt nur noch ein Filter nachgeschaltet werden, nämlich der sogenannte *TIF* (*Transportstream Information Filter*). Denn dieser wird (im Hintergrund) vom BDA Network Provider kontaktiert um die Metadaten des Transport-Streams auszuwerten. Diese Metadaten nennt man auf MPEG-2-Ebene PSI (Program Specific Information), worin eine sogenannte PAT (Program Association Table) als Wurzel-Tabelle enthalten ist und auf die Packet-IDs (PIDs) aller PMTs (Program Map Tables) hinweißt, welche wiederum auf die jeweiligen PIDs von Video, Audio und Program-Clock-Reference zeigen. Auf DVB-S/S2-Ebene nennt man die Metadaten SI (Service Information). Dort listet die SDT (Service Description Table) die Programme (Services) auf und die EIT (Event Information Table) den Zeitplan für Ausstrahlung der TV-Sendungen (Events). Diese Informationen kann man sich vom *Mpeg2Data* Filter dekodieren lassen, indem man diesen an Pin5 des MPEG-2 Demultiplexers hängt. Wie aber bereits gesagt, ist dies nicht erforderlich um einen BDA-Wrapper zu programmieren. Ebenfalls nicht erforderlich für den BDA-Wrapper sind Video-Dekoder, Video Renderer, Audio-Dekoder und Audio-Ren-

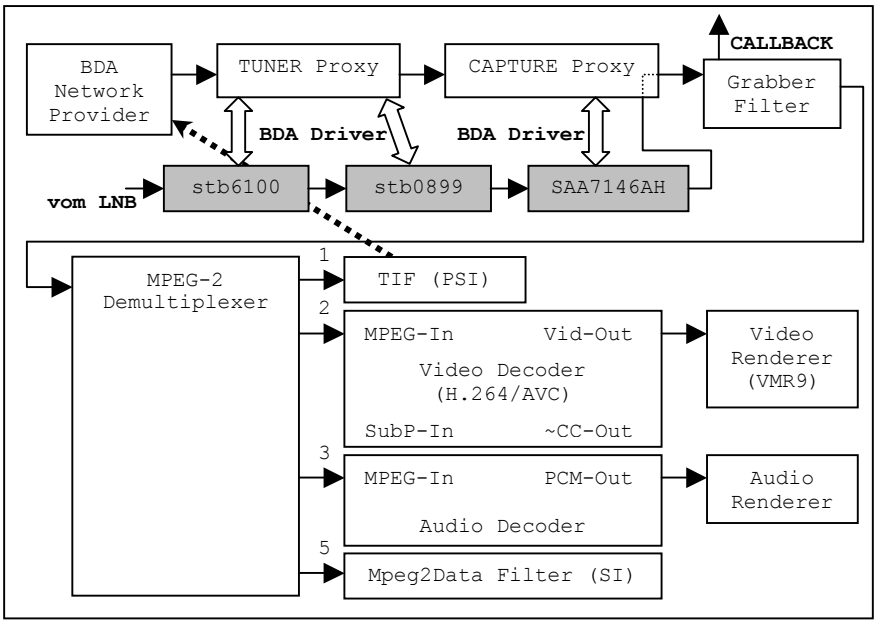


**Bild 3: Vollständiger BDA Filter-Graph**

derer.

Da das Thema sehr wichtig ist, hier noch ein paar Bemerkungen zum **Video-Renderer**. Dieser implementiert das Rausschreiben der Bilder auf den Bildschirm. Das kann über *DirectDraw*, aber auch über das, nicht so effiziente, normale *GDI* (*Graphics Device Interface*) gemacht werden. DirectShow versteckt aber das Handle zum Video-Fenster, damit Applikationen das Abspielen des Videos nicht stören können. Um trotzdem Text oder Graphik über das Video legen zu können wurde der *Overlay Mixer Renderer* entwickelt, welcher *DXVA* (*DirectX Video Acceleration*) unterstützt, also den Zugang zu der 2D-Hardware-Beschleunigung des Graphik-Prozessors. Dieser Koprozessor macht Dinge wie z.B. die *Inverse Diskrete Cosinus Transformation*, *Huffmann Kodierung*, *Farb-Korrektur*, *De-Interlacing* und so weiter. In Windows XP (NT5.1) hat Microsoft zum ersten Mal den *VMR* (*Video Mixing Renderer*) eingebaut. VMR basiert auf dem bereits mit Windows 2000 (NT5.0) ausgelieferten *DirectX 7.0* und wird deshalb auch *VMR7* genannt. Dieser Renderer erlaubt keine GDI Ausgabe mehr. Er unterstützt DXVA und kann mehrere Video-Streams, Graphiken und Texte mixen und erlaubt so den Applikationen Graphiken und Text über das Video zu legen. Obwohl es keinen technischen Grund dafür gab, stellte Microsoft diese Funktionalität nur für Windows XP bereit. Mit *DirectX 9.0* jedoch kam dann ein neues VMR, nämlich *VMR9*, auf den Markt, welches auf *Direct3D9* aufbaut und auf allen Windows NT Betriebssystemen genutzt werden kann. Schließlich wurde für Vista wieder ein neuer Video-Renderer entwickelt und man gab ihm den Namen *EVR* (*Enhanced Video Renderer*). EVR unterstützt das neue *DXVA 2.0*, welches ebenfalls für Vista entwickelt wurde. Wenn man jedoch die *.NET Framework 3.0* Laufzeitumgebung auf Windows XP installiert, dann wird der EVR aber auch dort zur Verfügung gestellt.

