國立交通大學

電機與控制工程學系

碩 士 論 文

MOST (Media Oriented System Transport)網路之影音伺服器設計

Design of Audio/Video Server under MOST (Media Oriented System Transport) Network

研 究 生：黃俊穎

指導教授：王啟旭 教授

中華民國九十六年九月

MOST (Media Oriented System Transport)網路之影音伺服器設計

Design of Audio/Video Server under MOST (Media Oriented System Transport) Network

研 究 生：黃俊穎　　Student：Jun-Ying Huang

指導教授：王啟旭 教授　　Advisor：Chi-Hsu Wang

國 立 交 通 大 學

電機與控制工程學系

碩 士 論 文

A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical Engineering and Computer Science

National Chiao-Tung University

In Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering

September 2007

Hsinchu, Taiwan, Republic of China

中華民國九十六年九月

MOST (Media Oriented System Transport)網路之影音伺服器設計

研究生：黃俊穎　　　　　指導教授：王啟旭 教授

國立交通大學電機與控制工程學系

摘　要

本論文主要在探討一種為汽車產業開發的多媒體網路技術--MOST (Media Oriented System Transport，多媒體導向系統傳輸)，並示範其多媒體應用。利用 OASIS SiliconSystems 公司(現為 SMSC 的子部門)所提供的 MOST 網路服務應用程式介面 (MOST NetServices API)，我們撰寫程式去控制在 MOST 網路上的多媒體設備，例如 DVD 播放器、視訊解碼器、收音機和聲音放大器。在應用中，我們將個人電腦主機用作網路的中控台，透過 MOST 網路介面卡 (MOST PCI Board)或光纖網路分析介面盒 (Optolyzer Interface Box)，來建立 MOST 網路與個人電腦主機之間的通訊。藉由各種多媒體應用之介紹，在文中將說明 MOST 網路服務應用程式介面的使用方式；並且在最後，我們設計和試驗了一段獨特的程序，來實現在 MOST 網路應用下的影音伺服器。

Design of Audio/Video Server under MOST (Media Oriented System Transport) Network

Student：Jun-Ying Huang Advisor：Chi-Hsu Wang

Department of Electrical and Control Engineering

National Chiao Tung University

ABSTRACT

The main purpose of this thesis is to explore and demonstrate the versatile multimedia applications under MOST (Media Oriented System Transport) network which is originally developed for the automotive industry. With the MOST NetServices API and an example application provided by OASIS SiliconSystems (which is now part of SMSC), we can build our programs to control multimedia devices for MOST, such the DVD Player, Video Decoder, Radio Tuner and Audio Amplifier are installed in our MOST network. Besides, both the MOST PCI Board and the Optolyzer Interface Box are used for establishing the communication between the MOST network and a PC which serves as a control central for the MOST network. The details of understanding the NetServices API are explained through different multimedia applications. In addition to the ordinary MOST applications, a new Audio/Video server under MOST network is also designed.

ACKNOWLEDGEMENT

My deepest gratitude goes first and foremost to my advisor, Prof. Chi-Hsu Wang, for his constant encouragement and guidance. Without his illuminating instruction, this thesis could not have reached its present form. Secondly, I would like to thank my senior, Jian-Xun Wu, and Support Team from SMSC Automotive Infotainment Systems, for their instructions in my research. Besides, I am grateful to everyone in our ECL Lab. With their companion, I always have a good time during these days in NCTU. Finally I would like to express my sincere gratitude to my family for the loving considerations and great confidence in me.

TABLE OF CONTENTS

LIST OF TABLES

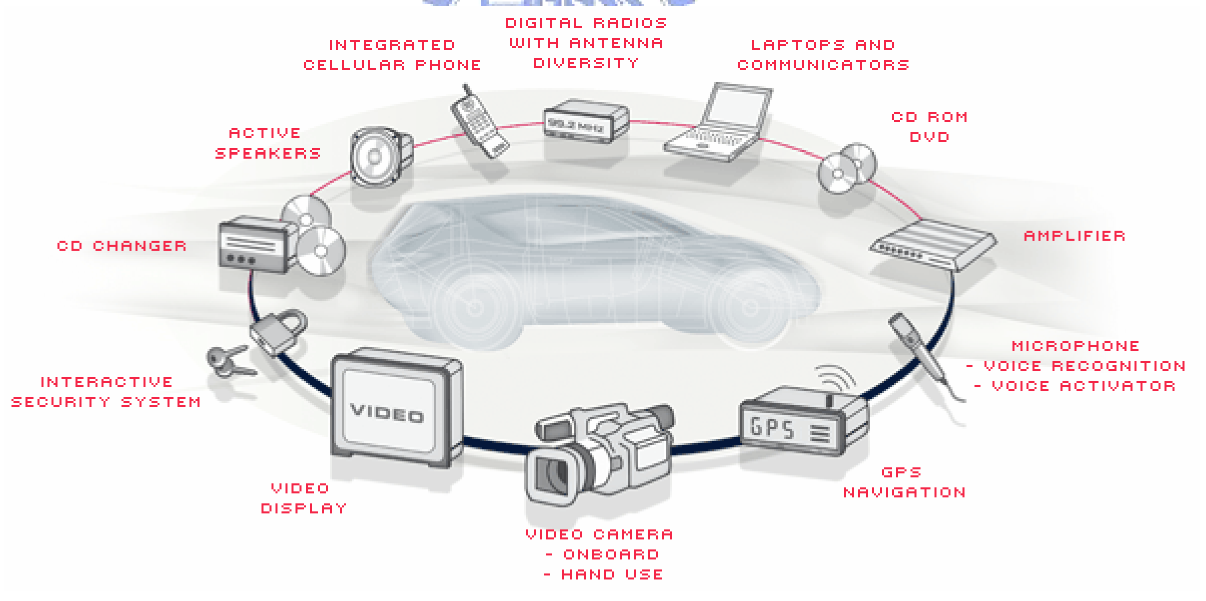
LIST OF FIGURES

CHAPTER 1

Introduction

In the master thesis of my senior, Jian-Xun Wu [1], several multimedia applications of MOST (Media Oriented System Transport) are developed with the MOST NetServices API [2] (Application Programming Interface). After his graduation, I took over the experiments on MOST in our lab. Continuing with Jian-Xun Wu's work, in this master thesis, the control program for multimedia applications of MOST will be explored and enhanced, which contributes the design of Audio/Video server under MOST network.

1.1 MOST Technology [3] [4]



INTEGRATED CELLULAR PHONE
DIGITAL RADIOS WITH ANTENNA DIVERSITY
LAPTOPS AND COMMUNICATORS
CD ROM DVD
ACTIVE SPEAKERS
CD CHANGER
AMPLIFIER
MICROPHONE - VOICE RECOGNITION - VOICE ACTIVATOR
INTERACTIVE SECURITY SYSTEM
VIDEO
GPS
VIDEO DISPLAY
GPS NAVIGATION
VIDEO CAMERA - ONBOARD - HAND USE

Figure 1.1 Multimedia and Communication peripherals

MOST (Media Oriented Systems Transport) is a multimedia network developed for the automotive industry. In-vehicle consumer devices such as DVD players, MP3 players, GPS systems, car phones and Bluetooth devices can be linked together to work as a single system on the MOST network. With standardized connectors and media e.g., low cost Plastic Optical

Fiber (POF) or Unshielded Twisted Pair (UTP), high bandwidths up to 25 Mbps and 50 Mbps are supported.

MOST technology not only includes the physical connection between devices, but also defines properties and methods for devices to interact with each other. The software interfaces allow applications running on different devices to communicate and exchange information. A transport mechanism that sets up a link for streaming data between devices is also defined.

1.2 Scope of work

The MOST devices in our lab, including the RadioTuner4MOST, Amplifier4MOST, DVDPlayer4MOST, VideoDecoder4MOST, the MOST PCI Board and the Optolyzer Interface Box, from SMSC [4], are connected in a ring topology as a multimedia network system. Via the MOST PCI Board, a PC is capable of communicating with the MOST network for management. Moreover, by means of the NetServices API, we can develop our control programs on the Windows platform to implement different multimedia applications of MOST.

Compared with Jian-Xun Wu's work, in this thesis, the VideoDecoder4MOST will be made functional for our applications and more functions will be integrated in the control program. With a small tool combined with the new PCI Board drivers, it is possible for us to stream a media file from a PC to MOST, as an Audio/Video server.

1.3 Structure of the Thesis

Chapter 2 and Chapter 3 comprise the basics of MOST. Chapter 2 is an overview of MOST specification while Chapter 3 introduces MOST NetServices and the API.

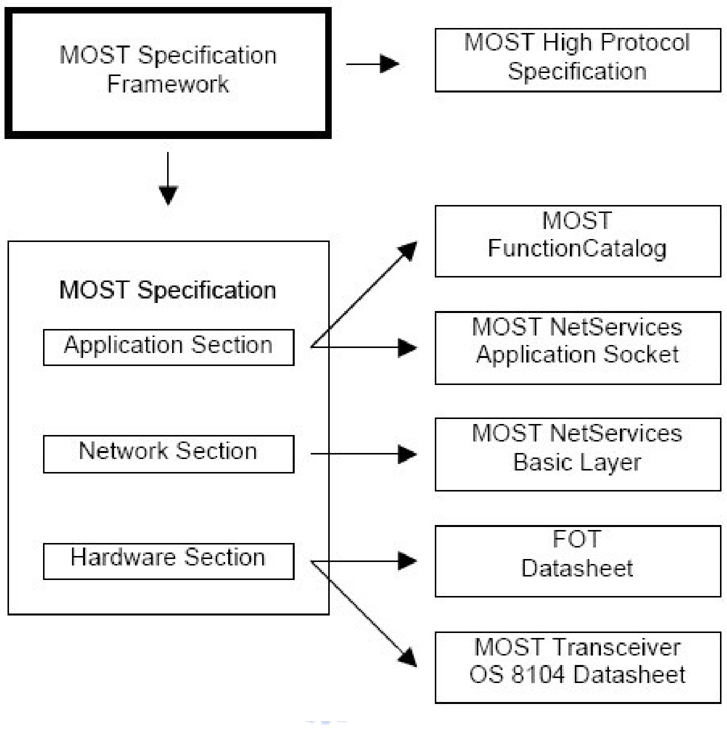
In Chapter 4 before our multimedia applications of MOST are described, introductions to the hardware and software utilities are given.

In Chapter 5, the application, A/V server under MOST network, is discussed apart from those of Chapter 4.

Finally, the conclusion of this thesis is given in Chapter 6, and some possible future works for our lab are suggested, too.

CHAPTER 2
MOST Specification

Figure 2.1 is the structural overview of the MOST specification documentation:



MOST Specification Framework
MOST High Protocol Specification
MOST FunctionCatalog
MOST Specification
Application Section
MOST NetServices Application Socket
Network Section
MOST NetServices Basic Layer
Hardware Section
FOT Datasheet
MOST Transceiver OS 8104 Datasheet

Figure 2.1 Structural overview of the documentation [5]

In this chapter, based on the documents “MOST Specification Framework” [6] and “MOST Specification” [5], we wish to give an introductory overview of the MOST System and its abilities. A MOST System often indicates a MOST network which is a group of MOST Devices linked together by some way.

A MOST System can be described by three definition areas which are MOST Interconnect, MOST System Services, and MOST Devices. Except MOST System Services will be introduced separately in Chapter 3, the concepts of MOST Interconnect and MOST Devices should comprise the main content of this chapter.

At first, we start with MOST Topology, the way of physical interconnection between nodes. Then we get in the section of Data Transport, to see what is transported on the MOST network.

The third part of this chapter is Logical Device Model in which the logical components of a MOST device are described along with their functionalities. This section covers the contents from the application level to the lower physical layer.

For developing our applications of MOST, it is necessary to follow the MOST protocols; therefore, before moving on to the next chapter and exploring the Network Services, the protocols are introduced in the last section.

2.1 MOST Topology [6]

2.1.1 Point to Point Link

There are two basic system architectures supported by MOST Technology. The first is a one-way, point-to-point link requiring a simple transmitter on one side and a receiver on the other. This is an extremely cost effective, one-way synchronous digital connection for applications such as connecting a digital audio source to active speakers, or a digital video source to a monitor. However, this basic configuration can easily be extended to more complex structures, such as branches or bi-directional links.

2.1.2 Ring Topology

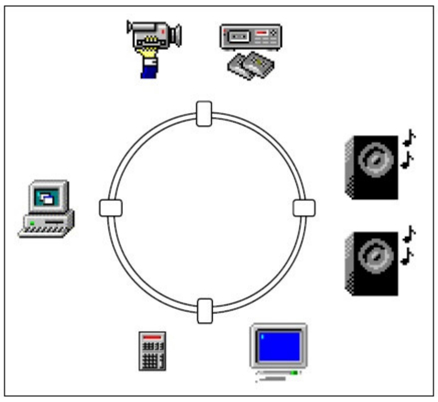


Figure 2.2 Ring Topology

The second basic system architecture is a network with a ring topology. This network can have as few as two nodes, effectively a full duplex point-to-point connection, or as many as 64 nodes, as shown in Figure 2.2.

There are many advantages of a ring topology, which include the following:

- Minimum use of physical layer media, reducing system cost and weight.
- Low overall cost and constant cost per node since there is no overhead cost in the form of a hub, switch, or other shared network resource.
- Ease of expansion and no change in the basic architecture (such as the wiring infrastructure) is required.
- Availability of all source data (e.g. digitized audio) at each device.

One major disadvantage of a traditional ring topology is that the failure of one node can cause the entire network to fail. However, it can be largely overcome in a MOST network. Many failure mechanisms of a node such as power loss, or loss of lock will result in the transceiver going into bypass mode such that the rest of the network is unaffected.

Variations of the ring topology are possible as well, which will be introduced in next two sections. Moreover, if system speed, reliability and minimization of downtime are paramount, a dual ring topology can be used.

2.1.2.1 Rings Incorporating Splitters

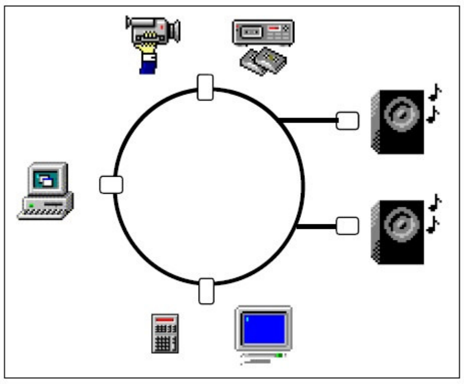


Figure 2.3 Rings Incorporating Splitters

As shown in Figure 2.3, optical splitters can be incorporated into a POF (Plastic Optical Fiber) Ring creating branches connected to receive-only nodes such as display or active speaker devices. One major advantage of this implementation is reduced cost, since no transmitter is required.

2.1.2.2 Star of Rings

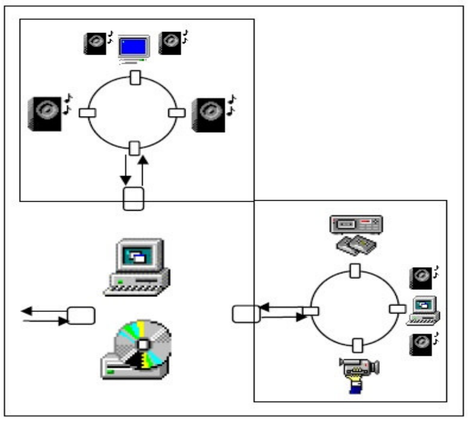


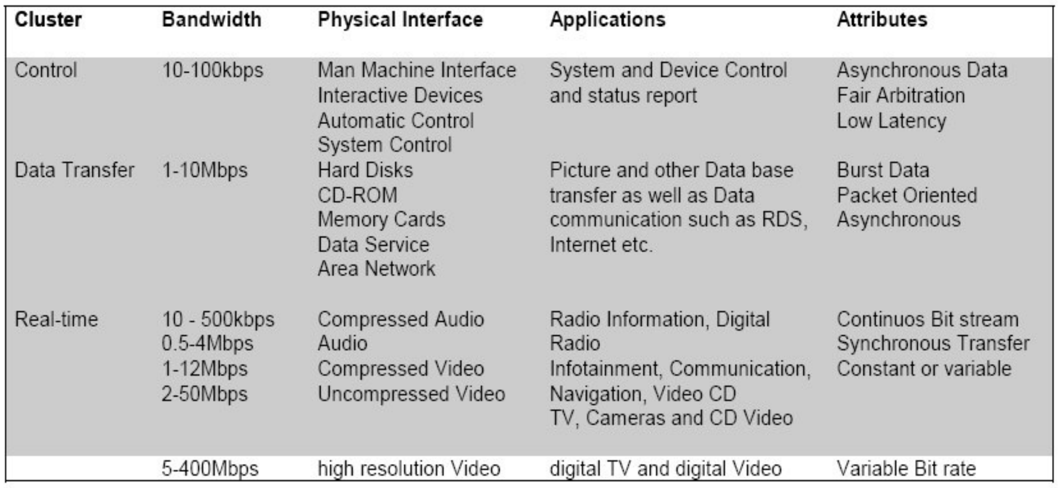
Figure 2.4 Star of Rings

Figure 2.4 shows a combination of Star and Ring topologies, and this may be the best choice for large networks. In structures as required for home networking such a configuration with a central server/router (i.e. set-top boxes) can have multiple branches, with each of them configured as a Ring structure. This offers the most flexible way of implementing complex structures with easy fault detection and high bandwidth.

2.2 Data Transport

Data is transferred in a continuous bi-phase encoded bit stream yielding more than a 24.8Mbps data rate, and the sample frequency in a MOST system can be chosen in a range between 30 kHz and 50 kHz.

Table 2.1 Speed requirements of different applications [6]



Cluster	Bandwidth	Physical Interface	Applications	Attributes
Control	10-100kbps	Man Machine Interface Interactive Devices Automatic Control System Control	System and Device Control and status report	Asynchronous Data Fair Arbitration Low Latency
Data Transfer	1-10Mbps	Hard Disks CD-ROM Memory Cards Data Service Area Network	Picture and other Data base transfer as well as Data communication such as RDS, Internet etc.	Burst Data Packet Oriented Asynchronous
Real-time	10 - 500kbps 0.5-4Mbps 1-12Mbps 2-50Mbps	Compressed Audio Audio Compressed Video Uncompressed Video	Radio Information, Digital Radio Infotainment, Communication, Navigation, Video CD TV, Cameras and CD Video	Continuos Bit stream Synchronous Transfer Constant or variable
	5-400Mbps	high resolution Video	digital TV and digital Video	Variable Bit rate

Before discussing with the transferred data types and the frame structure of MOST, let's take a look at Table 2.1, an overview of bandwidths needed by different applications, and how MOST fits into this picture:

2.2.1 Data Types [6]

Different types of information can be transmitted over MOST and the frame structure separates the data into three different sections: Control data, asynchronous packet data and synchronous stream data.

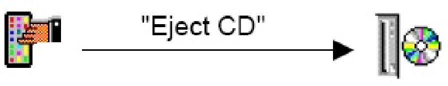


Figure 2.5 Control data

• **Control data transfer** is for device specific transfers and system management. As shown in Figure 2.5, control data “Eject CD” is sent to the CD player to eject the CD. This kind of data is transported in parallel to the real-time and asynchronous data.

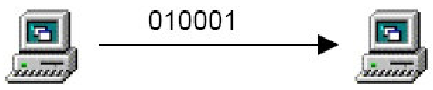


Figure 2.6 Packet data

• **Bulk data / burst data transfer** in asynchronous packets with variable bandwidth requirements. As shown in Figure 2.6, it is often used for TCP/IP packets between computers.

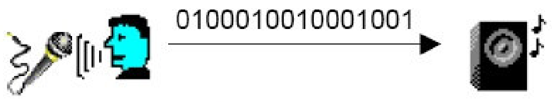


Figure 2.7 Synchronous data

• **Real-time data transfer** which requires a guaranteed bandwidth to maintain data quality. As shown in Figure 2.7, audio stream from microphone to the speaker is such kind of data. Indeed, all nodes on the MOST network have access to it.

2.2.2 Frame Structure [5]

2.2.2.1 Blocks

A block consists of 16 frames with 512 bits (64 bytes) each, running at the system sample rate as frame rate (typical 44.1 kHz). Organization of data transfer in blocks of frames is required for network management and control data transport tasks. A control message has a fixed size of 32 bytes and is time multiplexed over 16 frames (1 block), each having 2 control data bytes.

2.2.2.2 Frames

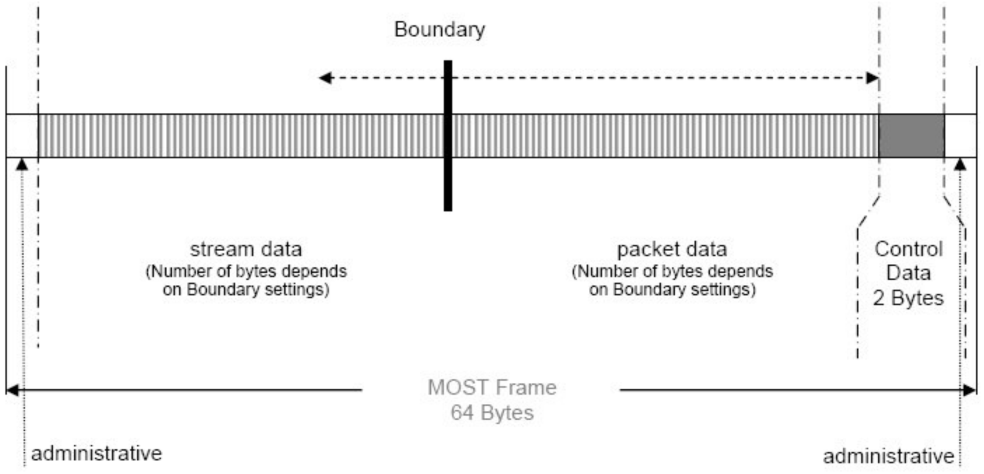


Figure 2.8 MOST Frame

Table 2.2 Structure of MOST Frame

Byte Number	Task
0	Administrative ◦ Preamble (bits 0-3) ◦ 4 bits Boundary Descriptor (bits 4-7)
1 - 60	60 data bytes
61 - 62	2 data bytes for Control Messages
63	Administrative ◦ Frame control and status bits ◦ Parity bit (last bit)

One MOST25 frame consists of 512 bits (64 bytes). The first byte is used for administrative purposes. The next 60 bytes are used for stream and packet data transfer, where the Boundary is defined in 4 byte steps. All Stream data bytes are transmitted before any Packet data bytes occur. The next two bytes of each frame are reserved for Control data and the last byte is another administrative byte. As shown in Figure 2.8 and Table 2.2.

• **Preamble**

The preambles are used for synchronization of MOST nodes. Different mechanisms are used for Slave and Master nodes. Please refer to [5] for more details.

• **Boundary Descriptor**

The value represents the number of 4 byte blocks (quadlets) of data used for synchronous data. It is used to determine the boundary between the synchronous and asynchronous data areas in the frame, on a 4 byte resolution. The Boundary can only have values between 6 and 15. Therefore, at least 24 bytes are available for Stream data transfer.

• **MOST System Control Bits**

All other bits within the frame are for management purposes on the network level. While the preamble provides synchronization and clock regeneration, the parity bit indicates reliable data content and is used for error detection and phase lock loop operation.

2.2.3 Transparent Channel

For detailed information about control data channel, asynchronous packet data channel, synchronous data channel, and the structure of a frame in different areas, please refer to [5].

Some applications, however, may provide asynchronous data interfaces such as RS 232 and as a consequence require transparent asynchronous data transfer capability. In this case the asynchronous data lines can be oversampled at the synchronous channels and routed over the network to any other source data port.

2.3 Logical Device Model [5]

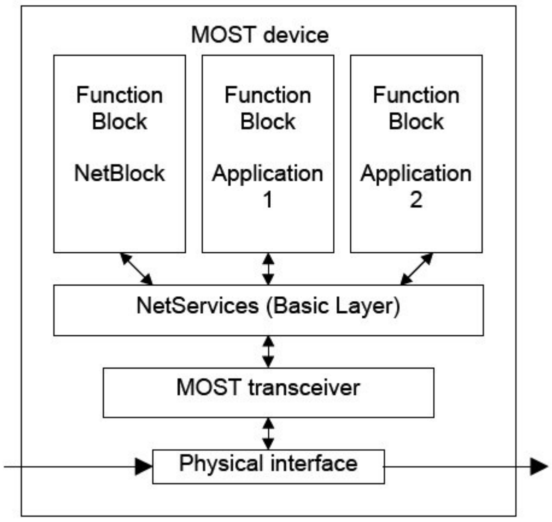

MOST device
Function Block NetBlock
Function Block Application 1
Function Block Application 2
NetServices (Basic Layer)
MOST transceiver
Physical interface

Figure 2.9 Model of a MOST Device

A MOST device is a physical unit which can be connected to a MOST network via a MOST Network Interface Controller (NIC). From the logical point of view, every MOST device consists of four kinds of components which are Function Blocks, NetServices, a MOST transceiver and a Physical interface, as shown in Figure 2.9.

2.3.1 Function Block

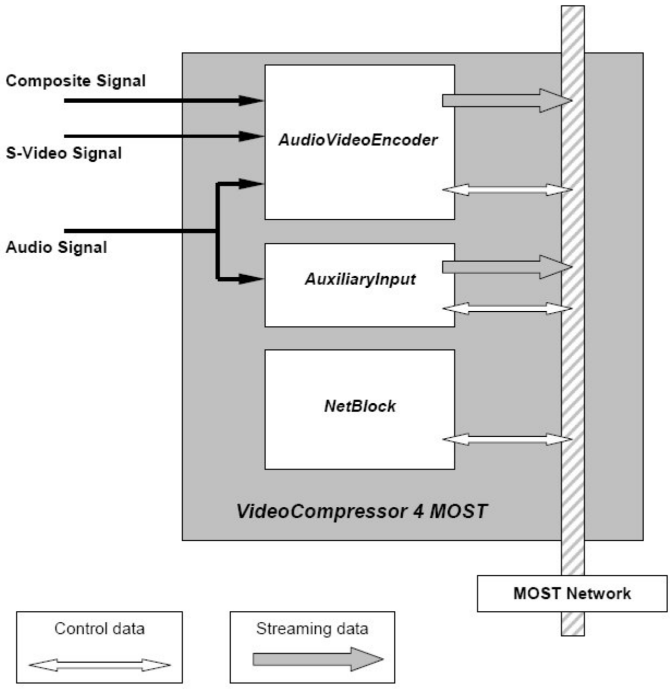


Figure 2.10 VideoCompressor4MOST Block Diagram [7]

On the application level, a MOST device contains multiple components which are called function blocks (FBlocks) in which functions are grouped together with respect to their contents. For example, Figure 2.10 is the block diagram of the VideoCompressor4MOST. There are three function blocks available: AudioVideoEncoder, AuxiliaryInput, and NetBlock.

- **Controller, Slave, HMI**

  Interaction with an FBlock requires two partners which are distributed over the MOST network: The Controller and the Slave.

  The FBlock functionality resides in the Slave, and FBlocks using functions in Slaves are called Controllers. The Controller sends commands to a Slave and in return receives reports from the Slave. The Slave executes the commands issued by a Controller and sends status reports to the Controller. Controllers which have an interface to the user are called Human Machine Interfaces (HMIs).

2.3.1.1 NetBlock

The NetBlock is mandatory for every MOST device. Unlike the other function blocks which represent applications, the NetBlock controls MOST network related functions and provides access to general information for MOST network administration.

2.3.1.2 Functions

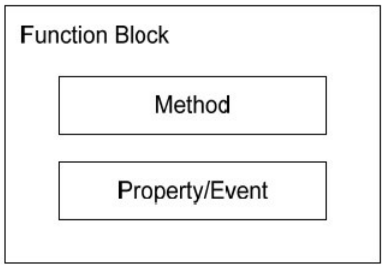


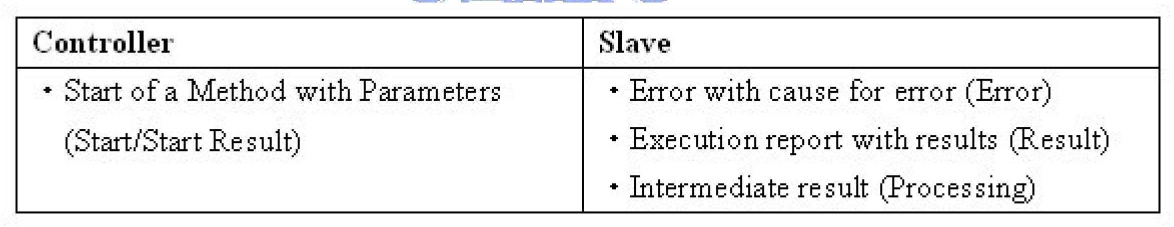
Figure 2.11 Class of Functions

Functions are accessible from outside the function block via the Function Interface (FI) and can be further divided into three groups (depending on what kind of operation they perform):

- **Methods** are defined as functions which can be started and which lead to the result after a certain period of time. Generally they are triggered only once and are used to control the FBlocks. After a method is called, depending on what kind of operation type we use, the error or report message should be sent to the initiator.

  For example, after allocating audio channels for the DVDPlayer4MOST, the results of process may be the index numbers of allocated audio channels. The following kinds of messages are exchanged via the bus for executing methods:

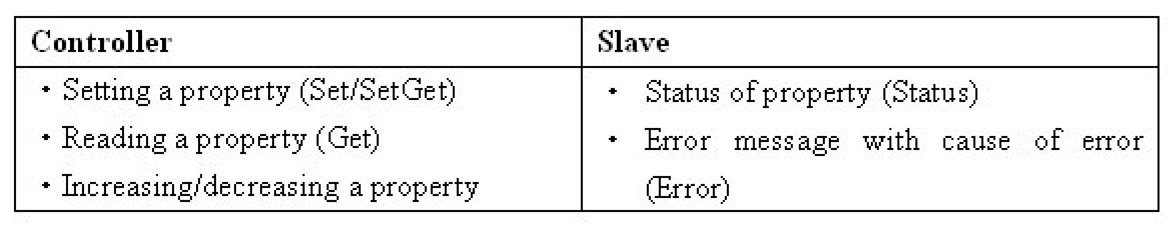
Table 2.3 Messages for Methods



Controller	Slave
• Start of a Method with Parameters (Start/Start Result)	• Error with cause for error (Error) • Execution report with results (Result) • Intermediate result (Processing)

- **Properties** are used to change or get the state of the device. For example, to change the status of the DVDPlayer4MOST from "Play" to "Stop". Operations allowed for each property are specified. For changing and reading of properties, the following types of messages are exchanged via the bus:

Table 2.4 Messages for Properties



Controller	Slave
• Setting a property (Set/SetGet) • Reading a property (Get) • Increasing/decreasing a property	• Status of property (Status) • Error message with cause of error (Error)

- **Events** are basically the same as properties; the only distinction is that events occur without any external request.

  To display the values of properties which may change without external influence, such as the current time of a CD player, a cyclical reading would be required. Without explicit requests, events which occur in a controlled FBlock initiate the sending of a status report (Notification) about the changes in properties. Thus, communication with FBlocks can be reduced.

2.3.1.3 Function Interfaces

A function interface (FI) represents the interface between a function in a FBlock, and its usage in another FBlock.

To communicate with a function, a Controller or an HMI needs information about the available parameters, their limits, and the allowed operations. In general, this information is available in the control device, and is encoded in the control program.

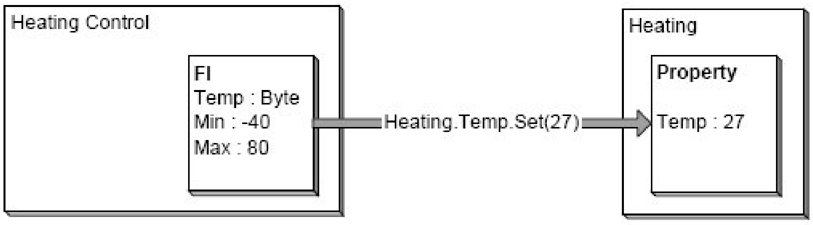


Figure 2.12 Example for a Function Interface

Take Figure 2.12 for example, the FI contains information about the data type of the function and about minimum and maximum value. In real implementations, a FI contains more information.

2.3.2 NetServices

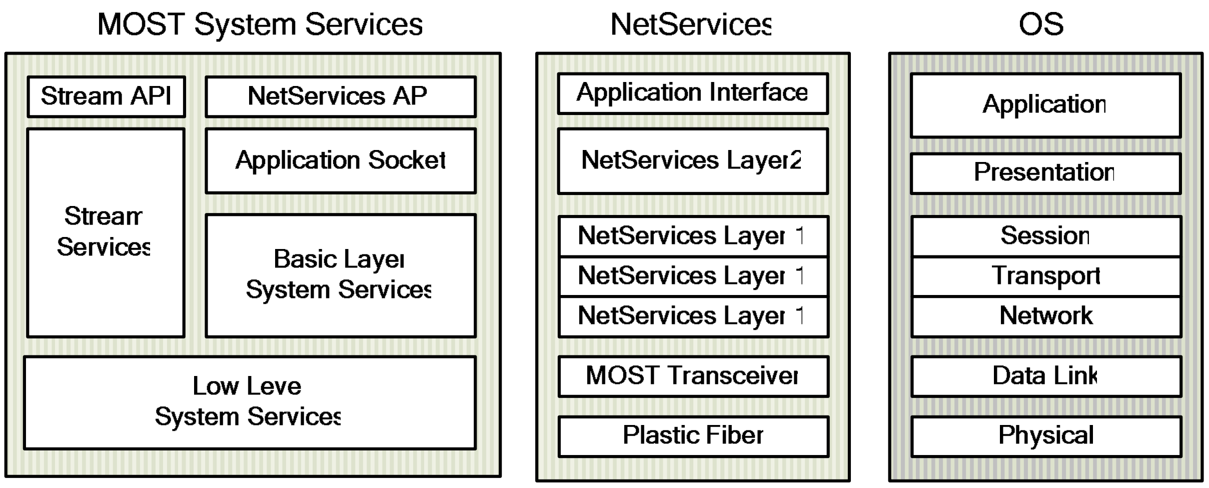


Figure 2.13 Mapping of MOST-OSI [8]

MOST NetServices is organized into two layers where Layer 1 is the Basic Layer System Services and Layer 2 is the Application Socket. A mapping of the MOST system towards the OSI reference model is done in Figure 2.13. NetServices provides services to simplify the handling of the MOST transceiver and is an intermediate layer between the MOST transceiver and MOST FBlocks.

In the next chapter, we will have more descriptions of NetServices and its API (Application Programming Interface).

2.3.3 MOST Transceiver and its Internal Services

In addition to converting to and from the format used on the MOST network, the MOST Transceiver OS8104 of Oasis SiliconSystems provides extensive tools for operating the MOST bus.

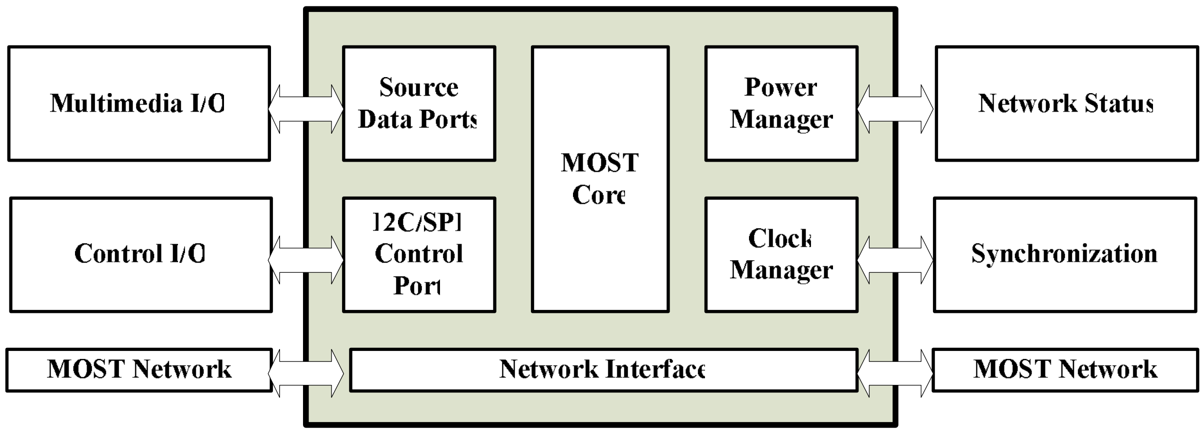


Figure 2.14 MOST Transceiver Block Diagram [8]

As shown in Figure 2.14, the Transceiver has several different interfaces. It can send and retrieve information via the Control Port, the Source Data Port or via the Network Interface (optical port). The Source Data Ports are typically connected to multimedia sources and sinks which handle audio and video streams. The optical port has two pins, the receive (RX) pin and the transmit (TX) pin. They are connected to the FOT (Fiber Optical Transceiver) which can transform between electrical and optical signals.

It is also possible to get the network status out of the Power Manager and to synchronize to other systems using the Clock Manager. In this section, its internal services will be introduced.

2.3.3.1 Bypass

- **Electrical Bypass (All Bypass)**

  Each device has its own bypass function. If a bypass is activated, all signals received at the input interface are transferred directly to the output interface. In this state, the device is passive and from the MOST bus point of view, it is "invisible".

  .

- **Source Data Bypass**

  In order to put data on the MOST network, the source data bypass should be opened (deactivated) in the device. As a result, the data is no longer passed through the chip without being processed, but can be routed from a source data port to the bus.

2.3.3.2 Timing Master

A MOST system consists of up to 64 nodes with identical MOST transceivers. In order to establish communication, one of those devices must become a Timing Master. The main task is to generate and transport the system clock, the blocks and the frames. All other devices (slaves) are synchronized with the Frame Generator by using internal PLL (Phase-Locked Loop).

2.3.3.3 Addressing

The MOST Transceiver supports four different ways of addressing:

- **Node Position in the ring -** Automatic generating of the position during the locking procedure.
- **Unique Node Address (2 bytes) -** An application can set unique address to a certain node.
- **Group Address (1 byte)** - Devices with the same address numbers create a group. The group addresses can be used to control devices with the same features (e.g. speakers).
- **Broadcast -** The broadcast address is a special group address. When used, the message is received by all nodes in the ring. During the broadcast message a control channel is locked for all other communication messages, until the acknowledgement from last device for broadcast is received.

In table 2.5, the different ways of addressing are mapped into the address area of a MOST Transceiver:

Table 2.5 Addressing modes vs. Address range

Address range	Mode
0x0001...0x02FF 0x0500…0xFFFF	Normal addressing (Point to point) based on unique node address
0x0400...0x04FF	Node position addressing: Address = 0x400 + Node position of target node
0x0300...0x03C7 0c03C9...0x03FF	Group addressed: Address = 0x300 + Address of desired group Group address 0xC8 reserved for broadcast
0x03C8	Broadcast addressing

2.3.3.4 Channel administration

The following internal services are implemented in the MOST network to provide functional data transfer and to allocate necessary timeslots for synchronous and asynchronous data.

- **System Startup**

  When the network is established every device receives a unique number starting with Timing Master 0x00 and then incremented by one for each next device. At the end of this procedure, the final value is distributed over the network (i.e. every connected device knows how many devices are connected to the bus).

- **Delay recognition**

  Each MOST transceiver has information about the source data delay. By taking into consideration the fact that each device can operate in passive or active mode (bypass is open or close), every active device creates two frames of delay.

- **Detection of unused channels**

  The timing master device can establish if there are some unused channels (i.e. allocated channels without any data transfer). By using resource allocation, the table re-allocation of unused channels can be recommended.

- **Automatic channel allocation**

  To allocate a channel each device has to send a request to the timing master device. As a reply, information about available channels is sent.

2.3.3.5 Power Management

The MOST Transceiver has three power states:

- **Normal operation mode**

  In normal operation mode, all sections of the chip are running and the chip is fully accessible.

- **Low power mode**

  Low power mode is used to lower the power consumption of devices that are not in use at the moment. All sections of the chip that handle source data are switched off, and source data bypass is active. The chip is visible from the network's point of view, but no communication is possible, except messages with a wakeup-preamble from the Timing Master.

- **Zero power mode**

  In zero power mode, all sections of the chip are deactivated, except a small wakeup logic. The wakeup logic activates the receiving FOT unit in regular intervals and scans for modulated light. If there is modulated light, the chip wakes up.

2.3.4 Physical Interface [6]

The physical interface provides mechanical connection between the Fiber Optical Transceiver (FOT) units and the Plastic Optical Fiber (POF). In addition to that, it connects the FOT units electrically with the MOST transceiver. The Physical Layer is optimized for the use of POF as the transfer medium, but copper cable (e.g. UTP, Coax) is permissible too. In fact, the UTP (Unshielded Twisted-Pair) cable is used in the new MOST50 ePHY (electrical physical layer) as transmission media providing doubled bandwidth for new video-intensive applications.

2.4 Protocols [5]

2.4.1 Structure of MOST Protocols

On the application level, the functions are addressed independently of the devices they belong to. Furthermore, the particular functions are unified into the function blocks according to their content. Hence, a function is addressed in a function block. The MOST protocols on the application layer have the following structure:

DeviceID . FBlockID . InstID . FktID . OPType . Length (Data)

2.4.1.1 DeviceID

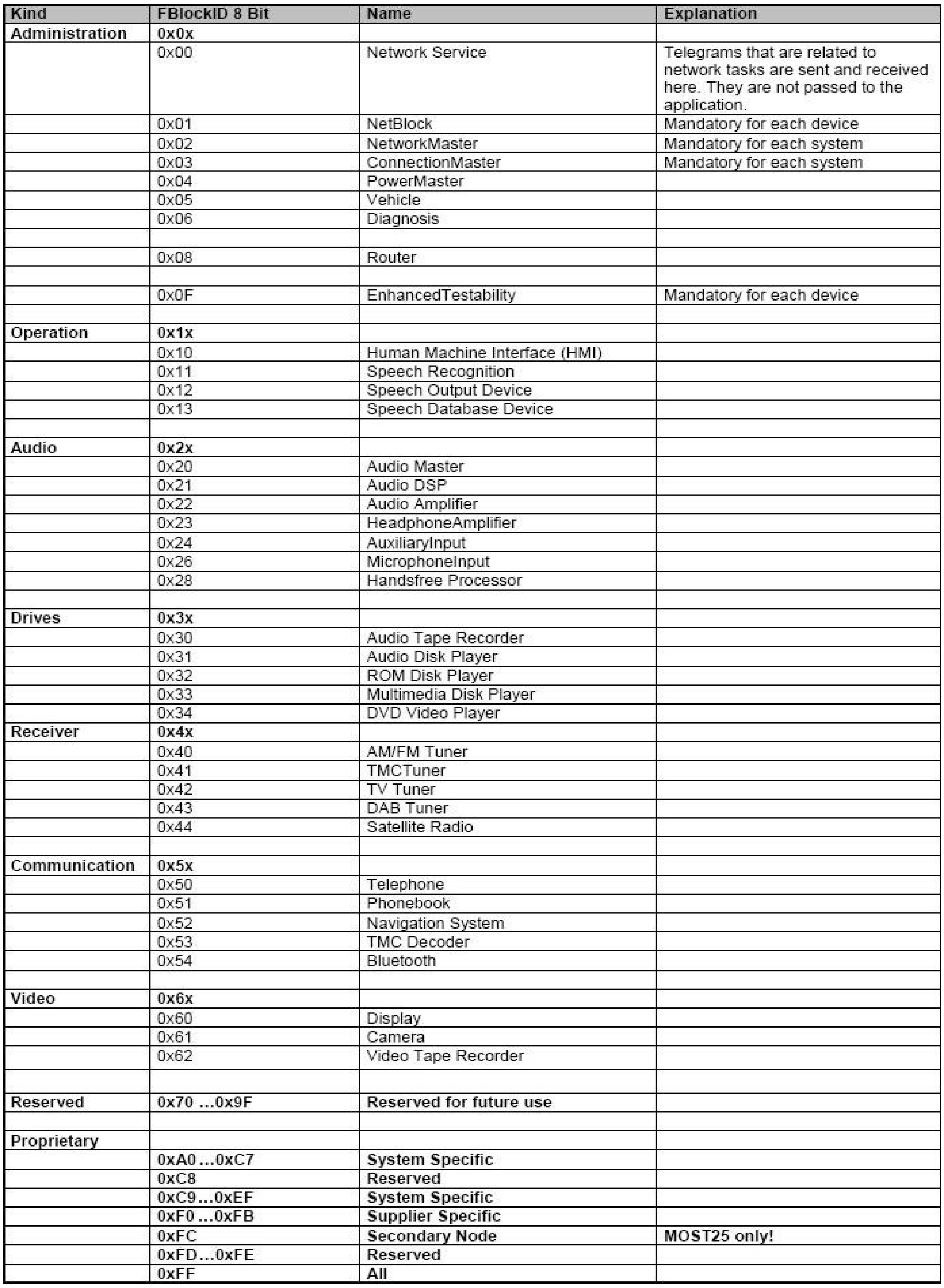
The DeviceID with a length of 16 bits, stands for a physical device or a group of devices (Group Address), and is network specific. It represents the logical node address of either the sender or the receiver, depending on the case. A group address or a broadcast address (0x03C8) could be used for the target too. In case the sender does not know the receiver's address, the DeviceID is set to 0xFFFF and will be corrected by the NetServices of the sender.

2.4.1.2 FBlockID

It specifies particular FBlock in the device. Every function block with a special FBlockID must contain certain specific functions other than the mandatory functions. Two kinds of proprietary FBlockIDs are defined: System specific and Supplier specific. System specific type can be used by any carmaker and is predefined, where the Supplier specific type is used by OEMs (Original Equipment Manufacturer) for development purpose.