### BDA-Wrapper: Initialize()

Nun kommen wir zur ersten der beiden Hauptfunktionen des BDA-Wrappers, nämlich *Initialize()*. Dort wird die COM Laufzeitumgebung und der Tuning-Space initialisiert (Listing 1 und 2) und dann wird der BDA-Filter-Graph aufgebaut.
Zum Aufbau des BDA-Filter-Graphen gehören folgende Schritte:

| Aufbauschritte des BDA Graphen |
|---|
| DVB-S Tuning-Space initialisieren |
| Default DVB Tune-Request erzeugen (Listing 3) |
| BDA Network Provider instanziieren |
| BDA Network Provider durch Übergabe des Default DVB Tune-Requests initialisieren |
| MPEG-2 Demultiplexer instanziieren |
| Grabber-Filter instanziieren |
| CLSID_FilterGraph instanziieren |
| BDA Network Providers laden |
| Laden des TUNER-Proxy aus *KSCATEGORY_ BDA_NETWORK_TUNER* |
| Laden des CAPTURE-Proxy aus *KSCATEGORY_ BDA_ RECEIVER_COMPONENT* |
| Grabber-Filter laden |
| Grabber-Filter CALLBACK setzen |
| MPEG-2 Demultiplexer laden |
| Laden des TIF Systemfilters aus *KSCATEGORY_BDA_ TRANSPORT_INFORMATION* |
| Verbinden von BDA Network Provider und TUNER-Proxy |
| Verbinden von TUNER-Proxy und CAPTURE-Proxy |
| Verbinden von CAPTURE-Proxy und Grabber-Filter |
| Verbinden von Grabber-Filter und MPEG-2 Demultiplexer |
| Verbinden von MPEG-2 Demultiplexer und TIF |

**Tabelle 3: BDA-Graph Aufbau**

Die Erläuterung der Implementierung würde bei weitem den Rahmen dieses Artikels sprengen. Dazu sollte man sich Code-Beispiele ansehen (wie z.B. DigitalWatch) und vor allem auch die MSDN Library hinzuziehen.

### BDA-Wrapper: StartRx()

Zum Code von *StartRx()* ist folgendes zu sagen. Auch hier ist das Wichtigste zu wissen, in welcher Reihenfolge man vorgeht. Und zwar muss man zunächst den Graphen stoppen, wenn er bereits gestartet wurde. Dann ruft man *Release()* auf dem Interface *IDVBTuneRequest* auf und dann erzeugt man ein neues DVB Tune-Request, wobei man sich beim Hersteller der BDA Karte erkundigen muss, welchen Wert dem Treiber für die *8PSK-Modulation* zu übergeben ist. Dann besorgt man sich das *ITuner* Interface des BDA Network Providers und übergibt der Methode *put_TuneRequest()* das Tune-Request. Danach wird die Modulation nochmals auf dem Demodulator-Chip über das Interface *IBDA_DigitalDemodulator und* die Funktion *put_ModulationType()* eingestellt (wie oben bereits erläutert). Und danach wird das *DiSEqC*-Kommando an den Tuner-Chip geschickt. Dazu muss man bei Hersteller erfragen, welchen Weg er dazu vorgesehen hat. TechnoTrend benutzt dazu

*IBDA_FrequencyFilter::put_Range()*

wie man leicht dem Konfigurations-Dialog von TT-Media-Center entnehmen kann. Danach wird dann endlich auf dem *IGraphBuilder* Interface ein Query nach *IMediaControl* gemacht und die

```
HRESULT hr = ::CoInitializeEx(
           NULL,COINIT_MULTITHREADED|COINIT_DISABLE_OLE1DDE);
if(hr == S_OK) { //ok
}
else if(hr == S_FALSE) { //bereits initialized
}
else if(hr == RPC_E_CHANGED_MODE) {
//geaendert/reinitialisiert
}
else {
   ...; //Fehlerbehandlung
}
```

**Listing 1: COM initialisieren**

Methode *Run()* zum Starten des Graphen aufgerufen. Zum kurzen Anhalten gibt es dort die Methode *Pause()* und zum Stoppen die Methode *Stop()*.