The following table shows an (incomplete) collection of FBlockIDs:

Table 2.6 FBlockIDs



Kind	FBlockID 8 Bit	Name	Explanation
Administration	**0x0x**		
	0x00	Network Service	Telegrams that are related to network tasks are sent and received here. They are not passed to the application.
	0x01	NetBlock	Mandatory for each device
	0x02	NetworkMaster	Mandatory for each system
	0x03	ConnectionMaster	Mandatory for each system
	0x04	PowerMaster	
	0x05	Vehicle	
	0x06	Diagnosis	
	0x08	Router	
	0x0F	EnhancedTestability	Mandatory for each device
Operation	**0x1x**		
	0x10	Human Machine Interface (HMI)	
	0x11	Speech Recognition	
	0x12	Speech Output Device	
	0x13	Speech Database Device	
Audio	**0x2x**		
	0x20	Audio Master	
	0x21	Audio DSP	
	0x22	Audio Amplifier	
	0x23	HeadphoneAmplifier	
	0x24	AuxiliaryInput	
	0x26	MicrophoneInput	
	0x28	Handsfree Processor	
Drives	**0x3x**		
	0x30	Audio Tape Recorder	
	0x31	Audio Disk Player	
	0x32	ROM Disk Player	
	0x33	Multimedia Disk Player	
	0x34	DVD Video Player	
Receiver	**0x4x**		
	0x40	AM/FM Tuner	
	0x41	TMCTuner	
	0x42	TV Tuner	
	0x43	DAB Tuner	
	0x44	Satellite Radio	
Communication	**0x5x**		
	0x50	Telephone	
	0x51	Phonebook	
	0x52	Navigation System	
	0x53	TMC Decoder	
	0x54	Bluetooth	
Video	**0x6x**		
	0x60	Display	
	0x61	Camera	
	0x62	Video Tape Recorder	
Reserved	**0x70 …0x9F**	**Reserved for future use**	
Proprietary			
	0xA0…0xC7	**System Specific**	
	0xC8	**Reserved**	
	0xC9…0xEF	**System Specific**	
	0xF0…0xFB	**Supplier Specific**	
	0xFC	**Secondary Node**	**MOST25 only!**
	0xFD…0xFE	**Reserved**	
	0xFF	**All**	

2.4.1.3 InstID

It is an identifier of instance of function block. If the FBlockID is not unique within the system this identifier specifies the function block. FBlockID and InstID create the functional address which has to be unique.

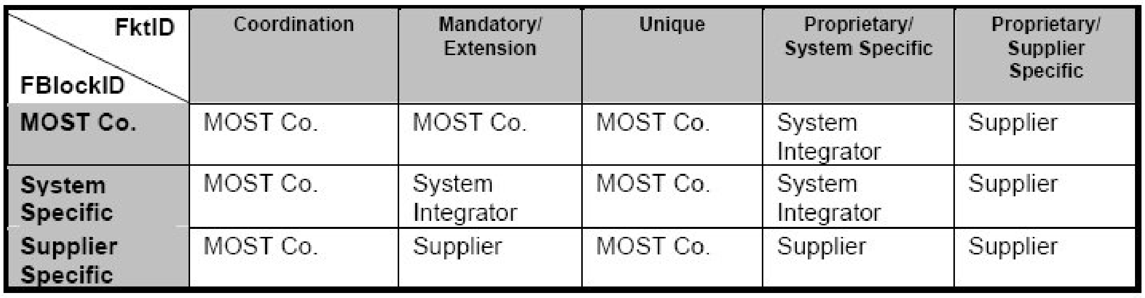
- **0x01 -** By default, every function block has instance ID 0x01. In case there are several FBlocks of the same kind within one MOST device, the default number (within the device) starts at 1.
- **0x00 -** Don't care. The device dispatches the message to one specific FBlock in the device.
- **0xFF -** Broadcast (within a device). The message is dispatched to all instances of the matching FBlock.

2.4.1.4 FktID

The FktID stands for a function. This means a function unit (Object) within a device which provides operations (e.g. "Eject CD") can be called via the network. The FktID is encoded in 12 bits on network level and is extended to 2 bytes on application level. The address range of FktIDs is subdivided into the following sections: Coordination, Mandatory, Extensions, Unique, System Specific and Supplier Specific.

The following table regulates who is authorized to use certain FBlockID/FktID combinations.

Table 2.7 Responsibilities for FBlockID and FktID ranges

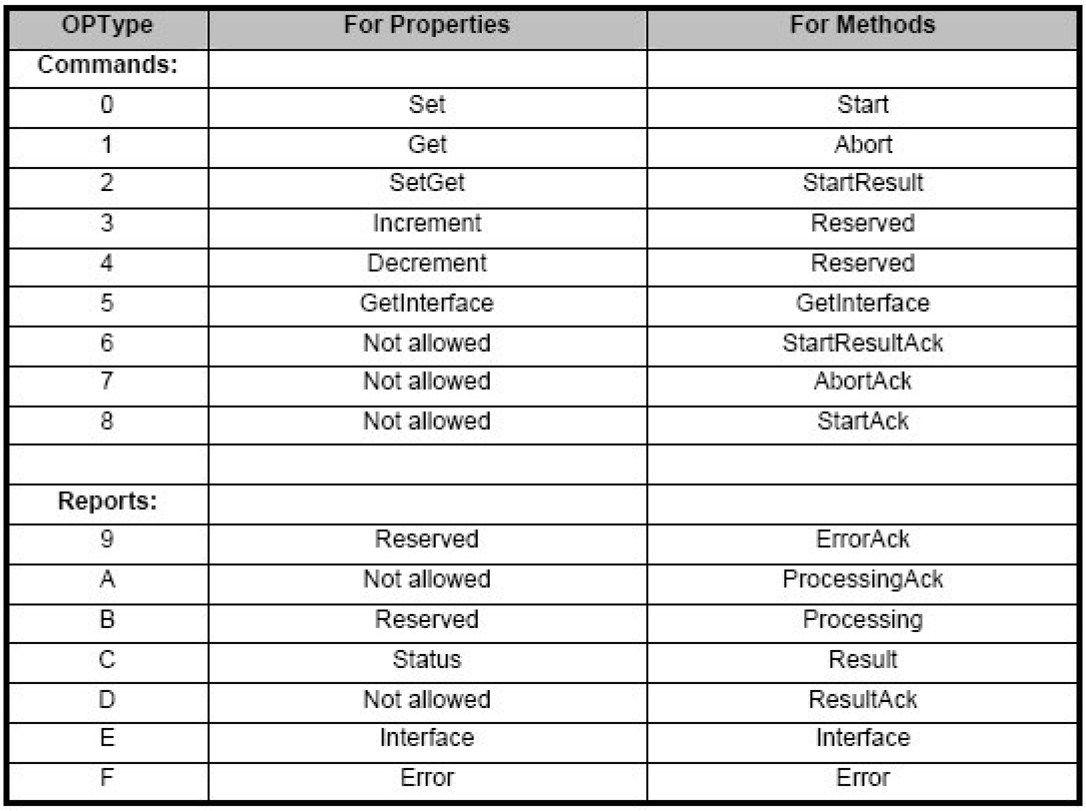


FktID FBlockID	Coordination	Mandatory/ Extension	Unique	Proprietary/ System Specific	Proprietary/ Supplier Specific
MOST Co.	MOST Co.	MOST Co.	MOST Co.	System Integrator	Supplier
System Specific	MOST Co.	System Integrator	MOST Co.	System Integrator	Supplier
Supplier Specific	MOST Co.	Supplier	MOST Co.	Supplier	Supplier

2.4.1.5 OPType

The OPType indicates which operation must be applied to the property or method specified in FktID:

Table 2.8 OPTypes for properties and methods



OPType	For Properties	For Methods
Commands:		
0	Set	Start
1	Get	Abort
2	SetGet	StartResult
3	Increment	Reserved
4	Decrement	Reserved
5	GetInterface	GetInterface
6	Not allowed	StartResultAck
7	Not allowed	AbortAck
8	Not allowed	StartAck
Reports:		
9	Reserved	ErrorAck
A	Not allowed	ProcessingAck
B	Reserved	Processing
C	Status	Result
D	Not allowed	ResultAck
E	Interface	Interface
F	Error	Error

- **Error** is reported only to the Controller that has sent the instruction. On Error, an error code is reported in the data field (Data[0]). Additional information for error codes can be found in [5].

2.4.1.6 Length

Length is encoded in 16 bits which specifies the length of the data field in bytes. This parameter is not transferred through the control channel, but is computed in the receiving node.

2.4.1.7 Data

In principle, the data field of a message in the application layer (also referred to as Application Message) may have any length up to 65535 (specified by the 16-bits Length) bytes. In a telegram on the control channel of the MOST bus, the maximum length is 12 bytes. Longer protocols must be segmented, that is, be sent divided up in several telegrams. Within a data field, none, one, or multiple parameters in any combination of certain data types can be transported in a way that the parameters can be displayed directly. But please note that no floating point format would be possible here.

2.4.2 Communication on Application Level

Protocols on the application layer are described universally; hence communication on application layer is independent of the implemented physical medium. Virtually, they are transported from one application to the other, and in reality, they are transmitted with the help of Network Services and MOST NICs, as shown in Figure 2.15.

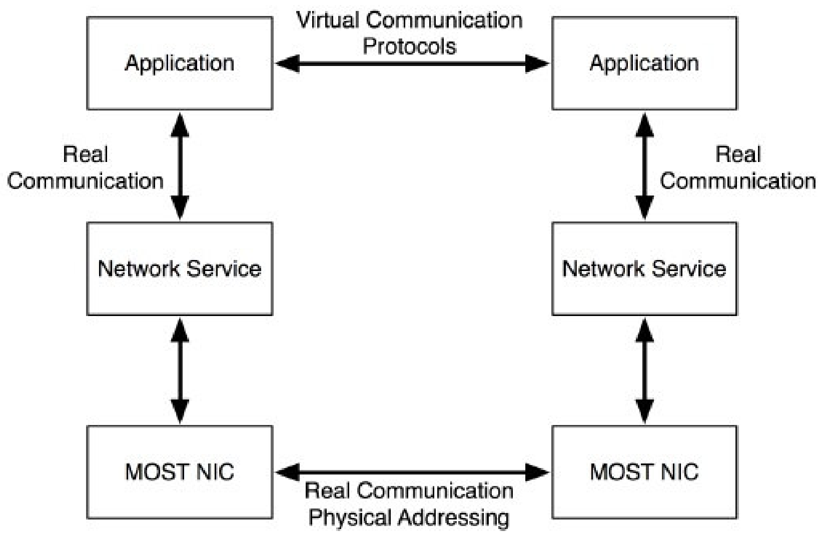


Figure 2.15 Virtual and real communication between two devices

All application protocols are finally transferred via the control channel of the MOST network. From the application's point of view, all protocols are passed on to the Network Service.

CHAPTER 3
MOST NetServices

This chapter consists of two parts. Section 3.1 gives an overview of MOST System Services in which MOST NetServices are included. It can be regarded as an introduction to the MOST Specification from a different point of view. In Section 3.2, MOST NetServices Layer 1 API is introduced, since we will construct our multimedia control program with the API. We survey MOST System Services and Basic Layer System Services API in this chapter, and would not go deep into the specifications.

3.1 System Services Overview [6]

The MOST System Services provide all basic functionality to operate a MOST System, and are consists of the following four parts:

- **Application Socket**
- **Basic Layer System Services**
- **Low Level System Services**
- **Stream Services**

As shown in Figure 3.1, every block in the diagram will be simply explained later.

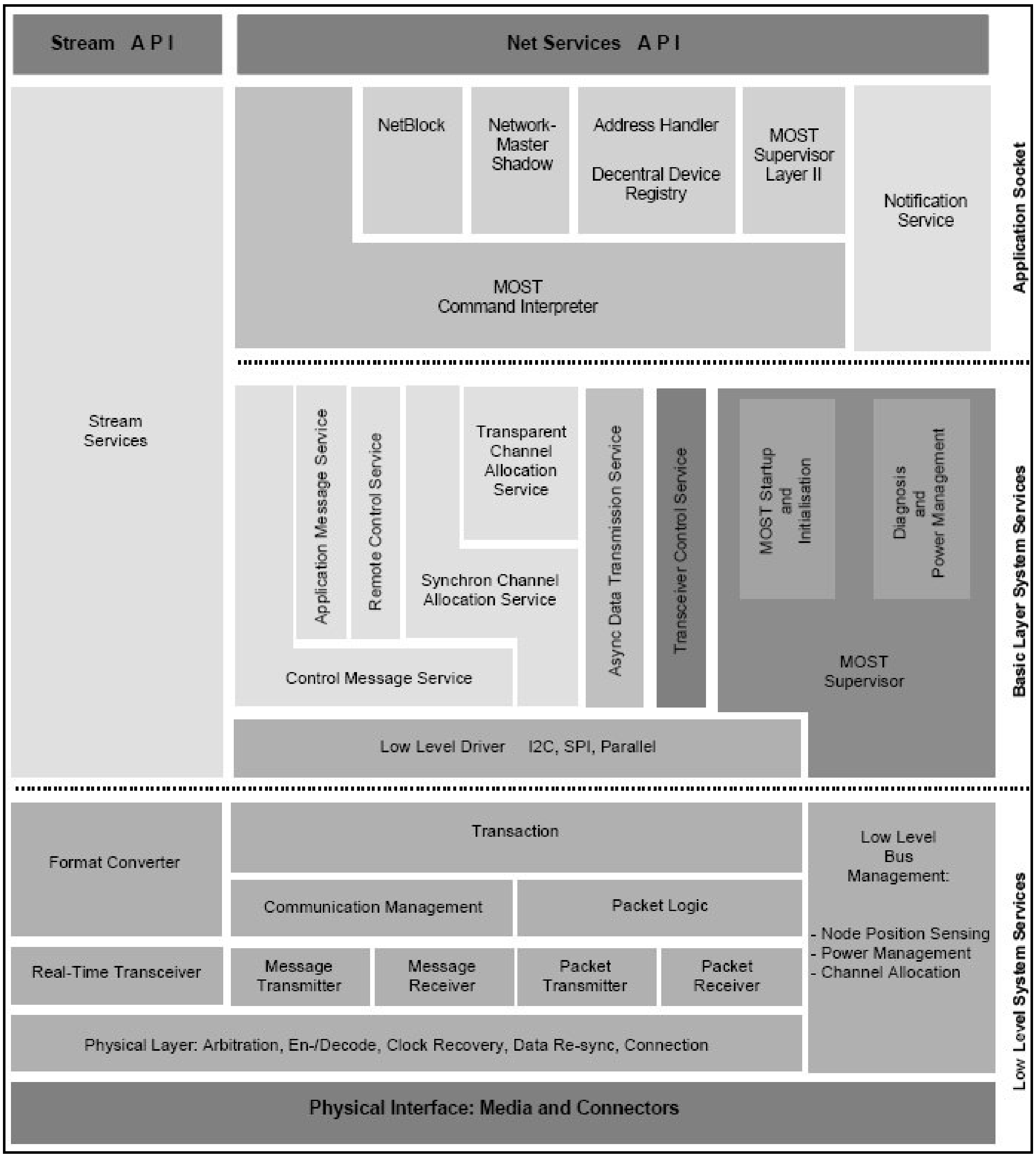

Stream API
Net Services API
NetBlock
Network-Master Shadow
Address Handler
Decentral Device Registry
MOST Supervisor Layer II
Notification Service
MOST Command Interpreter
Application Socket
Stream Services
Application Message Service
Remote Control Service
Transparent Channel Allocation Service
Synchron Channel Allocation Service
Async Data Transmission Service
Transceiver Control Service
MOST Startup and Initialisation
Diagnosis and Power Management
MOST Supervisor
Control Message Service
Low Level Driver I2C, SPI, Parallel
Basic Layer System Services
Format Converter
Transaction
Communication Management
Packet Logic
Low Level Bus Management:
- Node Position Sensing
- Power Management
- Channel Allocation
Real-Time Transceiver
Message Transmitter
Message Receiver
Packet Transmitter
Packet Receiver
Physical Layer: Arbitration, En-/Decode, Clock Recovery, Data Re-sync, Connection
Physical Interface: Media and Connectors
Low Level System Services

Figure 3.1 Block Diagram of the MOST System Services [5]

3.1.1 Application Socket

The Application Socket offers a wide variety of functions for building an application. Some of the functions are mandatory and some depend on the kind of application and are therefore optional. The Application Socket (Layer 2) together with the Basic Layer System Services

(Layer 1) are implemented in the NetServices software which contains all basic functionality a MOST Device needs.

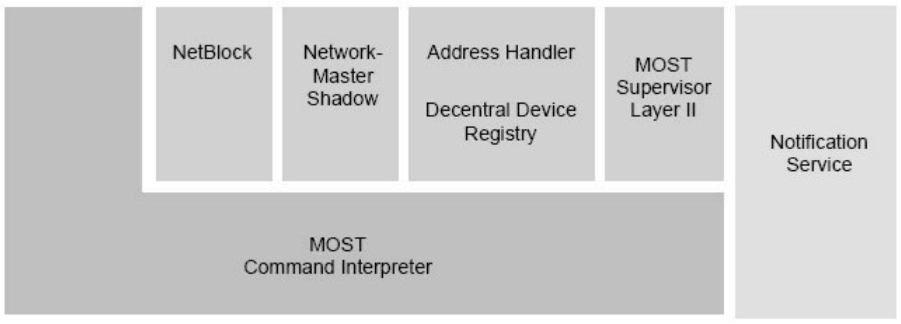


Figure 3.2 Application Socket

3.1.1.1 MOST Command Interpreter

The MOST Command Interpreter is based on the Application Message Service of the Basic Layer. It interprets and distributes received messages to the different function blocks of a MOST Device.

3.1.1.2 NetBlock

The Net Block is a function block which is mandatory for every MOST Device. It contains basic properties of the entire device (e.g. Addresses and available function blocks).

3.1.1.3 Network Master Shadow

The Network Master is a central instance (on highest level) in a MOST network, and it handles administrative tasks like the "Central Registry". Network Master Shadow is a local image of the Network Master which is needed for being able to receive messages of the Network Master. It is optional, since these functions can be handled in a de-central manner too.

3.1.1.4 Address Handler, De-Central Device Registry

If the actual logical MOST address of a function block is unknown, this optional service seeks it anywhere in the MOST network. It is possible to keep a local device registry, which then contains the respective logical addresses.

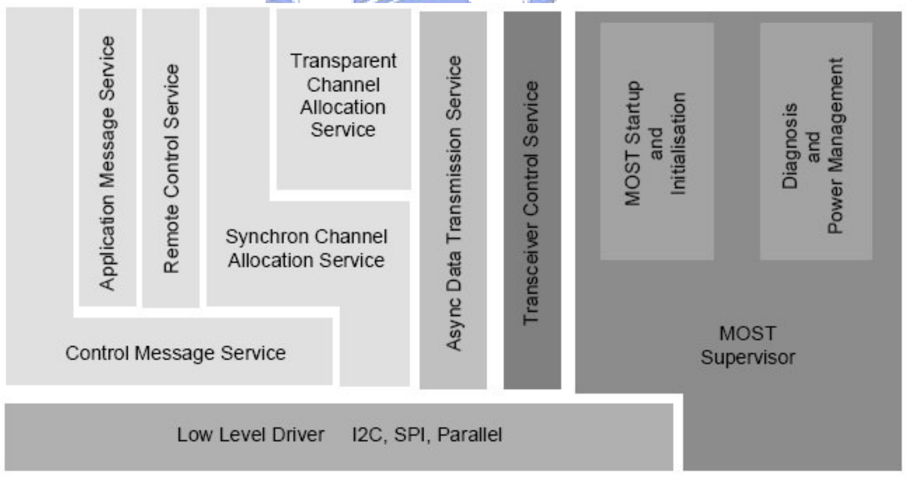
3.1.1.5 MOST Supervisor Layer II

MOST Supervisor Layer II provides initialization of a device on highest level, e.g. of the logical address.

3.1.1.6 Notification Service

This service organizes the sending of notification messages which are sent if a property is changed.

3.1.2 Basic Layer System Services



Application Message Service
Remote Control Service
Transparent Channel Allocation Service
Synchron Channel Allocation Service
Async Data Transmission Service
Transceiver Control Service
MOST Startup and Initialisation
Diagnosis and Power Management
MOST Supervisor
Control Message Service
Low Level Driver I2C, SPI, Parallel

Figure 3.3 Basic Layer System Services

The Basic Layer of the System Services is also known as MOST NetServices Layer 1 which provides comfortable access on the Low Level System Services.

3.1.2.1 MOST Supervisor

This service offers transceiver initialization and network startup on a higher level. Failure diagnosis and power management are also supported by MOST Supervisor.

3.1.2.2 Low Level Driver

The Low Level Driver provides an interface between microcontroller area and the MOST transceiver. It therefore offers different interface formats. For adapting to the respective micro controller (μC) environment, the Low Level Driver is complemented by user definable hardware specific functions.

3.1.2.3 Control Message Service

The Control Message Service buffers normal control or system messages when sending and receiving. Error detection and notification is provided too.

- **Application Message Service**

  When sending, this service splits application messages

  FBlockID.InstID.FktID.OPType(Data)

  into several segments with the length of 17 bytes. When receiving, segmented messages are restored to their original structure. Messages are buffered. Error detection and notification are also part of the Application Message Service.

- **Remote Control Service**

  This service provides the sending and receiving of remote system messages, including error detection and notification.

3.1.2.4 Synchronous Channel Allocation Service

The Synchronous Channel Allocation Service is based upon embedded Channel Allocation which is part of the Low Level Bus Management. It allocates or de-allocates resources for real time data streaming on the MOST network and "routes" this data to the desired destination. It contains functions to make real time data handling comfortable, e.g. by providing a list of the offsets of all physical channel related to one logical channel with respect to the synchronous data stream.

3.1.2.5 Transparent Channel Allocation Service

This service provides allocating and de-allocating as well as routing of transparent data.

3.1.2.6 Asynchronous Data Transmission Service

Sending and receiving of asynchronous data is the task of this service. Errors are detected and notified.

3.1.2.7 Transceiver Control Service

This service provides access to configuration and address registers of the MOST transceiver.

3.1.3 Low Level System Services

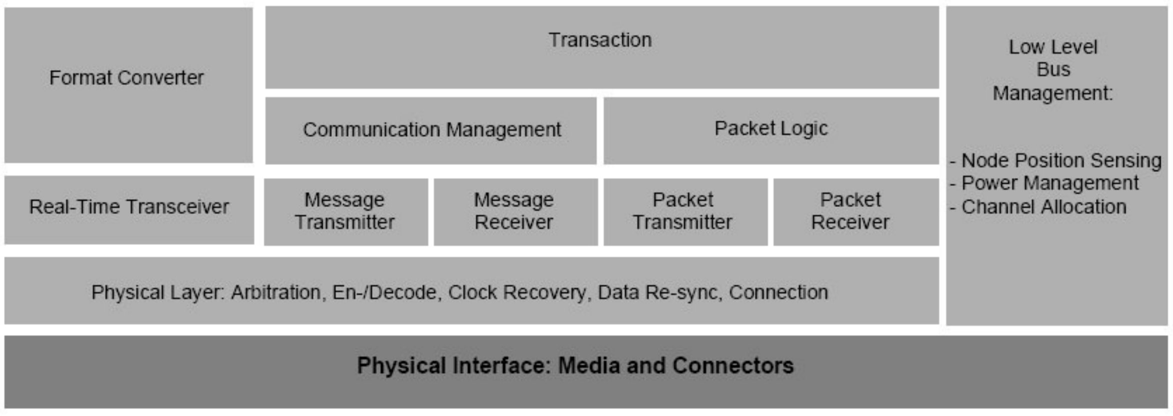


Figure 3.4 Low Level System Services

Low Level System Services, except for the physical interface, are implemented in the MOST Transceiver (OS8104) [9].

3.1.3.1 Physical Interface

As described in Section 2.3.4.

3.1.3.2 Physical Layer

The physical layer as implemented in the MOST transceiver OS 8104, provides connection of the MOST devices. In addition to that, clock recovery, data de- and encoding as well as arbitration for asynchronous channels are implemented.

3.1.3.3 Low Level Bus Management

The Low Level Bus Management provides a set of functions which handles Power Management as well as Channel Allocation for real - time data, and Node Position Sensing.

3.1.3.4 Packet Logic

Packet Logic controls sending and receiving of packet/bulk data in Packet Transmitter and Packet receiver. It includes error detection for packet data.

3.1.3.5 Communication Management

Communication management controls sending, receiving and error detection of MOST control messages (normal messages as well as system messages).

3.1.3.6 Transaction Level

Provides access to data received either as packet data, or MOST control message, and to the transmit areas for sending data (asynchronous and control).

3.1.3.7 Real Time Transceiver

The Real Time Transceiver receives real time data from the external world, or sends this kind of data to the external world. In addition to that, it performs "Routing" on real time data, that means it puts the data to the right destination (network or application).

3.1.3.8 Format Converter

The format converter converts a variety of serial source data formats (S/PDIF, Sony, Matsushita, I2S, ...) into a format that can easily be transported via MOST network, or vice versa.

3.1.4 Stream Services

The Stream Services provide transport services for real-time source data. This allows handling source data in the respective parts of the application area.

3.2 NetServices Layer 1 API [2]

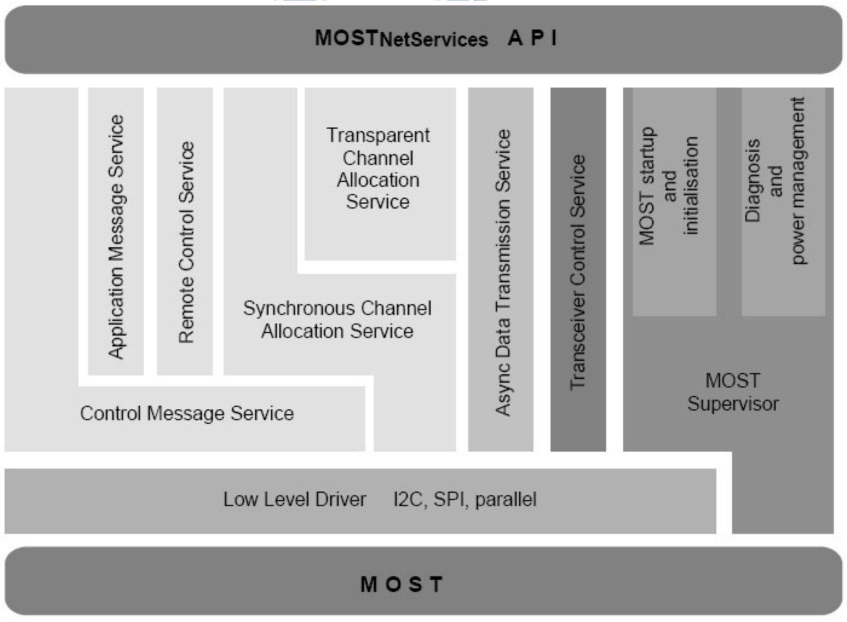

MOSTNetServices API
Application Message Service
Remote Control Service
Transparent Channel Allocation Service
Synchronous Channel Allocation Service
Async Data Transmission Service
Transceiver Control Service
MOST startup and initialisation
Diagnosis and power management
MOST Supervisor
Control Message Service
Low Level Driver I2C, SPI, parallel
MOST

Figure 3.5 Structure of MOST NetServices Layer 1

MOST NetServices Layer 1 API is the interface between the MOST Low Level System Services and an application. All functions are combined in a library that they can be included if required. MOST NetServices API is implemented in ANSI C and can be adapted by the file "*adjust.h*" which contains all the parameters needed to set up NetServices according to the application.

In the sections below, the available services are listed with their features, and for each of the service a brief explanation is given.

3.2.1 MOST NetServices Kernel (MNS)

- Initialization of all services
- NetServices trigger functions

This module initializes all MOST services and implements all NetServices trigger functions i.e., callback functions.

- **Callback Functions**

  Communication between MOST NetServices and application is handled via "Callback Functions". Those functions are called by MOST NetServices and therefore must be made available by the user in the application.

  Depending on the user's demands how the application reacts when the functions is called, Callback Functions of type "A" may be ignored, while Callback Functions of type "B" have to return the requested value.

3.2.2 MOST Supervisory and Startup Functions (MSV)

- Transceiver Initialization
- Master/Slave Selection

- Compiler Selectable Source Port Configuration
- Network Start Up and Shut Down
- Power Management
- Failure Diagnosis and Reporting
- Self Testing Services

This service provides a state machine which supervises the NetInterface and influences it if needed. For more information on the state machine and its dynamic behavior, please refer to [2].

3.2.3 Control Message Service (CMS)

- Initialization
- Control Message Receive Service
- Control Message Transmit Service
- Message Buffering and Handling
- Error Handling and Notification

This service allows sending and receiving of control messages. It is divided into a Tx (Transmission) and an Rx (Reception) section. Messages in both sections are represented by respective data structures, Ctrl_Tx_Type and Ctrl_Rx_Type, which will be introduced in Section 4.3.1.2.

3.2.4 Application Message Service (AMS)

- Initialization
- Application Message Send Handler
- Application Message Receive Handler
- Message Buffering and Handling

- Error Handling and Notification

Based on CMS, this service organizes incoming and outgoing messages on a higher level. The length of a message is no longer limited to 17 bytes. For sending and receiving application messages, AMS is separated in a TX and a RX section, so both sections are independent from each other. Since most functionalities of our program are accomplished by AMS, its usage will be explained in the next chapter.

3.2.5 Remote Control Service (RCS)

- Initialization
- Remote Write Service
- Remote Read Service
- Remote Error Handling and Notification

This service provides the sending of remote control messages that enable the node to access registers in other transceivers. It is subdivided into the services Remote Read and Remote Write which are independent from each other.

3.2.6 Synchronous Channel Allocation Service (SCS)

- Channel Allocation
- Allocation and Routing
- De-Allocation
- De-Allocation and Routing Disconnect
- Source Connect
- Source Disconnect
- Sink Connect
- Sink Disconnect

- Detect Channel by Label

This service handles reserving and releasing of synchronous source data channels. It can also establish connections between the source data ports of the transceiver and the data channels. The connecting is done by writing to the Routing Engine (RE) of the MOST Transceiver.

3.2.7 Transparent Channel Allocation Service (TCS)

- Channel Allocation
- Allocation and Routing
- De-Allocation
- De-Allocation and Routing Disconnect
- Source Connect
- Source Disconnect
- Sink Connect
- Sink Disconnect

Built up in the same modular manner as SCS, this service contains functions for reserving and releasing of channels for transporting transparent data, and for connecting inputs and outputs of the MOST Transceiver.

3.2.8 Asynchronous Data Transmission Service (ADS)

- Initialize – SetDataAddress
- Buffered Transfer of Data Packages
- Buffered Receive of Data Packages
- Data Transfer Error Handling

It provides functionality for using asynchronous data channels. Buffers and error handling are supported.

3.2.9 MOST Transceiver Control Service (MCS)

- Addressing
- Accessing important Transceiver's Register
- Selecting RMCK Frequency
- Master Functions

This service provides functions for accessing important registers on the transceiver, to adjust certain properties like its logical address or group address. It also adds functions used by Timing Masters only.

CHAPTER 4
Multimedia Applications

4.1 Hardware Overview

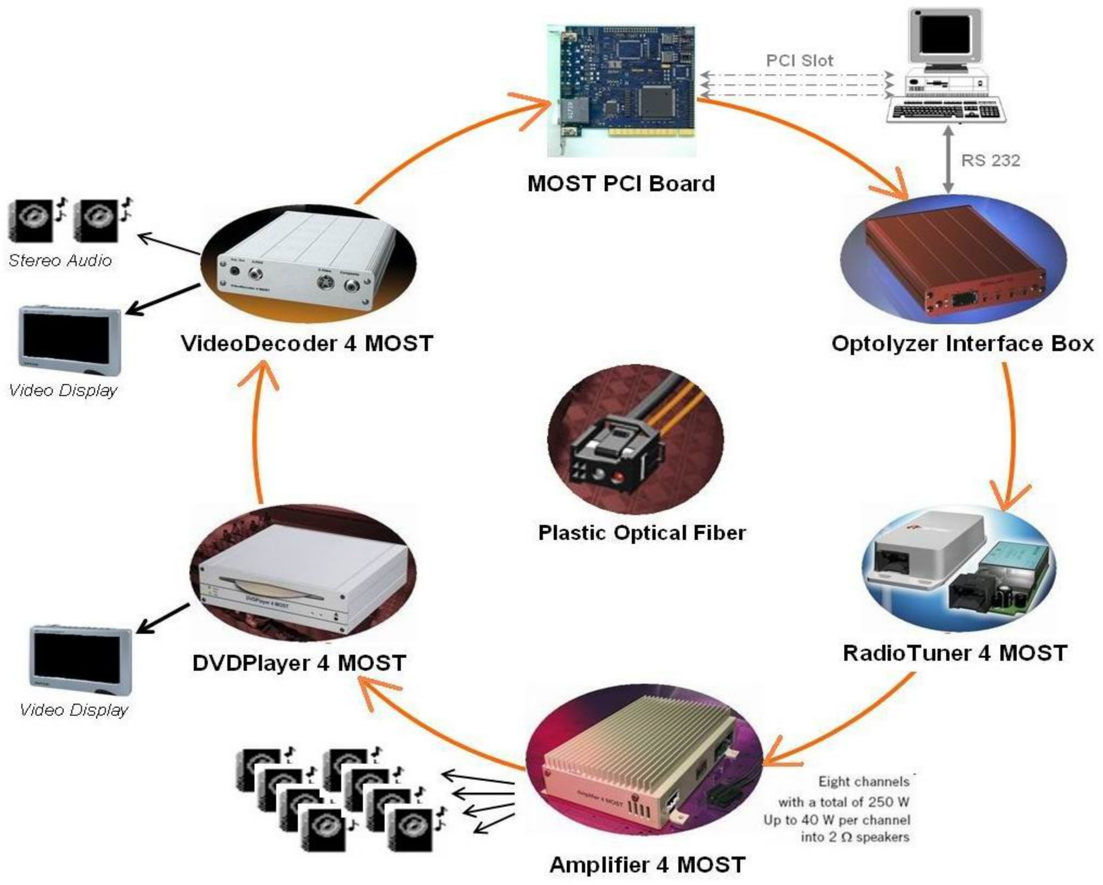


Figure 4.1 MOST Devices assembled for our applications

In the beginning of Chapter 4, this section provides an overview of the MOST devices assembled for our applications; moreover, Figure 4.1 depicts how the devices are connected via Plastic Optical Fibers (POF) in a ring topology. Manufactured by OASIS SiliconSystems (which is now part of SMSC), the rapid prototyping devices provide a convenient way to test and demonstrate multimedia applications under MOST network. In addition to that, the MOST PCI Board and Optolyzer Interface Box are also useful to our applications.

4.1.1 RadioTuner4MOST [10]

Figure 4.2 RadioTuner4MOST

The RadioTUner4MOST is a rapid prototyping tool that demonstrates the flexibility of SMSC's application specific media controller series. The tuner is implemented using a single OS88350 Radio Controller which does all the processing necessary to implement an AM/FM receiver and puts its audio information onto a MOST network.

4.1.2 Amplifier4MOST [11]

Figure 4.3 Amplifier4MOST

The Amplifier4MOST is an 8-channel power amplifier with MOST network interface, and a rapid prototyping tool as well. It is implemented using a single OS88558 amplifier controller which does all the processing necessary to transport audio information from a MOST network to external speakers.

4.1.3 DVDPlayer4MOST [12]

Figure 4.4 DVDPlayer4MOST

The rapid prototyping device, DVDPlayer4MOST, is a versatile video and audio playback system that distributes DVD video and audio as well as standard CD audio to a MOST network. The multimedia processor OS8805 from SMSC is used. The chip communicates with the MOST network and its built-in firmware controls the DVD module as well as the MPEG encoder circuitries without any glue logic.

4.1.4 VideoDecoder4MOST [13]

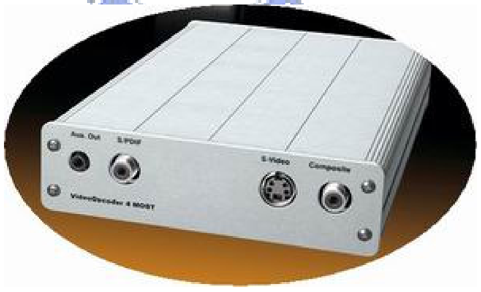


Figure 4.5 VideoDecoder4MOST

The VideoDecoder4MOST is a multimedia device that converts MPEG-2 streams broadcasted over a MOST network to analog A/V signals. It sends out both CVBS (Color Video Blanking Signal) composite and S-Video (Y/C) signals for an external PAL video display. An analog stereo output is available for connecting external audio amplifiers. In addition, the audio signal is digitized and may be fed into the MOST network.

4.1.5 MOST PCI Board [14]

Figure 4.6 MOST PCI Board [15]

The MOST PCI Board belongs to PC Interfaces MOST25 oPHY that provide a high-speed connection between a PC and a MOST network. It is allowed to access to real-time streaming transfer, packet data transport, and control message service of the MOST network. The chip set used for PC Interfaces MOST25 oPHY consists of the OS8104 MOST Network Interface Controller and a PC interface chip. Whereas the OS8104 is used as an interface to the MOST network, the PC interface chip implements the connection to the PCI bus.

4.1.6 Optolyzer Interface Box [16]

Figure 4.7 Optolyzer Interface Box

The OptoLyzer is an analysis and development tool for exploring MOST networks. It provides full access to the MOST real-time channels as well as to control messages of a MOST network. The control and the communication of messages are available through the 115 kbps RS232 serial interface between the host and the OptoLyzer Interface Box.

4.2 Software Utilities

With screenshots provided, this section introduces several software utilities which are used when implementing our multimedia applications under MOST network. They are:

- **MOST Interface Control [17] -** It is used to make communication with the MOST PCI Board. It can also start the MOST network by setting the device as Timing Master.
- **MOST Radar [18] -** A very useful tool for gathering the information of the MOST network. Accessing registers of the MOST Transceiver of a node is reachable.
- **GraphEdit -** With GraphEdit, we can build and test multimedia streaming applications in block flow diagram form.

4.2.1 MOST Interface Control

MOST Interface Control (MICtrl) is a small application made by Oasis SiliconSystems, which can access the registers of the MOST Transceiver chips located in underlying MOST hardware devices like:

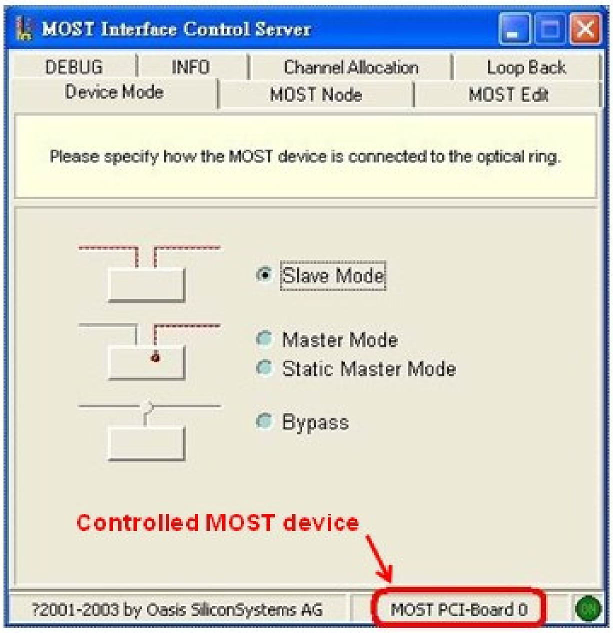
- MOST PCI Board
- MOST ISA Board
- OptoLyzer4MOST Interface Box

Please note: After upgrading the MOST PCI Board drivers from MOST25 v2.7 to MOST50 v1.4.4, it is necessary to start MICtrl before we can access the MOST PCI Board in our applications.

4.2.1.1 Device Mode

As shown in Figure 4.8, MICtrl is connected to the MOST PCI Board. Under “Device Mode”, we can set the device in “Master Mode” to start the MOST network, or leave it in “Slave Mode” that the MOST network could be started by another device. In “Slave Mode”, the

MOST PCI Board will behave as a regular node in the optical ring. If “Static Master Mode” is selected, the MOST Transceiver needs to be reset, and the Master will never turn off the light (i.e. even if the ring is not closed). If the device is set to “Bypass”, then it is not visible in the network.

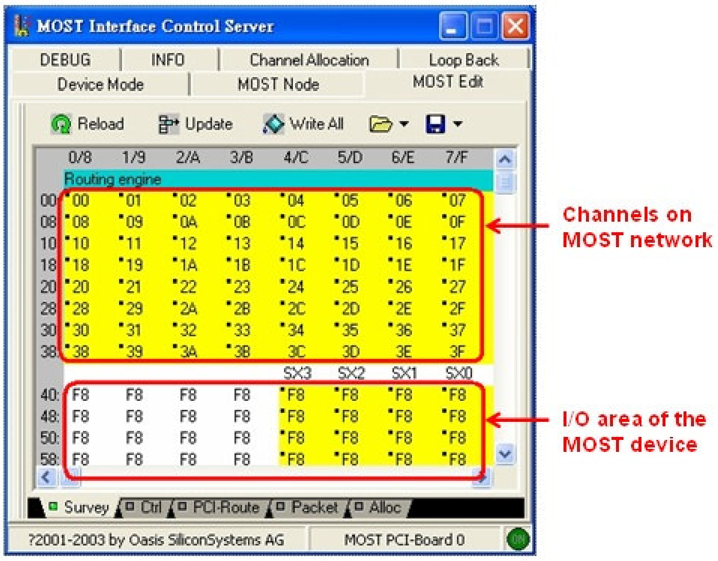


Please specify how the MOST device is connected to the optical ring.

[✓ Slave Mode] Master Mode Static Master Mode Bypass

Figure 4.8 Device Mode of MICtrl

4.2.1.2 MOST Edit



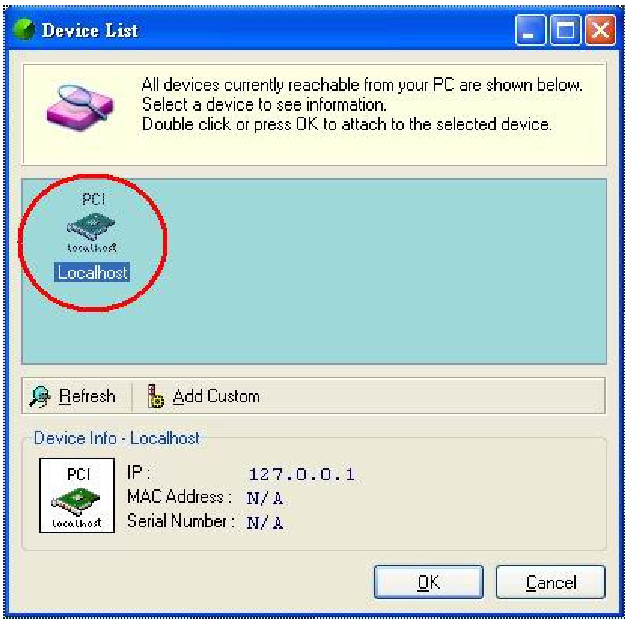
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
	Routing engine							
00	•00	•01	•02	•03	•04	•05	•06	•07
08	•08	•09	•0A	•0B	•0C	•0D	•0E	•0F
10	•10	•11	•12	•13	•14	•15	•16	•17
18	•18	•19	•1A	•1B	•1C	•1D	•1E	•1F
20	•20	•21	•22	•23	•24	•25	•26	•27
28	•28	•29	•2A	•2B	•2C	•2D	•2E	•2F
30	•30	•31	•32	•33	•34	•35	•36	•37
38	•38	•39	•3A	•3B	3C	3D	3E	3F
					SX3	SX2	SX1	SX0
40:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
48:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
50:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
58:	F8	F8	F8	F8	•F8	•F8	•F8	•F8

Figure 4.9 MOST Edit of MICtrl

In addition to connecting the device to MOST networks, “MOST Edit” of MICtrl is useful for viewing and editing of parameters of the MOST Transceiver on register level, as shown in Figure 4.9. The upper register area of the Routing Engine (RE) represents the channels on the MOST network, which are capable to transport stream data. The lower area starting from 0x40 represents the input/output channels of the MOST device. The routing is established, by writing the channel addresses of the registers of the MOST device I/O area (Address References) into the registers representing the MOST channels [19]. Practical examples for RE will be given in later sections of this chapter.

4.2.2 MOST Radar

MOST Radar is an analyzing tool for MOST networks with the possibility to edit channels and device settings. It is a software product provided by SMSC; hence, a license key is needed to start the program. In the beginning, we should select a device that MOST Radar can attach to, to get information from the MOST network. MOST Radar can be used as a Plug-In of OptoLyzer4MOST PC software or MOST Interface Control. In our case, only “MICtrl Server“ is available to host application, as shown in Figure 4.10.



All devices currently reachable from your PC are shown below.
Select a device to see information.
Double click or press OK to attach to the selected device.