## Schlußwort

Wie man leicht erahnen kann, ist eine BDA-Ansteuerung keine Sache, die man an einem Nachmittag verstehen und umsetzen kann. Mit guten COM Kenntnissen, diesem Artikel, Beispielen aus dem Internet und etwas Geduld wird man jedoch mit großen Schritten dem Ziel näher kommen.

Für die kompetente und freundliche Unterstützung bedanke ich mich ausdrücklich bei Alexander Beresowskij, Holger Schönstedt, Steve Lachard und Thomas Wrede.

```
HRESULT hr = m_pITuningSpaceContainer.CoCreateInstance(
                                        CLSID_SystemTuningSpaces);
if(FAILED(hr)) {
   ...; //Fehlerbehandlung
}
CComBSTR bstrTuningSpaceString(L"TT-DVB-S");
CComPtr<IEnumTuningSpaces> pIEnumTuningSpaces;
hr = m_pITuningSpaceContainer->get_EnumTuningSpaces(
                                              &pIEnumTuningSpaces);
while(S_OK == pIEnumTuningSpaces->Next(
                    1,(ITuningSpace**)
&m_pITuningSpace,NULL))
{
   CComBSTR bstrTemp;
   hr = m_pITuningSpace->get_UniqueName(&bstrTemp);
   if(bstrTemp == bstrTuningSpaceString)
      break; //gefunden
   m_pITuningSpace.Release(); //COM Interface freigeben
}
if(!m_pITuningSpace) {
   ...; //Fehlerbehandlung
}
```

**Listing 2: Tuning Space für TT S2-3200 / SkyStar HD initialisieren**

```
HRESULT CreateDVBTuneRequest(
                        IDVBSTuningSpace* pIDVBSTuningSpace,
                        ModulationType& eModulation,
                        long lSATFrequency_kHz,
                        const Polarisation& ePolarization,
                        long lSymbolRate_ksym_per_s,
                        IDVBTuneRequest** ppIDVBTuneRequest)
{
   if(!pIDVBSTuningSpace)
      return E_POINTER;
   CComPtr<IDVBSLocator> pIDVBSLocator;
   HRESULT hr = pIDVBSLocator.CoCreateInstance(
                                           CLSID_DVBSLocator);
   if(FAILED(hr))
      return hr;

   hr = pIDVBSLocator->put_CarrierFrequency(
                                        lSATFrequency_kHz);
   if(FAILED(hr))
      return hr;
   hr = pIDVBSLocator->put_SignalPolarisation(ePolarization);
   if(FAILED(hr))
      return hr;
   hr = pIDVBSLocator->put_SymbolRate(
                                    lSymbolRate_ksym_per_s);
   if(FAILED(hr))
      return hr;
   hr = pIDVBSLocator->put_Modulation(eModulation);
   if(FAILED(hr))
      return hr;

   CComPtr<ITuneRequest> pITuneRequest;
   hr = pIDVBSTuningSpace->CreateTuneRequest(&pITuneRequest);
   if(FAILED(hr))
      return hr;
   hr = pITuneRequest->QueryInterface(
               IID_IDVBTuneRequest,
               reinterpret_cast<void**>(ppIDVBTuneRequest));
   if(FAILED(hr))
      return hr;

   return (*ppIDVBTuneRequest)->put_Locator(pIDVBSLocator);
}
```

**Listing 3: Erzeugung eines DVB-Tune-Requests**

**Digitale TV-Übertragungs-Standards in Europa (http://www.etsi.org/)**:
- *DVB-S/S2*: Digital Video Broadcasting - Satellite (*ETSI EN 302 307*, *ETSI TR 102 376* & *ETSI EN 300 468*)
- *DVB-C*: Digital Video Broadcasting - Cable
- *DVB-T*: Digital Video Broadcasting - Terrestrial
- *DVB-H*: Digital Video Broadcasting - Handheld