Device Info - Localhost

IP: 127.0.0.1
MAC Address: N/A
Serial Number: N/A

Figure 4.10 Attaching MOST Radar to a device

4.2.2.1 Main Window and Node Symbol

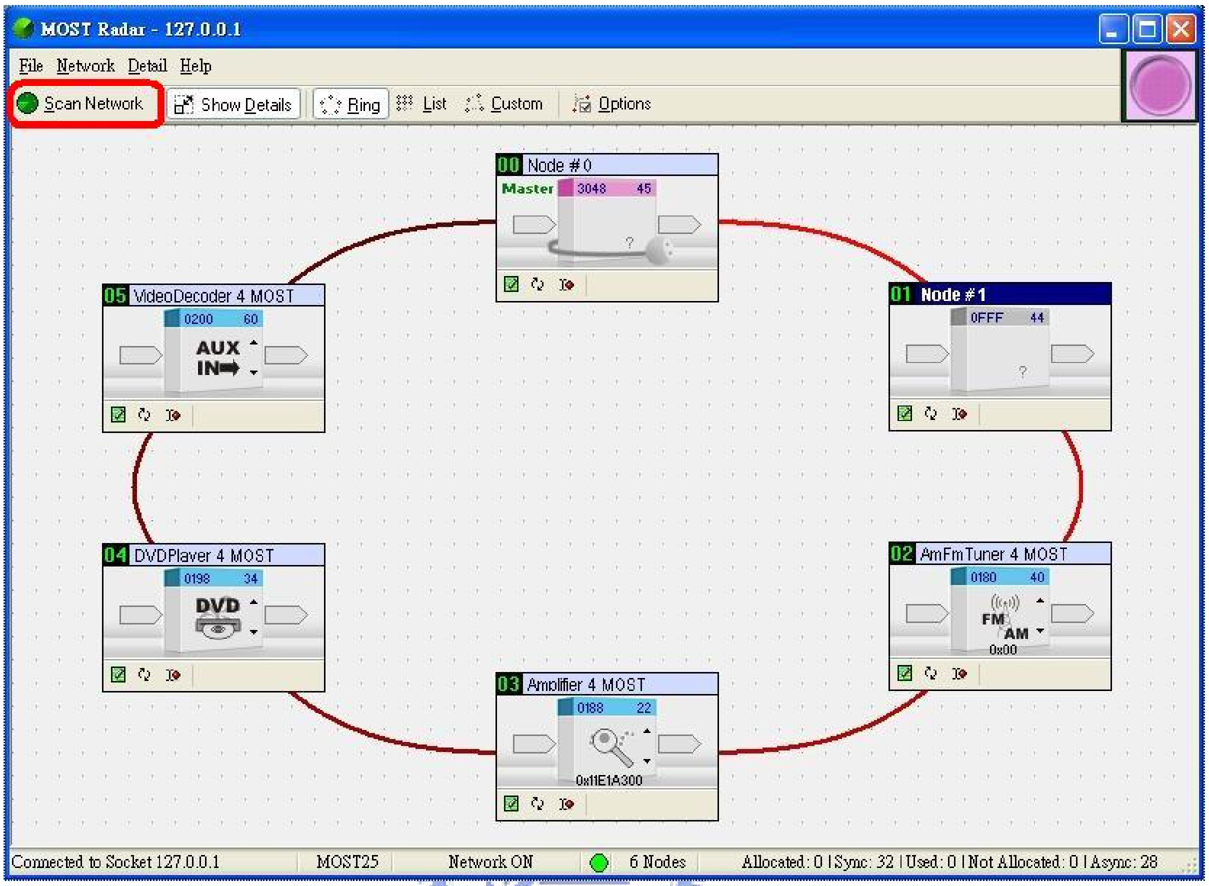


Figure 4.11 Main Window of MOST Radar

Figure 4.11 shows the main window of MOST Radar. The main window gives a survey of the entire MOST network and its status in a ring view. Clicking onto the "Scan Network" button activates the scanning, and then information on the individual nodes is collected during the scanning of a MOST network.

MOST Radar generates a node symbol for each active node in the ring. After scanning a node, information is displayed as shown in Figure 4.12; furthermore, Table 4.1 describes every single part of a node symbol.

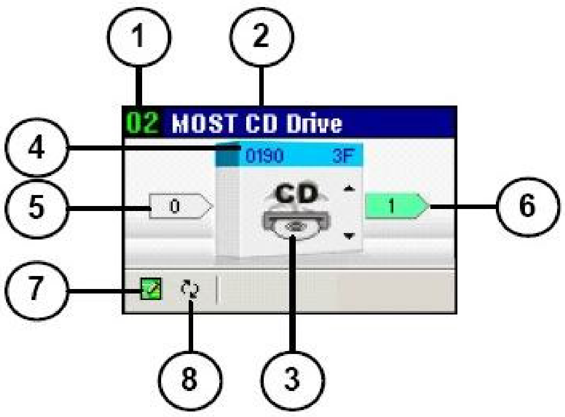


Figure 4.12 Node Symbol of MOST Radar [18]

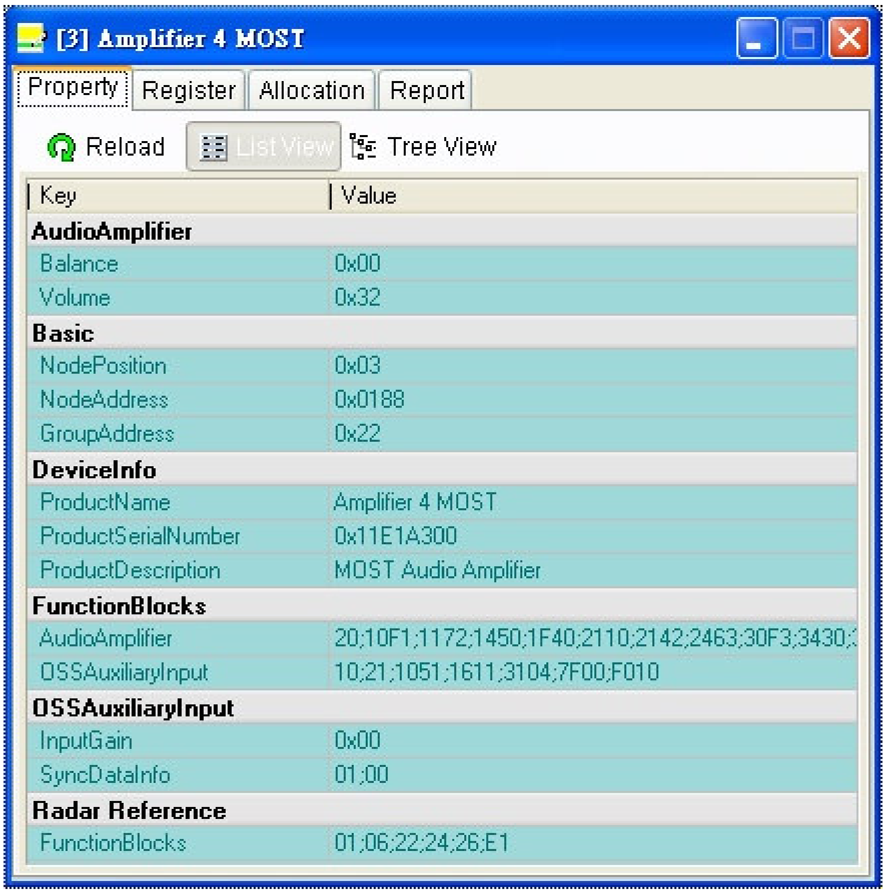
Table 4.1 Node Symbol Description [18]

No.	Description
1	Node Position. This number refers to the position of the device in the ring and starts at zero (from timing master).
2	Product name.
3	Function Block icon.
4	Node Address / Group Address
5	Allocation indicator from MOST network, colored if there is an allocation. The number inside indicates the amount of incoming MOST channels relating to this device. Information is only displayed if available.
6	Allocation indicator to MOST network, colored if there is an allocation. The number inside indicates the amount of outgoing MOST channels relating to this device. Information is only displayed if available.
7	Property button, click to open the property window.
8	Update button, click to refresh the information on this device.

4.2.2.2 Property Window

The property window presents information of a node in detail. It is possible to open a property window for each node simultaneously. The property window provides four different tabs, shown as follows: (take “Node 3 - Amplifier4MOST” for example)

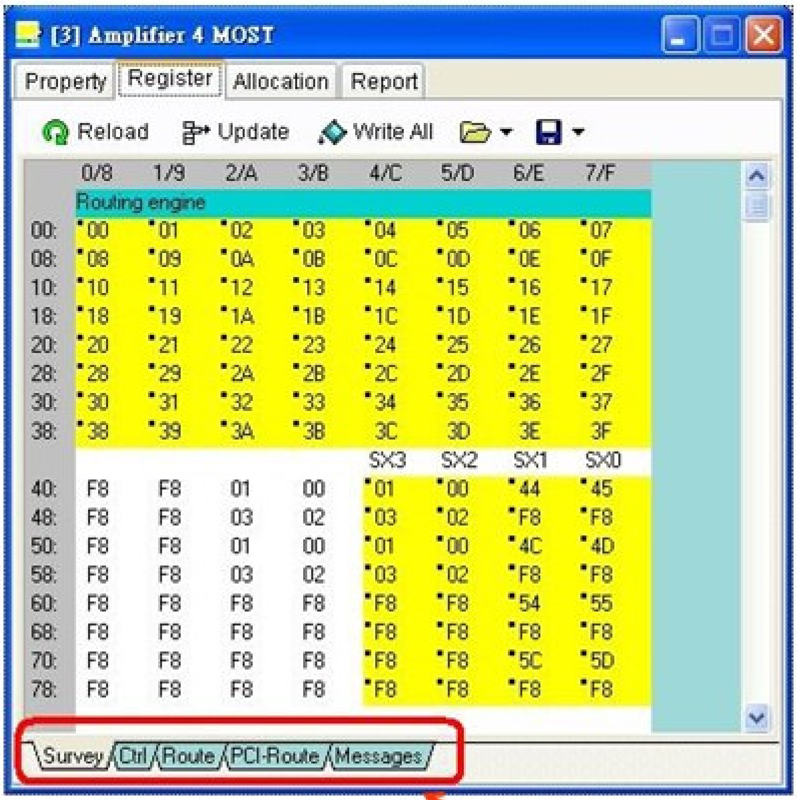
- **Property Tab -** The Property tab provides a reload button and two different views for the properties. Figure 4.13 shows the “List View”.



Key	Value
AudioAmplifier	
Balance	0x00
Volume	0x32
Basic	
NodePosition	0x03
NodeAddress	0x0188
GroupAddress	0x22
DeviceInfo	
ProductName	Amplifier 4 MOST
ProductSerialNumber	0x11E1A300
ProductDescription	MOST Audio Amplifier
FunctionBlocks	
AudioAmplifier	20;10F1;1172;1450;1F40;2110;2142;2463;30F3;3430;
OSSAuxiliaryInput	10;21;1051;1611;3104;7F00;F010
OSSAuxiliaryInput	
InputGain	0x00
SyncDataInfo	01;00
Radar Reference	
FunctionBlocks	01;06;22;24;26;E1

Figure 4.13 Property Tab

- **Register Tab -** As shown in Figure 4.14, the Register tab is similar to "MOST Edit" of MICtrl which allows the changing of parameters of OS8104 MOST chips at register level, only MICtrl must work with PC interface devices.

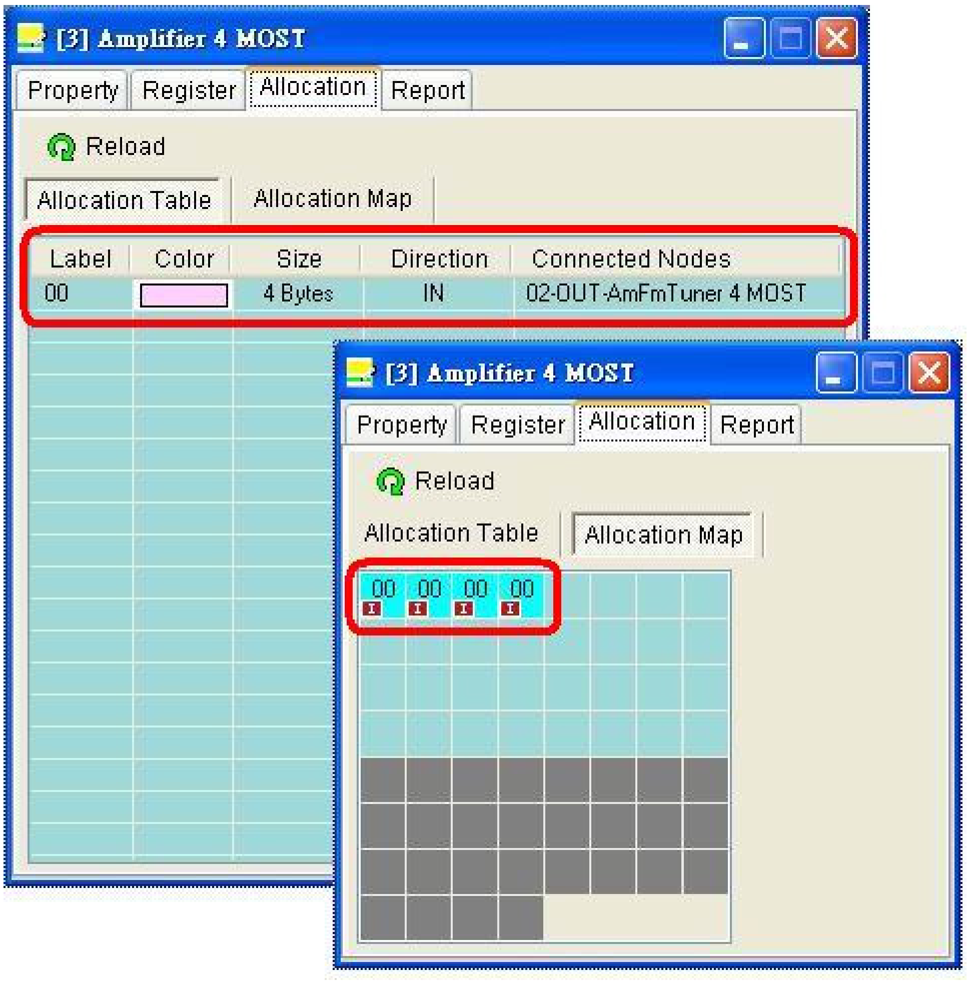


	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
	Routing engine							
00:	•00	•01	•02	•03	•04	•05	•06	•07
08:	•08	•09	•0A	•0B	•0C	•0D	•0E	•0F
10:	•10	•11	•12	•13	•14	•15	•16	•17
18:	•18	•19	•1A	•1B	•1C	•1D	•1E	•1F
20:	•20	•21	•22	•23	•24	•25	•26	•27
28:	•28	•29	•2A	•2B	•2C	•2D	•2E	•2F
30:	•30	•31	•32	•33	•34	•35	•36	•37
38:	•38	•39	•3A	•3B	3C	3D	3E	3F
					SX3	SX2	SX1	SX0
40:	F8	F8	01	00	•01	•00	•44	•45
48:	F8	F8	03	02	•03	•02	•F8	•F8
50:	F8	F8	01	00	•01	•00	•4C	•4D
58:	F8	F8	03	02	•03	•02	•F8	•F8
60:	F8	F8	F8	F8	•F8	•F8	•54	•55
68:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
70:	F8	F8	F8	F8	•F8	•F8	•5C	•5D
78:	F8	F8	F8	F8	•F8	•F8	•F8	•F8

Figure 4.14 Register Tab

Please note: Figure 4.9 and Figure 4.14 only display the "Route" (Routing Engine) section of the register editor. There are some registers arranged in other editor pages, e.g. "Transceiver Control" registers or "Message handling and controlling" registers.

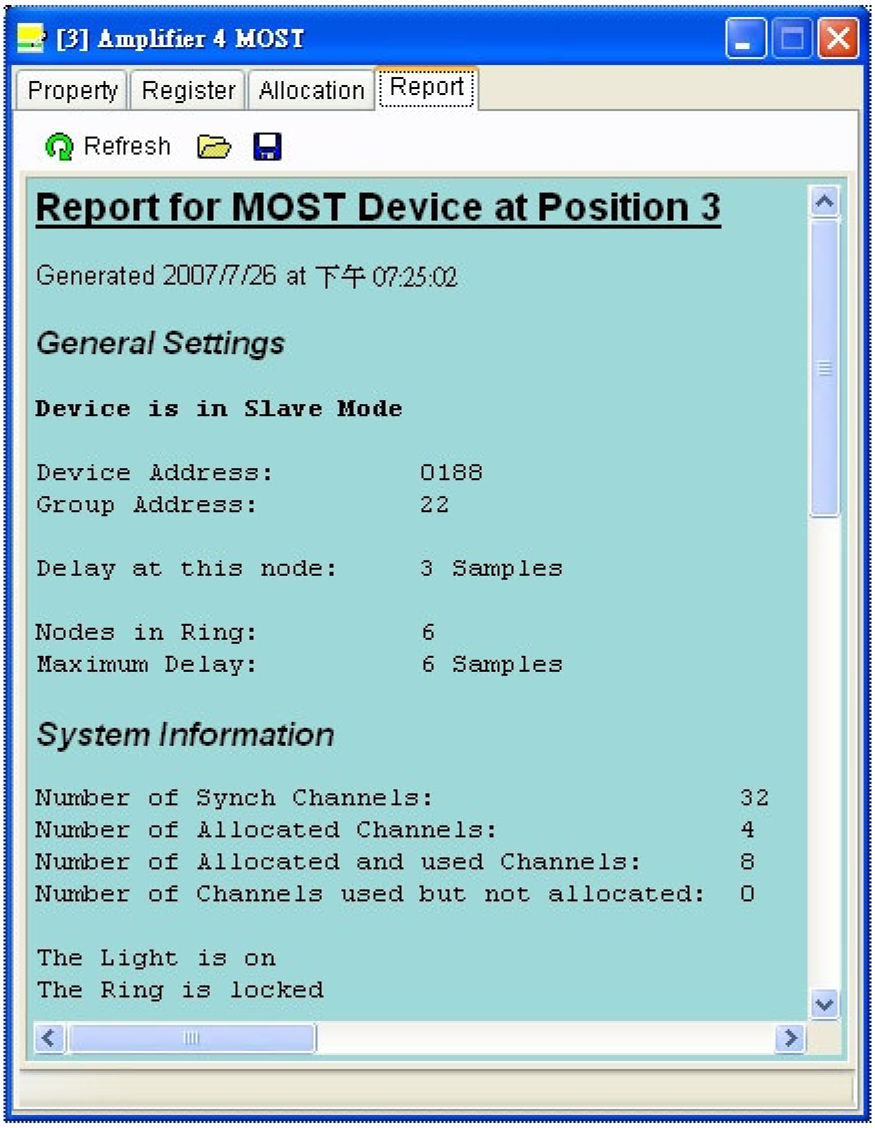
- **Allocation Tab -** Figure 4.15 is a combination of "Allocation Table" and "Allocation Map" screenshots. We can see that 4 bytes of channels labeled with "00" are allocated and routed in; furthermore, we can see that the device is connected to the AmFmTuner4MOST (with Node Position 02) from which synchronous data streams are being routed out.



Label	Color	Size	Direction	Connected Nodes
00	▭	4 Bytes	IN	02-OUT-AmFmTuner 4 MOST

Figure 4.15 Allocation Tab

- **Report Tab -** If the Report tab is selected, a report of the node is generated. All details are collected and presented in the window as depicted in Figure 4.16. The report can be saved for further processing, or previous saved reports can be loaded to view it again. Click the Refresh button to collect current values.



Report for MOST Device at Position 3

Generated 2007/7/26 at 下午 07:25:02

General Settings

```
Device is in Slave Mode

Device Address:          0188
Group Address:           22

Delay at this node:      3 Samples

Nodes in Ring:           6
Maximum Delay:           6 Samples
```

System Information

```
Number of Synch Channels:                    32
Number of Allocated Channels:                4
Number of Allocated and used Channels:       8
Number of Channels used but not allocated:   0

The Light is on
The Ring is locked
```

Figure 4.16 Report Tab

4.2.3 GraphEdit

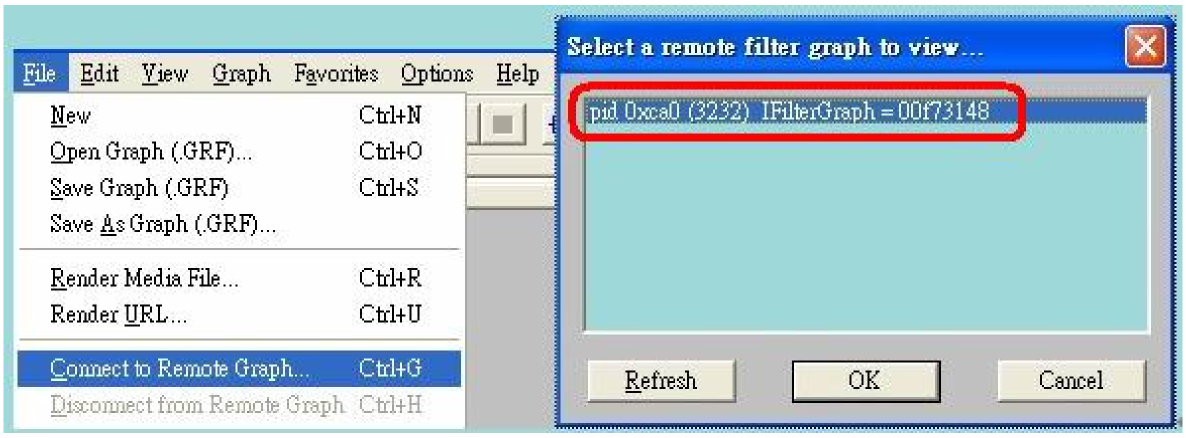

File
Edit
View
Graph
Favorites
Options
Help
New Ctrl+N
Open Graph (.GRF)... Ctrl+O
Save Graph (.GRF) Ctrl+S
Save As Graph (.GRF)...
Render Media File... Ctrl+R
Render URL... Ctrl+U
Connect to Remote Graph... Ctrl+G
Disconnect from Remote Graph Ctrl+H
Select a remote filter graph to view...
pid 0xca0 (3232) IFilterGraph = 00f73148
Refresh
OK
Cancel

Figure 4.17 Connect GraphEdit to Remote Graph

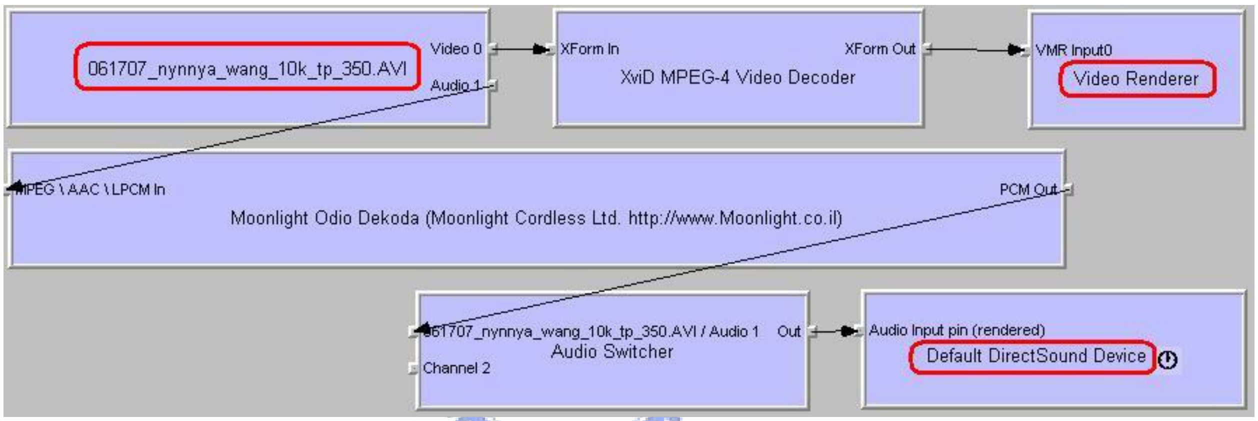
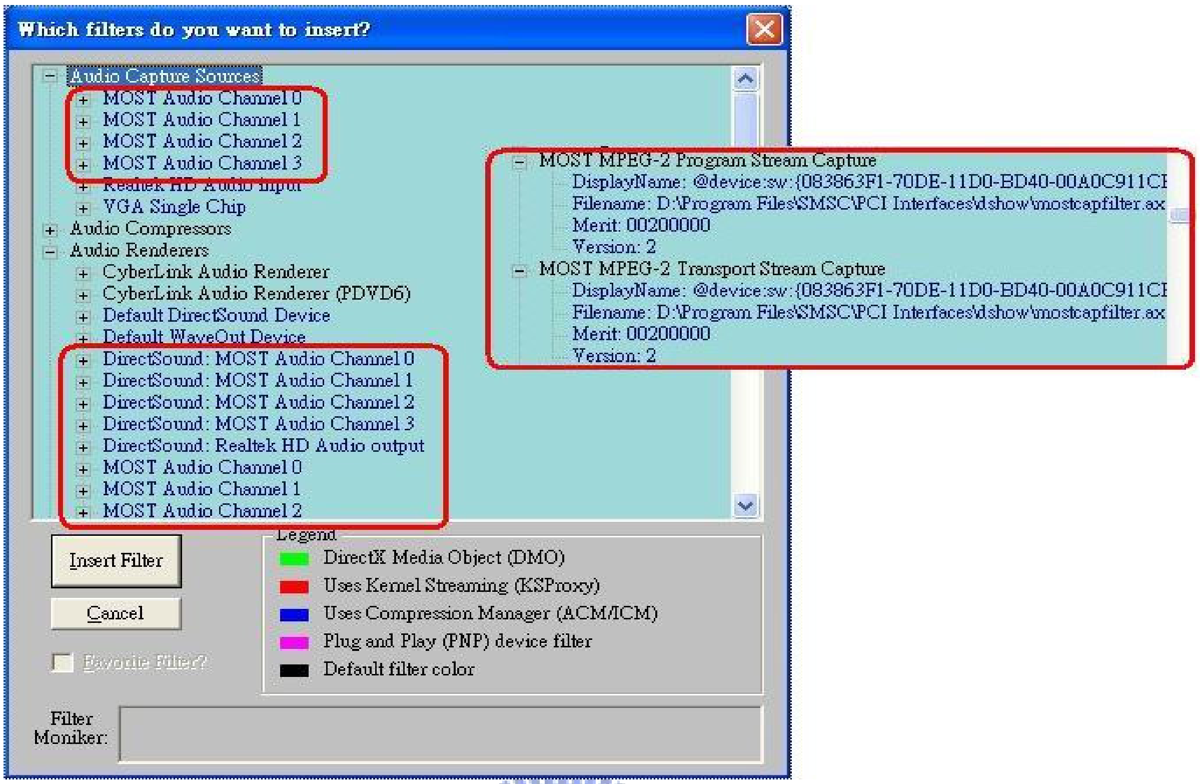

061707_nynnya_wang_10k_tp_350.AVI
Video 0
Audio 1
XForm In
XForm Out
XviD MPEG-4 Video Decoder
VMR Input0
Video Renderer
MPEG \ AAC \ LPCM In
PCM Out
Moonlight Odio Dekoda (Moonlight Cordless Ltd. http://www.Moonlight.co.il)
061707_nynnya_wang_10k_tp_350.AVI / Audio 1
Out
Audio Switcher
Channel 2
Audio Input pin (rendered)
Default DirectSound Device

Figure 4.18 A Filter Graph Example

GraphEdit is a visual tool for building and testing filter graphs. It is provided as an executable with the Microsoft DirectX SDK (Software Development Kit). With GraphEdit, we can quickly build and test filter graphs to see if they function as we expect [20]. It is also possible to view a filter graph created by an application running in another process. Figure 4.17 shows how we connect GraphEdit to a remote graph, and Figure 4.18 depicts the connected graph. We can see that the AVI file being played is transferred into video and audio streams, rendered by “Video Renderer” and “Default DirectSound Device”, respectively.

- **DirectShow Filters for MOST**


Which filters do you want to insert?

```
⊟ Audio Capture Sources
   ⊞ MOST Audio Channel 0
   ⊞ MOST Audio Channel 1
   ⊞ MOST Audio Channel 2
   ⊞ MOST Audio Channel 3
   ⊞ Realtek HD Audio input
   ⊞ VGA Single Chip
⊞ Audio Compressors
⊟ Audio Renderers
   ⊞ CyberLink Audio Renderer
   ⊞ CyberLink Audio Renderer (PDVD6)
   ⊞ Default DirectSound Device
   ⊞ Default WaveOut Device
   ⊞ DirectSound: MOST Audio Channel 0
   ⊞ DirectSound: MOST Audio Channel 1
   ⊞ DirectSound: MOST Audio Channel 2
   ⊞ DirectSound: MOST Audio Channel 3
   ⊞ DirectSound: Realtek HD Audio output
   ⊞ MOST Audio Channel 0
   ⊞ MOST Audio Channel 1
   ⊞ MOST Audio Channel 2

⊟ MOST MPEG-2 Program Stream Capture
     DisplayName: @device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE
     Filename: D:\Program Files\SMSC\PCI Interfaces\dshow\mostcapfilter.ax
     Merit: 00200000
     Version: 2
⊟ MOST MPEG-2 Transport Stream Capture
     DisplayName: @device:sw:{083863F1-70DE-11D0-BD40-00A0C911CE
     Filename: D:\Program Files\SMSC\PCI Interfaces\dshow\mostcapfilter.ax
     Merit: 00200000
     Version: 2
```

Insert Filter

Cancel

☐ Favorite Filter?

Legend
- 🟩 DirectX Media Object (DMO)
- 🟥 Uses Kernel Streaming (KSProxy)
- 🟦 Uses Compression Manager (ACM/ICM)
- 🟪 Plug and Play (PNP) device filter
- ⬛ Default filter color

Filter Moniker:

Figure 4.19 DirectShow Filters for MOST

Microsoft DirectShow is an architecture for streaming media on the Microsoft Windows platform [21]. Depending on the functionality, DirectShow filters can be roughly divided into three types: Source Filters, Transform Filters and Rendering Filters. With the drivers for MOST PCI Board installed, several Source Filters and Rendering Filters are available for constructing multimedia streaming applications for MOST. Figure 4.19 shows a list of filters to be inserted in GraphEdit; furthermore, filters for MOST are circled in this figure. “MOST Audio Channels 0...3” have both “Audio Capture Sources” and “Audio Renderers” uses, which means audio streams can be transferred up or down via MOST Audio Channels. For capturing video streams from the MOST network, “MPEG-2 Program Stream (PS) Capture” and “MPEG-2 Transport Stream (TS) Capture” are provided.

4.3 Multimedia Control Program

NSL1_EX is an example application provided by SMSC, and it is also obtainable on the website of SMSC: http://www.smsc-ais.com/AIS/content/view/125/151/ . It is a MS Windows application built with Borland C++ Builder and allows performing the basic tasks required in a MOST system. We can learn how to use the NetServices Layer 1 API from this example application. Please refer to [2] to learn more about NSL1_EX. In the first part of this section, we will introduce some basic original functions of NSL1_EX which are used for implementing our multimedia applications. The C++ codes and data structures of the messages will be presented. In the second part, enhanced features of the program will be shown, e.g. controlling of the prototyping MOST devices. The used functions and parameters will be listed.

4.3.1 NetServices Layer 1 Example (NSL1_EX)

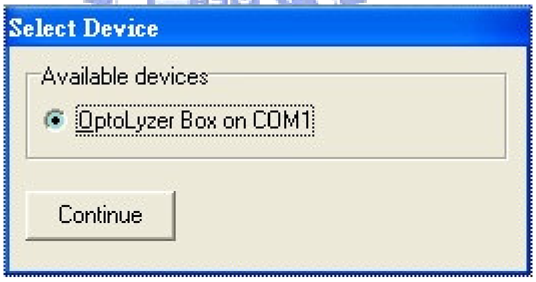
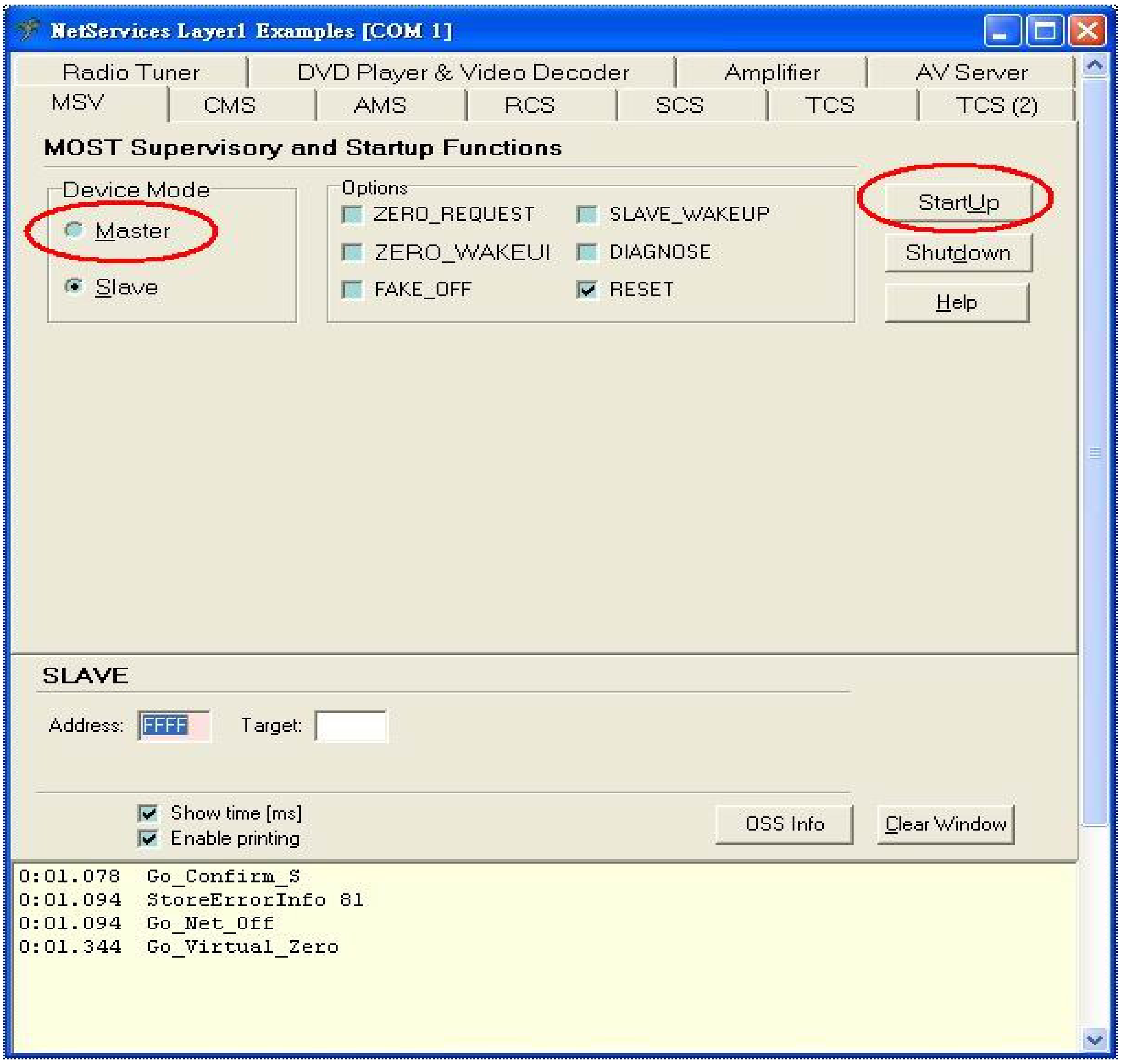


Figure 4.20 Select Device for NSL1_EX

Connection to the MOST network can be achieved by the OptoLyzer4MOST Interface Box, or MOST PCI Board. In our case as shown in Figure 4.20, only OptoLyzer Box is available.



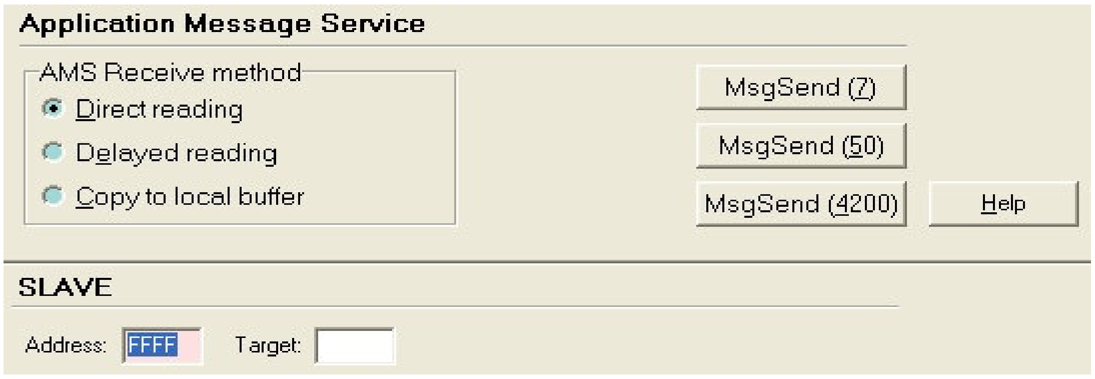
```
0:01.078   Go_Confirm_S
0:01.094   StoreErrorInfo 81
0:01.094   Go_Net_Off
0:01.344   Go_Virtual_Zero
```

Figure 4.21 Main Window of NSL1_EX

If connection with the device is successful, then the main window of the program appears, as shown in Figure 4.21. In the upper area is a tab page that different pages of services can be easily switched. MSV is the default page in which we can set the device as Timing Master and hence start up the MOST network.

In the lower area there are several edit fields and a message list. The edit fields can be used to receive numeric input as parameters for control messages on runtime, and the message list displays all the messages generated by NetServices (Callback functions).

4.3.1.1 AMS



Address: FFFF

Direct reading [✓]

Figure 4.22 AMS Screenshot

Most of the tasks in our multimedia applications are implemented by AMS (Application Message Services); however, in the original NSL1_EX as shown in Figure 4.22, the messages of the AMS example are fixed. The numbers in the brackets indicate the length (in bytes) of a testing message to be sent via AMS. Therefore, the transmission of AMS can be tested, but the command may be invalid.

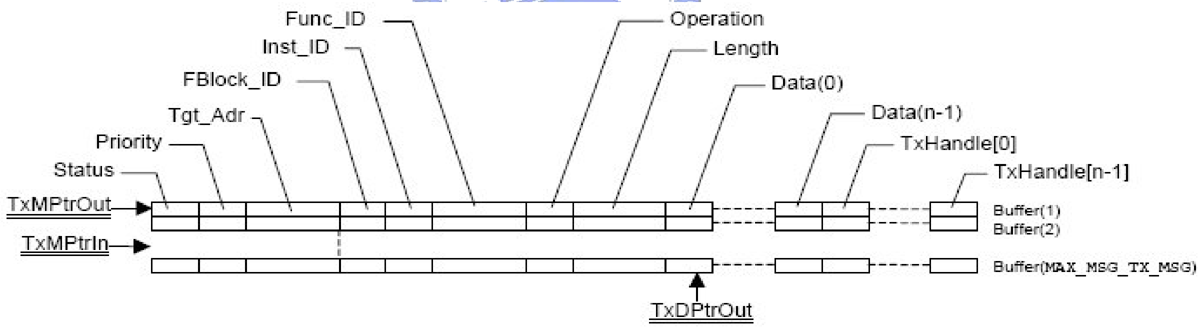


Figure 4.23 AMS TX buffers [2]

Figure 4.23 depicts the structure of AMS TX buffers in which messages are stored. Access is handled via pointers to the buffers. For the convenience of using AMS, the function *MsgSend1Example* is modified into another function with arguments, named *AmsSend*.

```
void AmsSend(word Target,byte Fb,word Fid,byte Op,int Length,char Data[])
{
    struct Msg_Tx_Type   *myMsgPtr;

    // MOSTNetServices: check if there is a free buffer entry
    LOCK_ACCESS;
    if ( MsgCheckTxBuffer() == 0 )
    {
        UNLOCK_ACCESS;
          return;
    }

     // MOSTNetServices: get a free buffer entry
     myMsgPtr = MsgGetTxPtr();

     if (myMsgPtr != NIL)
     {
          // store the message in the data structure Msg_Tx_Type
          myMsgPtr->Tgt_Adr      = Target;
          myMsgPtr->Inst_ID      = 0x00;
          myMsgPtr->FBlock_ID    = Fb;
          myMsgPtr->Func_ID      = Fid;
          myMsgPtr->Operation    = Op;
          myMsgPtr->Length       = Length; // specify the length of data

          // read data array of the length specified
          for (int i=0;i<=Length;i++) myMsgPtr->Data[i]=Data[i];

          // releases a message for being sent
          MsgSend(myMsgPtr);
     }
    UNLOCK_ACCESS;

    return;
}
```

Figure 4.24 Code of the function "AmsSend"

For example, if we want the DVDPlayer4MOST(0x198) to start playing the disc, the following message should be sent (referring to the manual of the device):

DVDVideoPlayer(0x34) . DeckStatus(0x200) . Set(0x00) . Play(0x00)

By substituting into the complete protocol:

DeviceID . FBlockID . InstID . FktID . OPType . Length (Data)

we have:

DVDPlayer4MOST . DVDVideoPlayer . 00 . Deckstatus . Set . 1 (Play)

0x198 . 0x34 . 0x00 . 0x200 . 0x00 . 1 (0x00)

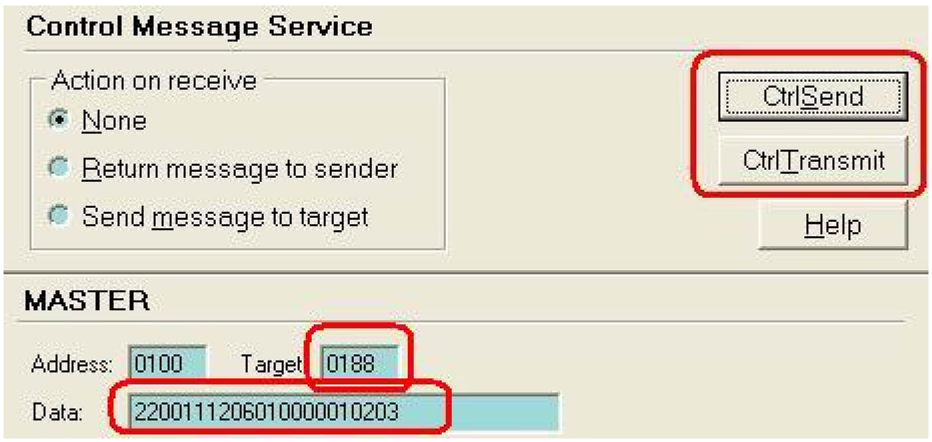
Then the function code is simply as follows:

(Please note that Inst_ID is set to 0x00 in *AmsSend* and therefore not given as a parameter when calling *AmsSend*.)

```
char data[]={0x00};
AmsSend(0x198,0x34,0x200,0x00,1,data);
```

Figure 4.25 An example code for using "AmsSend"

4.3.1.2 CMS


Control Message Service

Action on receive
- [✓ None]
- Return message to sender
- Send message to target

MASTER

Address: 0100 Target 0188
Data: 2200111206010000010203

Figure 4.26 CMS Screenshot

We know that AMS, based on CMS (Control Message Service), organizes the messages on a higher level and is friendlier to use. However, the original NSL1_EX has a CMS user interface (UI) in which we can enter a control message to be sent via CMS. Some of the tasks can also be done in this way. As shown in Figure 4.26, there are two different methods for sending messages via CMS. The “CtrlSend” button uses *CtrlGetTxPtr* and *CtrlSend*, while the “CtrlTransmit” button uses *CtrlTransmit*. Both methods are workable. Before explaining the usage of the UI, we should view the structure of CMS TX buffers, shown in Figure 4.27.

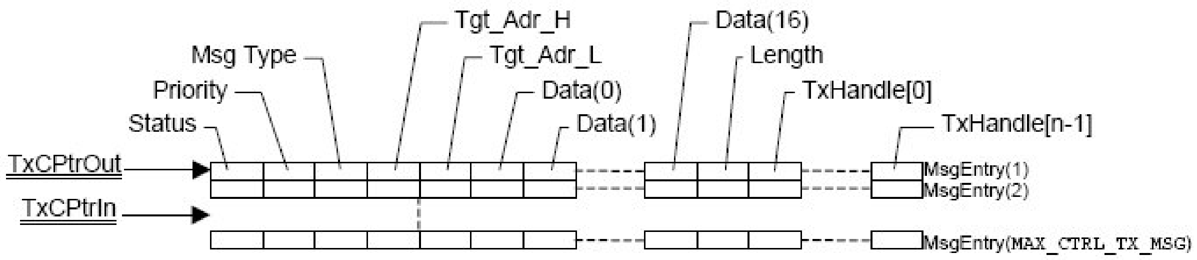
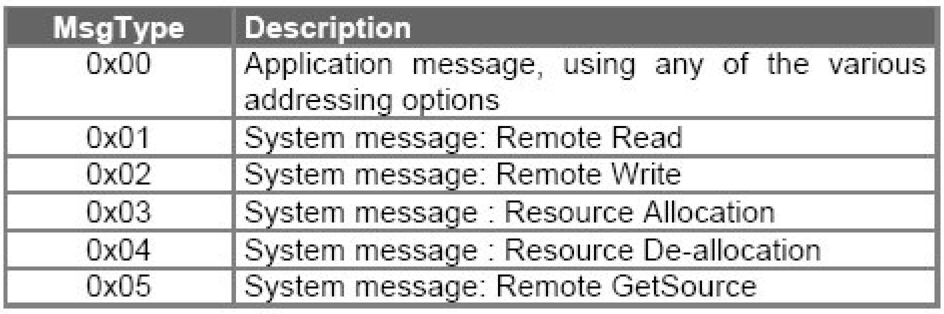


Figure 4.27 CMS TX buffers [2]

Table 4.2 Type of message to be sent via CMS



MsgType	Description
0x00	Application message, using any of the various addressing options
0x01	System message: Remote Read
0x02	System message: Remote Write
0x03	System message : Resource Allocation
0x04	System message : Resource De-allocation
0x05	System message: Remote GetSource

Different from the case of AMS, there is an essential element “Msg Type” which contains the type of the message to be sent, as described in Table 4.2. System messages are usually transmitted by the higher service. More information can be found in [2].

Besides, FBlock_ID, Inst_ID, Func_ID and Operation are not particularly defined in the structure for CMS buffers. In fact, they are included in the Data array which has a limited length of 17 bytes. Tgt_Adr (word) is also divided into Tgt_Adr_H (byte) and Tgt_Adr_L (byte). As a result, in Figure 4.26, we have input fields of “Address”, “Target” and “Data”; however, the “Address” here is corresponding to the address of the device controlled by the program and can be set to other values. “Target” is target address for sending data and “Data” corresponds to the Data array of the data structure. In the example of Figure 4.26, ”0188” and “22001112060100000010203” are filled in “Target” and “Data”. Then, this message can be sent by clicking on the “CtrlSend” or the “CtrlTransmit” button.