**SAT-Empfangsfrequenzen**:
- *Ku-Band*: *10,70 - 12,75 GHz*
  LOF zum Runtermischen auf L-Band: *9,75 GHz* (Low) und *10,60 GHz* (High)
  Umschalt-Frequenz (Wechsel der LOF): *11,725 GHz*
- *C-Band*: *3,6 - 4,2 GHz*
  LOF zum Runtermischen auf L-Band: *5,15 GHz*
- *L-Band*: *0,95 - 2,15 GHz*

**DVB-S/S2 Modulationsarten**:
- *DVB-S-QPSK*
  - kein BPSK-modulierter PLHEADER vorhanden
  - es sind keine Pilot-Blocks (36 Symbole mit unmoduliertem Träger) enthalten
  - Innere Vorwärtsfehlerkorrektur: *Convolutional Encoding*, einstellbare (innere) Effizienz: 1/2, 2/3, 3/4, 5/6, 7/8
  - Äußere Vorwärtsfehlerkorrektur: *Reed Solomon*, (äußere) Effizienz: 188/204
  - Demodulator-Chip von ST: *stv0299* (DVB-S) oder *stb0899* (DVB-S/S2)
  - Standard Nettobitraten: *33,79 MBit/s* (22000 kSymbols/s) und *38,015 MBit/s* (27500 kSymbols/s)
- *DVB-S2-QPSK* (sehr selten verwendet - bisher nur bei verschlüsselten Programmen)
  - BPSK-modulierter PLHEADER vorhanden
  - normalerweise sind Pilot-Blocks (36 Symbole mit unmoduliertem Träger) enthalten
  - Innere Vorwärtsfehlerkorrektur: *LDPC* (Low Density Parity Check)
  - Äußere Vorwärtsfehlerkorrektur: *BCH* (Bose-Chaudhuri-Hocquenghem)
  - Einstellbare Gesamt-Effizienz der Vorwärtsfehlerkorrektur: 1/2, 3/5, 2/3, 4/5, 3/4, 5/6, 7/8, 8/9, 9/10
  - Demodulator-Chip von ST: *stb0899* (DVB-S/S2)
- *DVB-S2-8PSK*
  - BPSK-modulierter PLHEADER vorhanden
  - normalerweise sind Pilot-Blocks (36 Symbole mit unmoduliertem Träger) enthalten
  - Innere Vorwärtsfehlerkorrektur: *LDPC* (Low Density Parity Check)
  - Äußere Vorwärtsfehlerkorrektur: *BCH* (Bose-Chaudhuri-Hocquenghem)
  - Einstellbare Gesamt-Effizienz der Vorwärtsfehlerkorrektur: 3/5, 2/3, 3/4, 5/6, 8/9, 9/10
  - Demodulator-Chip von ST: *stb0899* (DVB-S/S2)
  - Standard Nettobitrate: *42,5 MBit/s*

**LNB-Umschaltung**:
- Über *Gleich-/Wechselspannung*, die über das Antennenkabel an den LNB angelegt wird:
  *13V/=* → V, Low
  *18V/=* → H, Low
  *13V/22kHz* → V, High
  *18V/22kHz* → H, High
- Über bit-serielle *DiSEqC* Umschaltung, die über das Antennenkabel an den DiSEqC-Switch gesendet wird:
  *Position A, Option A* → LNB 1
  *Position B, Option A* → LNB 2
  *Position A, Option B* → LNB 3
  *Position B, Option B* → LNB 4
  Zur bit-seriellen Übertragung der kodierten Information wird *PWK* Modulation mit 22 kHz benutzt:
  *0,5 ms Puls + 1 ms Pause* → 0 (1/3 Puls)
  *1 ms Puls + 0,5 ms Pause* → 1 (2/3 Puls)

**DirectX9/BDA Filter-Dateien**:
- BDA Network Provider: *MSDvbNP.ax* [*BDANT.cap*]
- Tuner/Capture Proxy: *ksproxy.ax* [*DirectX.cap*]
- MPEG-2 Demultiplexer: *mpg2splt.ax* [*DirectX.cap*] (enthält auch AC3-Parser & MPEG-2-Splitter)
- TIF: *psisrndr.ax* [*BDANT.cap*]
  ACHTUNG! Die TIF Implementierung ist nicht sehr tolerant gegenüber korrupten (P)SI-Daten. Speziell bei einigen verschlüsselten DVB-S2 Transpondern führt dies zum Programmabsturz, da es wegen nicht-DVB-konformem Packetisieren der (P)SI Sections zu korrupten Daten und damit falschen Längenangaben kommt, wenn man DVB-konform dekodiert.

**Übersicht: SAT-TV - Die wichtigsten Parameter auf einen Blick**