This control message corresponds to:

Amplifier4MOST . AudioAmplifier . 0x00 . Connect . StartResult . 6 (SinkNr, SrcDelay, [Channels])

➔

0x188 . 0x22 . 0x00 . 0x111 . 0x02 . 6 (0x01,0x01,[0x00,0x01,0x02,0x03])

Under this command, the Amplifier4MOST shall connect its data ports to the MOST synchronous channels 0x00…0x03.

```
edtTarget->Text="0188";
SetTarget(); // Call the example function to set target address
edtMsg->Text="22001112060100000010203";
CtrlSendExample(); // Call the example function for sending the message
```

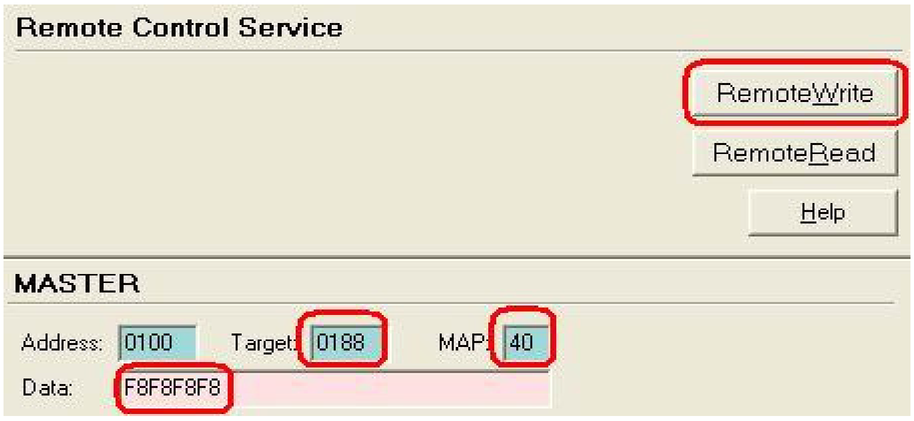
Figure 4.28 An example code for using “CtrlSendExample”

The simplest way of implementing this task by a function is to specify the values of both input fields, and then, to call the CMS example function for sending messages.

4.3.1.3 RCS

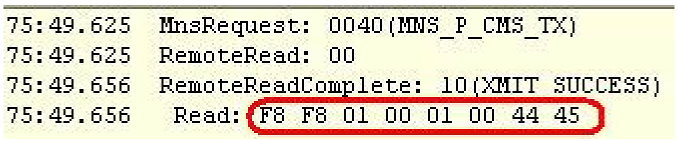
RCS (Remote Control Service) provides the sending of Remote System Messages. It is subdivided into the services “RemoteWrite” and “RemoteRead”, as the UI shown in Figure 4.29. The input field “Target” specifies the target address for sending data, and the “MAP” denotes “Memory Address Pointer” which gives the memory address in the target device.

"Data" gives the values to write to the memory address.



Address: 0100 Target: 0188 MAP: 40
Data: F8F8F8F8

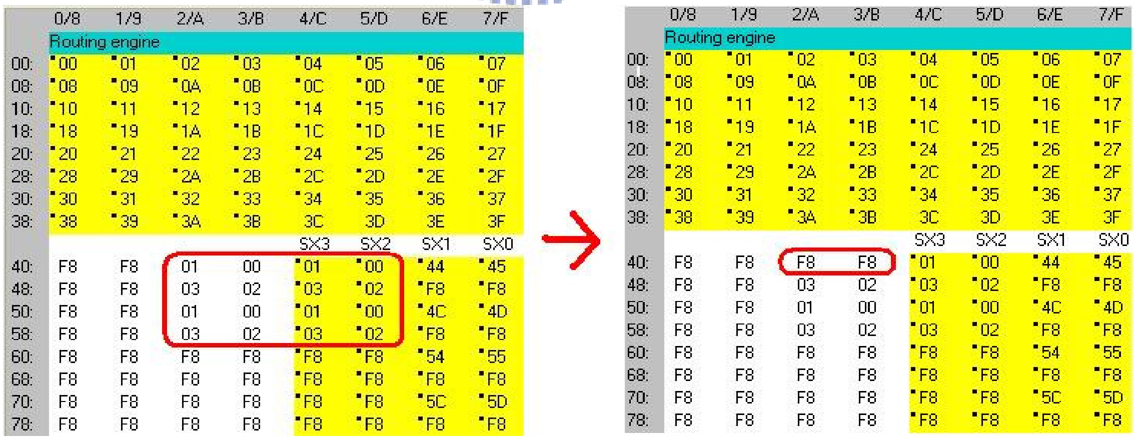
Figure 4.29 RCS Screenshot



```
75:49.625  MnsRequest: 0040(MNS_P_CMS_TX)
75:49.625  RemoteRead: 00
75:49.656  RemoteReadComplete: 10(XMIT SUCCESS)
75:49.656   Read: F8 F8 01 00 01 00 44 45
```

Figure 4.30 Perform "RemoteRead" of RCS

With the settings shown in Figure 4.29, click the "RemoteRead" button, we can get the information printed on the message list, as shown in Figure 4.30. It is continued with the last command in Section 4.3.1.2 (connecting the Amplifier4MOST to channels 0x00…0x03). With the help of MOST Radar, we can check the register mapping, as shown in Figure 4.31.



	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
	Routing engine							
00:	00	01	02	03	04	05	06	07
08:	08	09	0A	0B	0C	0D	0E	0F
10:	10	11	12	13	14	15	16	17
18:	18	19	1A	1B	1C	1D	1E	1F
20:	20	21	22	23	24	25	26	27
28:	28	29	2A	2B	2C	2D	2E	2F
30:	30	31	32	33	34	35	36	37
38:	38	39	3A	3B	3C	3D	3E	3F
					SX3	SX2	SX1	SX0
40:	F8	F8	01	00	01	00	44	45
48:	F8	F8	03	02	03	02	F8	F8
50:	F8	F8	01	00	01	00	4C	4D
58:	F8	F8	03	02	03	02	F8	F8
60:	F8	F8	F8	F8	F8	F8	54	55
68:	F8	F8	F8	F8	F8	F8	F8	F8
70:	F8	F8	F8	F8	F8	F8	5C	5D
78:	F8	F8	F8	F8	F8	F8	F8	F8

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
	Routing engine							
00:	00	01	02	03	04	05	06	07
08:	08	09	0A	0B	0C	0D	0E	0F
10:	10	11	12	13	14	15	16	17
18:	18	19	1A	1B	1C	1D	1E	1F
20:	20	21	22	23	24	25	26	27
28:	28	29	2A	2B	2C	2D	2E	2F
30:	30	31	32	33	34	35	36	37
38:	38	39	3A	3B	3C	3D	3E	3F
					SX3	SX2	SX1	SX0
40:	F8	F8	F8	F8	01	00	44	45
48:	F8	F8	03	02	03	02	F8	F8
50:	F8	F8	01	00	01	00	4C	4D
58:	F8	F8	03	02	03	02	F8	F8
60:	F8	F8	F8	F8	F8	F8	54	55
68:	F8	F8	F8	F8	F8	F8	F8	F8
70:	F8	F8	F8	F8	F8	F8	5C	5D
78:	F8	F8	F8	F8	F8	F8	F8	F8

Figure 4.31 Register mapping of MOST Transceiver on Amplifier4MOST

Compared with Figure 4.31 (the left side), it is clear that 8 bytes from 0x40 are read as displayed in the last line of Figure 4.30. Similarly, if "RemoteWrite" are activated with the settings in Figure 4.29, the values of the target registers will be replaced by the given data. As a result, the register mapping turns into the right side of Figure 4.31. With "RemoteRead" service, it is possible to implement a register editor UI by collecting contents of the registers, while their values can be modified via "RemoteWrite" service.

```
edtTarget->Text="3048";
SetTarget();   // Call the example function to set targeet address
edtMAP->Text="00";
SetMAP();   // Call the example function to set targeet address
edtMsg->Text="4445464740414243";
RemoteWriteExample();   // Call the example function for sending the message
```

Figure 4.32 An example code for using "RemoteWriteExample"

In our application, RCS is used to edit the Routing Engine of MOST PCI Board. To use RCS for our purpose, it is as simple as the case of CMS. By specifying the Target, MAP, and Data, as well as calling *RemoteWriteExample* in the end, the tasks can be done in a function call.

```
#ifdef RCS_1
void RemoteWriteExample(void)
{              // send the first 8 bytes from data input field to the target at address map
    word tgtAdr;
    byte map;
    byte buf[8];
    byte numBytes;
    byte status;
    tgtAdr   = targetAddress;            // target address of the remote chip
    map      = mapAddress;               // memory address pointer on remote chip
    // convert the contents of the input field and fill the buffer
    numBytes = FrmtAnsiToHex(&MyDataStr, &buf[0], 8);
    // MOSTNetServices: write to the target
    LOCK_ACCESS;
    status = RemoteWrite(tgtAdr, map, &buf[0], numBytes);
    UNLOCK_ACCESS;

    PrintError("RemoteWrite", status);
}
```

Figure 4.33 Code of the function "RemoteWriteExample"

At the end of this section, code of the function *RemoteWriteExample* is given in Figure 4.33. Called by clicking the "RemoteWrite" button, the function sends the first 8 bytes from the "Data" input field to the target at address map. Writing to the target is done by the NetServices Library function *RemoteWrite*.

4.3.2 Enhancement of the Program

In this section, the second part of Section 4.3, we will introduce several control programs created by ourselves. Attached to the example application NSL1_EX, our control programs are used for sending application messages to the respective MOST devices, including Radio Tuner, DVD Player, Video Decoder, Amplifier, and the MOST PCI Board. With these enhanced features, the NSL1_EX becomes practical for multimedia control. Different multimedia applications will be introduced in Section 4.4. We wish to provide a comfortable way to operate the MOST devices in our multimedia applications; hence, the user interfaces are designed for touch panel displays with a resolution of 640 x 480 pixels.

Please note: InstID is set to 0x00 in our programs (since every device in the MOST ring is unique); hence, it is omitted in the following protocol mapping tables.

4.3.2.1 Radio Tuner

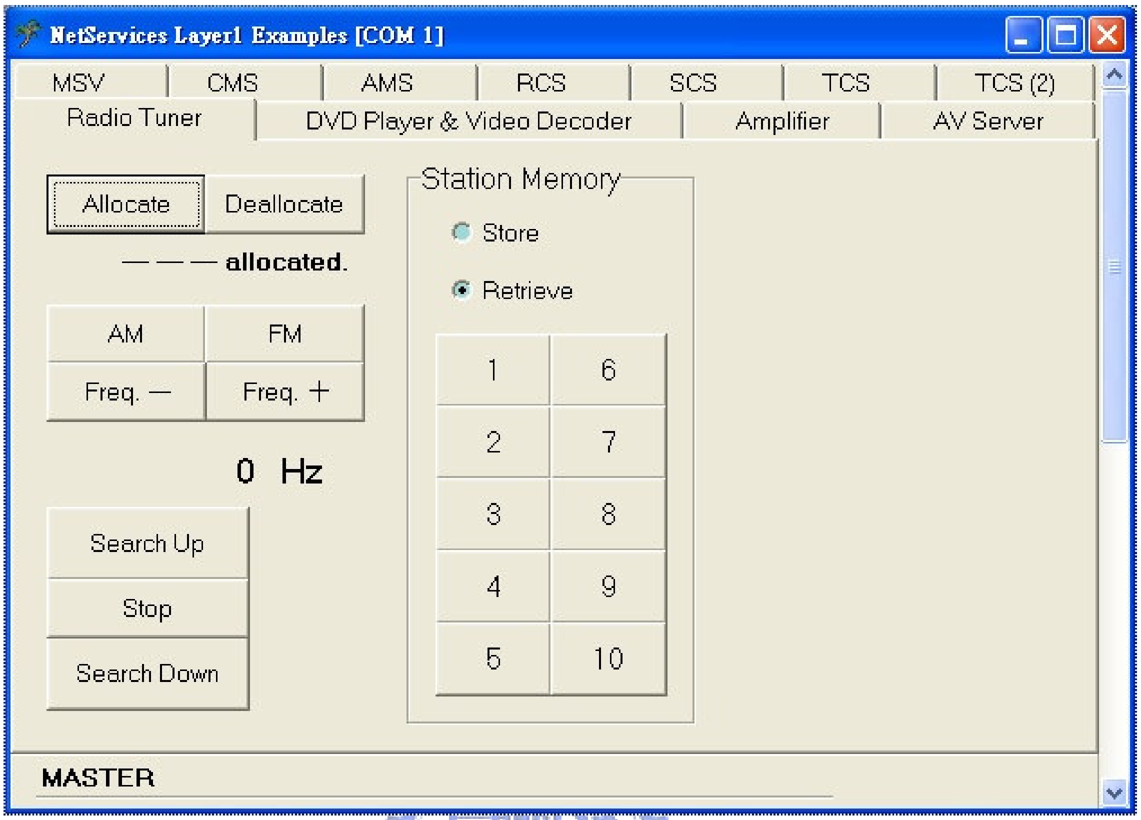


Figure 4.34 RadioTuner4MOST Control Program

Figure 4.34 shows the "Radio Tuner" page of the program. The controlled device is RadioTuner4MOST (DeviceID = 0x180), an audio source in the MOST ring. The "Allocate" function is necessary for the source node to put real-time data on the MOST network. The allocated channels by the function can be detected and displayed in a label in this program. As well as a normal AM/FM radio tuner, manual frequency stepping and automatic scanning are available. Current frequency is also detectable and displayed. On the right side, there are 10 buttons representing the EEPROM station memory of the device. Storing and retrieving stations from the station memory can be switched by the radio buttons.

Table 4.3 Protocol Mapping of RadioTuner4MOST Control Program [22]

(DeviceID = 0x180)

Button	FBlockID	FktID	OPType	Length	Data
Allocate	0x40 (AmFmTuner)	0x101 (Allocate)	0x02 (StartResult)	1	0x01 (SourceNr)
Deallocate		0x102 (DeAllocate)	0x02 (StartResult)	1	0x01 (SourceNr)
AM		0x200 (ATWaveband)	0x02 (SetGet)	3	0x0101,0x02 (Pos,AM)
FM					0x0101,0x01 (Pos,FM)
Freq. －		0x206 (ATFrequency)	0x04 (Decrement)	1	0x01 (NSteps)
Freq. ＋			0x03 (Increment)	1	0x01 (NSteps)
Search Up		0x204 (ATSeek)	0x00 (Set)	1	0x01 (Up Auto)
Stop					0x00 (Off)
Search Down					0x07 (Down Auto)
Store [0]		0x400 (ATPresetSave)	0x00 (Start)	3	0x0401,0x00 (FktID,PresetNumber)
Store [10]					0x0401,0x0A (FktID,PresetNumber)
Retrieve [0]		0x401 (ATPresetList1)	0x00 (Set)	3	0x00,0x0101 (Pos,PresetSelection)
Retrieve [10]					0x0A,0x0101 (Pos,PresetSelection)

Most functions in the control program are implemented by AMS; therefore, each button can be mapped to the MOST protocol as listed in Table 4.3. Please note that for “Store” as well as “Retrieve”, the number in the bracket represents the station of the memory preset list, and the station is to be accessed (by clicking on the respective button).

4.3.2.2 DVD Player

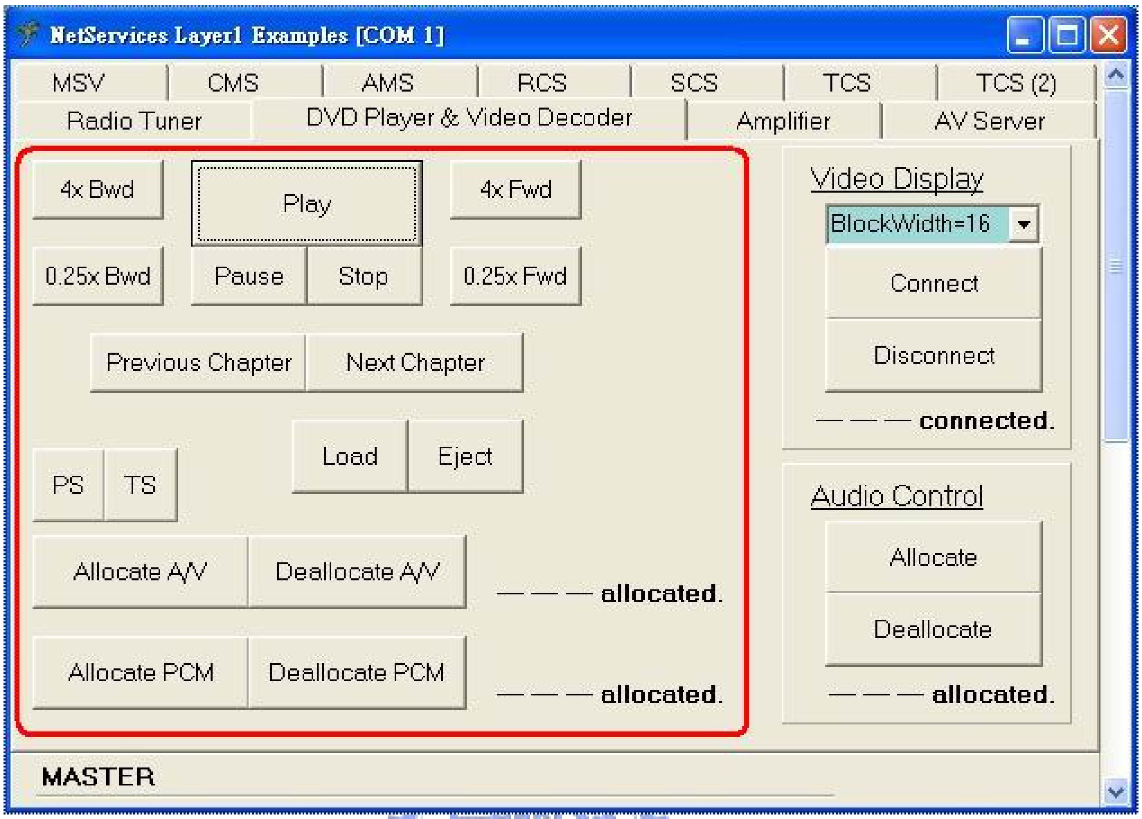


Figure 4.35 DVDPlayer4MOST Control Program

Figure 4.35 shows the "DVD Player & Video Decoder". On the left side is the region for DVD control. The controlled device is DVDPlayer4MOST (DeviceID = 0x198) which can generate both synchronous audio stream (PCM) and A/V mixed stream (MPEG-2) for the MOST network. The generated MPEG-2 stream can be Program Stream (PS) or Transport Stream (TS). In general, BlockWidth of the A/V stream is set to 16 (bytes). The other buttons, as well as a normal DVD player, implement functions for playing the DVD or an audio CD. For a DVD movie, it is able to perform "search forward", "search backward", "slow motion forward", or "slow motion backward".

The following Table 4.4 and Table 4.5 represent protocol mapping of the controls. Please note that "4x Bwd", "4xFwd", "0.25x Bwd" and "0.25x Fwd" both comprise two actions which are "setting search speed" and "performing the search".

Table 4.4 Protocol Mapping of DVDPlayer4MOST Control Program [23]

(DeviceID = 0x198)

Button	FBlockID	FktID	OPType	Length	Data
Play	0x34 (DVDVideoPlayer)	0x200 (DeckStatus)	0x00 (Set)	1	0x00 (Play)
Pause					0x02 (Pause)
Stop					0x01 (Stop)
4x Bwd (Search Backward)					0x06 (Search Backward)
4x Fwd (Search Forward)					0x05 (Search Forward)
0.25x Bwd (Slow Motion Backward)					0x21 (Slow Motion Backward)
0.25x Fwd (Slow Motion Forward)					0x20 (Slow Motion Forward)
Load					0x03 (Load)
Eject					0x04 (Unload)
4x Bwd (Speed)		0x458 (FastBwSpeed)	0x00 (Set)	1	0x04 (4 * normal speed)
4x Fwd (Speed)		0x457 (FastFwSpeed)	0x00 (Set)	1	0x04 (4 * normal speed)
0.25x Bwd (Speed)		0x456 (SlowBwSpeed)	0x00 (Set)	1	0x04 (1/4 of normal speed)

Table 4.5 Protocol Mapping of DVDPlayer4MOST Control Program (continued) [23]

(DeviceID = 0x198)

Button	FBlockID	FktID	OPType	Length	Data
0.25x Bwd (Speed)	0x34 (DVDVideoPlayer)	0x456 (SlowBwSpeed)	0x00 (Set)	1	0x04 (1/4 of normal speed)
0.25x Fwd (Speed)	0x34 (DVDVideoPlayer)	0x455 (SlowFwSpeed)	0x00 (Set)	1	0x04 (1/4 of normal speed)
Previous Chapter	0x34 (DVDVideoPlayer)	0x251 (VideoInteraction)	0x00 (Processing)	1	0x08 (Previous Chapter)
Next Chapter	0x34 (DVDVideoPlayer)	0x251 (VideoInteraction)	0x00 (Processing)	1	0x09 (Next Chapter)
PS	0x34 (DVDVideoPlayer)	0x100 (SourceInfo)	0x00 (Set)	3	0x02,0x21,0x10 (SourceNr, MPEG2 Program Stream, BlockWidth)
TS	0x34 (DVDVideoPlayer)	0x100 (SourceInfo)	0x00 (Set)	3	0x02,0x22,0x10 (SourceNr, MPEG2 Transport Stream, BlockWidth)
Allocate A/V	0x34 (DVDVideoPlayer)	0x101 (Allocate)	0x02 (StartResult)	1	0x02 (SourceNr)
Allocate PCM	0x34 (DVDVideoPlayer)	0x101 (Allocate)	0x02 (StartResult)	1	0x01 (SourceNr)
Deallocate A/V	0x34 (DVDVideoPlayer)	0x102 (DeAllocate)	0x02 (StartResult)	1	0x02 (SourceNr)
Deallocate PCM	0x34 (DVDVideoPlayer)	0x102 (DeAllocate)	0x02 (StartResult)	1	0x01 (SourceNr)

4.3.2.3 Video Decoder

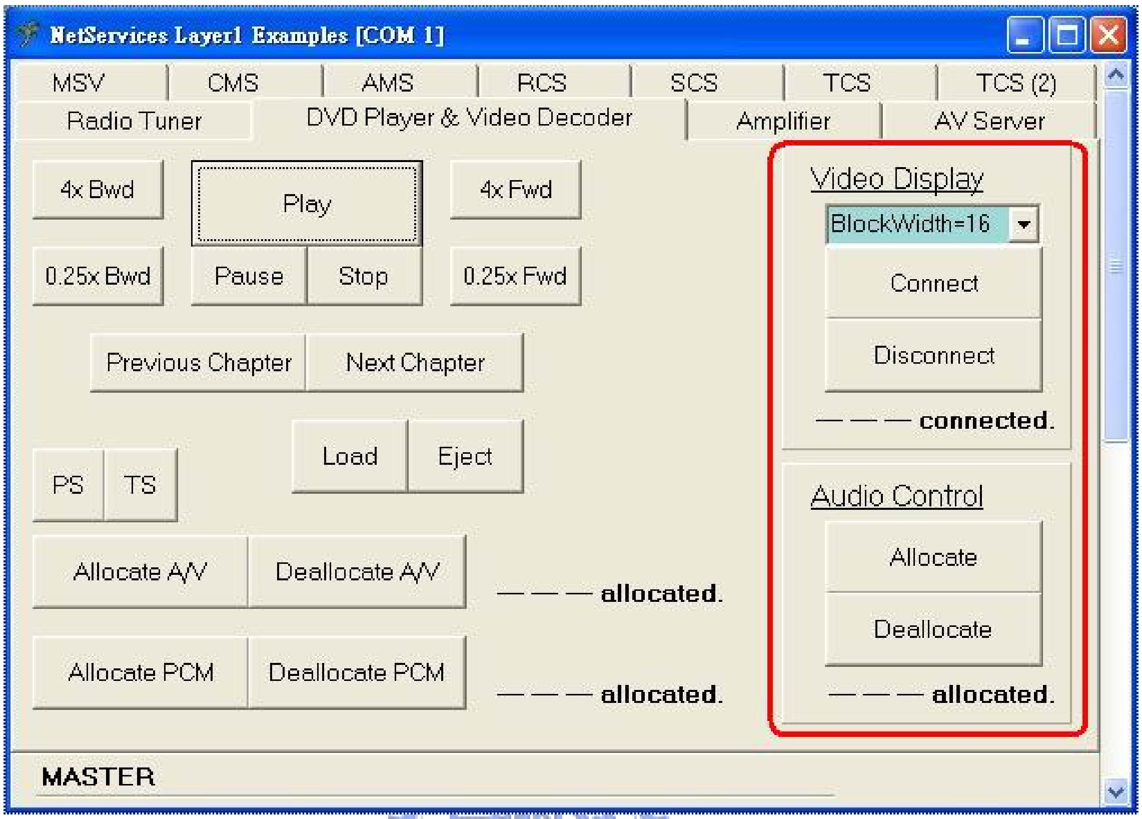


Figure 4.36 VideoDecoder4MOST Control Program

As shown on the right side of Figure 4.36, the control of VideoDecoder4MOST (DeviceID = 0x200) is divided into two parts. “Video Display” is implemented by the functions of the FBlock “GraphicDisplay”, and “Audio Control” is implemented by the functions of “AuxIn”. In “Video Display”, as a sink node, the device can be connected to the synchronous channels on the MOST network for MPEG2 reception. BlockWidth of the A/V stream could be 4 to 16 bytes in step of 4 bytes. In “Audio Control”, audio data decoded from the A/V stream can be fed into the MOST network via the allocated synchronous channels.

Table 4.6 Protocol Mapping of VideoDecoder4MOST Control Program [13]

(DeviceID = 0x200)

Button	FBlockID	FktID	OPType	Length	Data
Connect (BlockWidth=16)	0x60 (GraphicDisplay)	0x111 (Connect)	0x02 (StartResult)	18	0x01,0x00, 0x00…0x0F (SinkNr,SrcDelay, ChannelList)
Connect (BlockWidth=12)				14	0x01,0x00, 0x00…0x0B (SinkNr,SrcDelay, ChannelList)
Connect (BlockWidth= 8)				10	0x01,0x00, 0x00…0x07 (SinkNr,SrcDelay, ChannelList)
Connect (BlockWidth= 4)				6	0x01,0x00, 0x00…0x03 (SinkNr,SrcDelay, ChannelList)
Disconnect		0x112 (DisConnect)	0x02 (StartResult)	1	0x01 (SinkNr)
Allocate	0x24 (AuxIn)	0x101 (Allocate)	0x02 (StartResult)	1	0x01 (SourceNr)
Deallocate		0x102 (DeAllocate)	0x02 (StartResult)	1	0x01 (SourceNr)

Protocol mapping of the Video Decoder control is listed in Table 4.6. For “Connect” method, the “ChannelList” needs to be given as a part of Data. As a result, for connecting to a 16-BlockWidth source, the Data is up to 18 bytes long, which will result in 2 CMS messages (organized by AMS). Please note that the channel offset is assumed zero. In other words, the ChannelList for “Connect” always starts from 0x00.

4.3.2.4 Amplifier

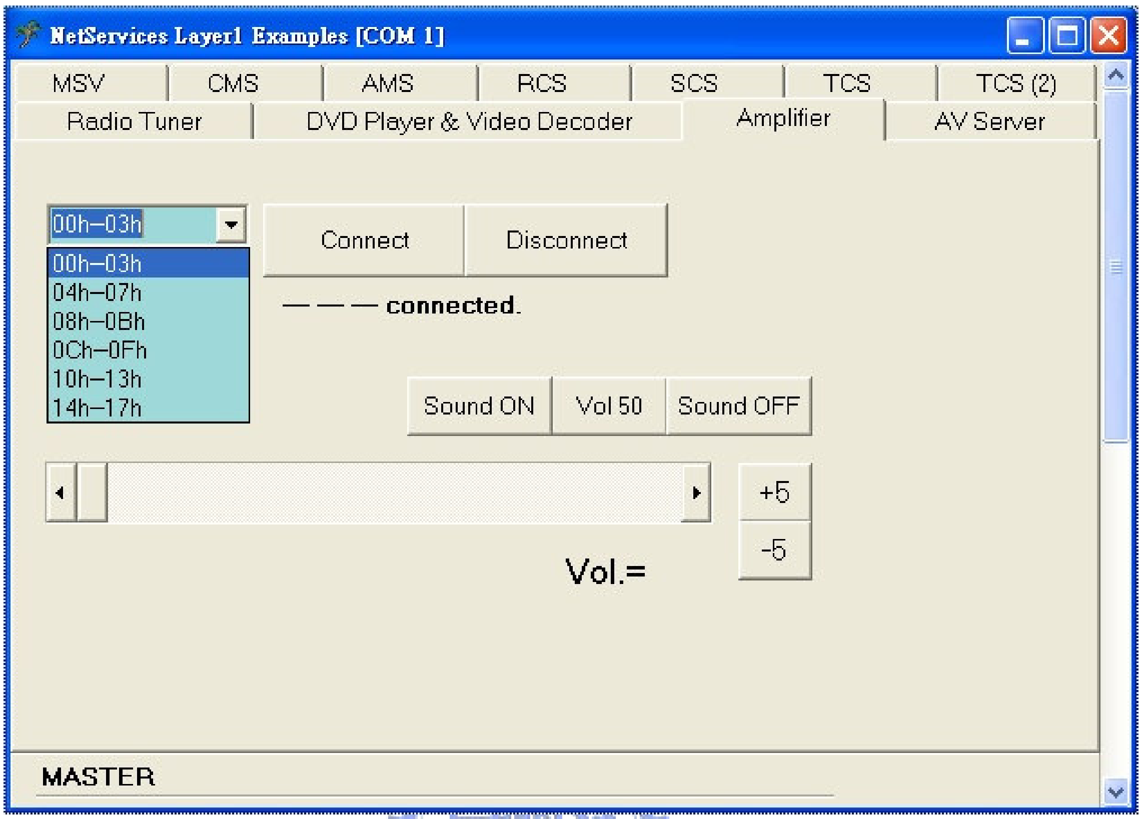


Figure 4.37 Amplifier4MOST Control Program

Figure 4.37 shows the "Amplifier" page in which Amplifier4MOST (DeviceID = 0x188) can be controlled. Mostly, the audio stream has a BlockWidth of 4. In the combo box as shown, we can select the synchronous channels (in step of 4 bytes) for the device to connect. Before we can hear the sound, the "Sound ON" needs to be activated. Besides, the "Vol 50" is used to set the volume to the center value in a quick manner. We put a scroll bar to help adjust the volume more easily. Position of the scroll bar (within the range of 0 to 100) corresponds to the current volume and affects it simultaneously. The current volume is also displayed in the label below the scroll bar.

Table 4.7 Protocol Mapping of Amplifier4MOST Control Program [24]

(DeviceID = 0x188)

Button	FBlockID	FktID	OPType	Length	Data
Connect (00h--03h)	0x22 (AudioAmplifier)	0x111 (Connect)	0x02 (StartResult)	6	0x01,0x01, 0x00…0x03 (SinkNr,SrcDelay, ChannelList)
Connect (10h--13h)					0x01,0x01, 0x10…0x13 (SinkNr,SrcDelay, ChannelList)
Disconnect		0x112 (DisConnect)	0x02 (StartResult)	1	0x01 (SinkNr)
Sound ON		0x113 (Mute)	0x02 (SetGet)	2	0x01,0x00 (SinkNr, MUTE OFF)
Sound OFF					0x01,0x01 (SinkNr, MUTE ON)
Vol 50		0x400 (Volume)	0x00 (Set)	1	0x32 (Volume)
+5 ("+1" * 5)			0x03 (Increment)	1	0x01 (NSteps)
-5 ("-1" * 5)			0x04 (Decrement)	1	0x01 (NSteps)

Table 4.7 represents protocol mapping of the Amplifier control program. The methods "Connect" and "DisConnect" of the FBlock "AudioAmplifier" have the same usage as those of the FBlock "GraphicDisplay". In this table, only connecting to "00h--03h" and "10h--13h" are taken for examples. Please note that the "+5" button (similar to "-5") is implemented by repeating "Increment" (+1) five times, since one "Increment" command can only affect the volume by 1.

4.3.2.5 MOST PCI Board

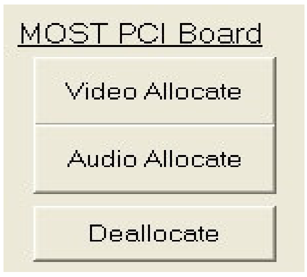
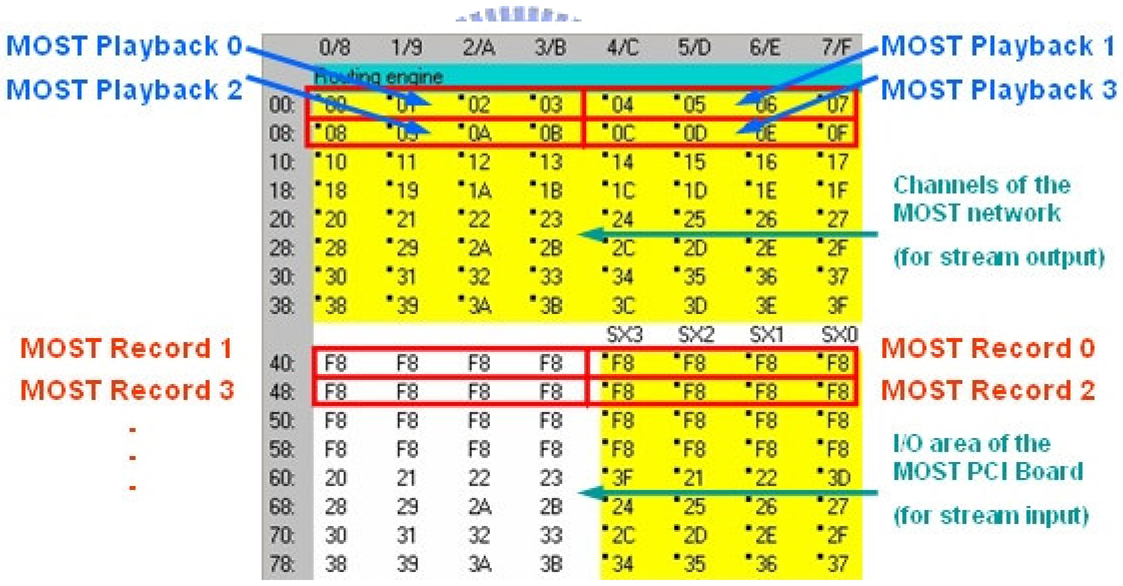


Figure 4.38 MOST PCI Board Control Program

As shown in Figure 4.38, the control program of MOST PCI Board consists of three buttons: “Video Allocate”, “Audio Allocate”, and “Deallocate”. In this program, RCS is used to edit Routing Engine (RE) of the MOST PCI Board. With a click on the button, the RE is modified for allocation uses.



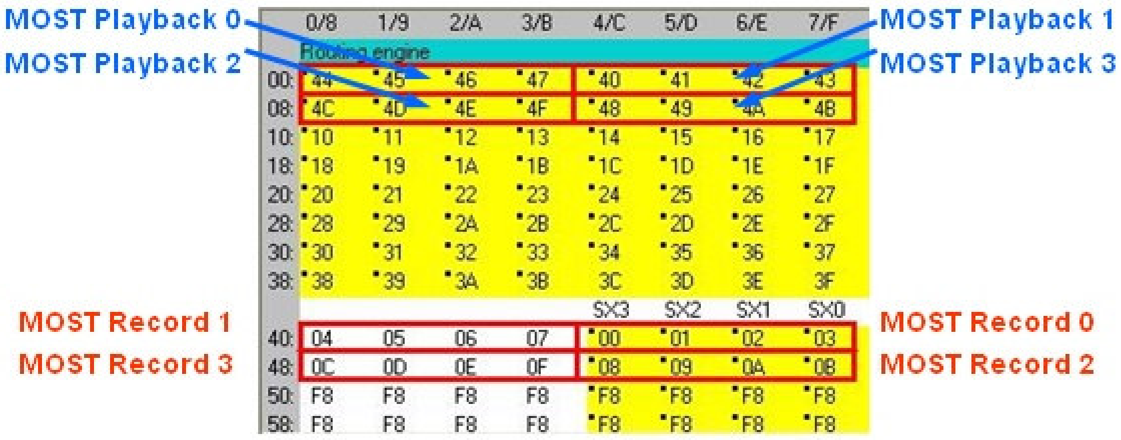
Routing engine

	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
00:	•00	•01	•02	•03	•04	•05	•06	•07
08:	•08	•09	•0A	•0B	•0C	•0D	•0E	•0F
10:	•10	•11	•12	•13	•14	•15	•16	•17
18:	•18	•19	•1A	•1B	•1C	•1D	•1E	•1F
20:	•20	•21	•22	•23	•24	•25	•26	•27
28:	•28	•29	•2A	•2B	•2C	•2D	•2E	•2F
30:	•30	•31	•32	•33	•34	•35	•36	•37
38:	•38	•39	•3A	•3B	3C	3D	3E	3F
					SX3	SX2	SX1	SX0
40:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
48:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
50:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
58:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
60:	20	21	22	23	•3F	•21	•22	•3D
68:	28	29	2A	2B	•24	•25	•26	•27
70:	30	31	32	33	•2C	•2D	•2E	•2F
78:	38	39	3A	3B	•34	•35	•36	•37

Figure 4.39 Routing Engine of MOST PCI Board [19]

At first, let’s take a look at Figure 4.39 which explains the routing scheme. “MOST Playback” and “MOST Record” stand for the I/O area of the MOST PCI Board. Addresses of “MOST Playback” should be written into the registers representing MOST channels in the upper area for stream output, while addresses of the registers representing MOST channels should be written into “MOST Record” in the lower area for stream input.

By clicking on the “Video Allocate” button, both output and input areas are modified, as shown in Figure 4.40.



Routing engine

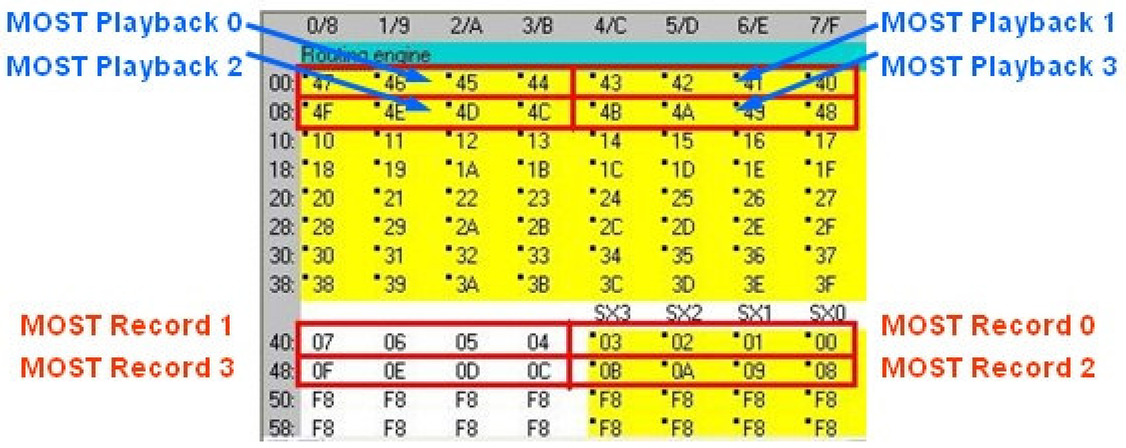
	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
00:	•44	•45	•46	•47	•40	•41	•42	•43
08:	•4C	•4D	•4E	•4F	•48	•49	•4A	•4B
10:	•10	•11	•12	•13	•14	•15	•16	•17
18:	•18	•19	•1A	•1B	•1C	•1D	•1E	•1F
20:	•20	•21	•22	•23	•24	•25	•26	•27
28:	•28	•29	•2A	•2B	•2C	•2D	•2E	•2F
30:	•30	•31	•32	•33	•34	•35	•36	•37
38:	•38	•39	•3A	•3B	3C	3D	3E	3F
					SX3	SX2	SX1	SX0
40:	04	05	06	07	•00	•01	•02	•03
48:	0C	0D	0E	0F	•08	•09	•0A	•0B
50:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
58:	F8	F8	F8	F8	•F8	•F8	•F8	•F8

Figure 4.40 Routing Engine for Video Allocation of MOST PCI Board

For A/V streams output to the MOST network, the register settings would route the source data outputs “MOST Playback 0…3” to the MOST channels 0x00...0x0F.

For A/V streams input from the MOST network, the register settings would route the MOST channels 0x00...0x0F to the source data inputs “MOST Record 0...3”.

Since the MOST channels 0x00...0x0F are simultaneously connected by the output data ports of the MOST PCI Board and routed to the input data ports of the MOST PCI Board, it is possible to transmit and receive the A/V streams both via the MOST PCI Board and MOST channels 0x00...0x0F at the same time! But generally in our applications, either Playback or Record of the MOST PCI Board is performed. Video uses and audio uses are also separated.



	0/8	1/9	2/A	3/B	4/C	5/D	6/E	7/F
00:	•47	•46	•45	•44	•43	•42	•41	•40
08:	•4F	•4E	•4D	•4C	•4B	•4A	•49	•48
10:	•10	•11	•12	•13	•14	•15	•16	•17
18:	•18	•19	•1A	•1B	•1C	•1D	•1E	•1F
20:	•20	•21	•22	•23	•24	•25	•26	•27
28:	•28	•29	•2A	•2B	•2C	•2D	•2E	•2F
30:	•30	•31	•32	•33	•34	•35	•36	•37
38:	•38	•39	•3A	•3B	3C	3D	3E	3F
					SX3	SX2	SX1	SX0
40:	07	06	05	04	•03	•02	•01	•00
48:	0F	0E	0D	0C	•0B	•0A	•09	•08
50:	F8	F8	F8	F8	•F8	•F8	•F8	•F8
58:	F8	F8	F8	F8	•F8	•F8	•F8	•F8

Figure 4.41 Routing Engine for Audio Allocation of MOST PCI Board

For audio uses, a click the “Audio Allocate” button will result in the following RE settings, as shown in Figure 4.41. Different from the video channels allocation, the bytes in a quadlet (= 4 bytes) are reversed, for both transmitting and receiving schemes. Here the “MOST Playback 0…3” and the “MOST Record 0...3” can be viewed as the “MOST Audio Channel 0...3” (mentioned in Section 4.2.3). For example, to route a 16bit wave output channel (which requires 4 bytes) to the MOST network, the values 0x47…0x44 (corresponding to “MOST Audio Channel 0”) are written into the registers 0x00…0x03. To route a 16bit Wave channel from the MOST network to the second Wave input channel (MOST Audio Channel 1), the according addresses of the MOST channels (e.g. 0x04...0x07) should be written into the bytes 0x43 down to 0x40 of the lower routing area.

As to the button “Deallocate”, it simply deactivates the routing by resetting the values of the registers to their initial states, as those in Figure 4.39.

4.4 Multimedia Applications (Step-by-Step Instructions)

In the applications below, it is assumed that the MOST network is ON (whether it is started by NSL1_EX or MICtrl); furthermore, if communication with the MOST PCI Board is required, the MICtrl process should be running.

4.4.1 Application 1: Listen to the Radio

Source: RadioTuner4MOST

Sink: Amplifier4MOST

- Listen to the radio on the MOST network.

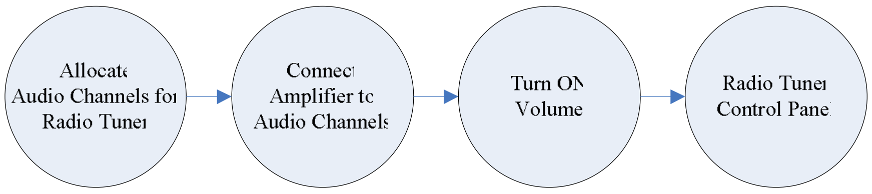

Allocate Audio Channels for Radio Tuner
Connect Amplifier to Audio Channels
Turn ON Volume
Radio Tuner Control Panel

Figure 4.42 Flow Diagram of Application 1

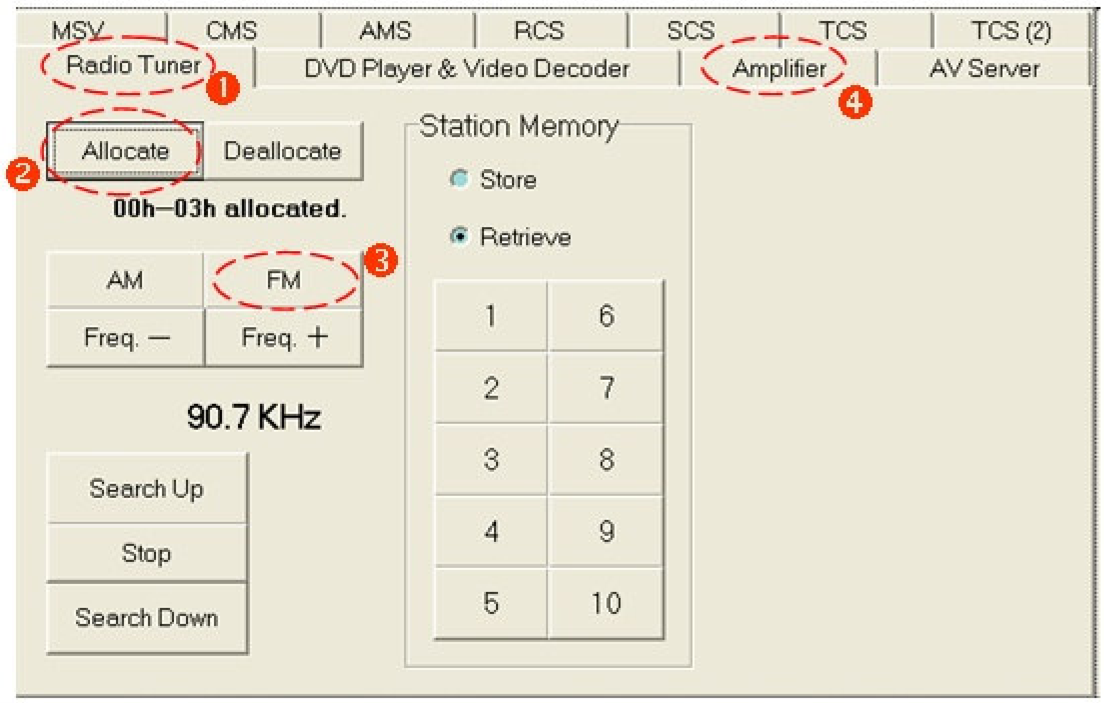
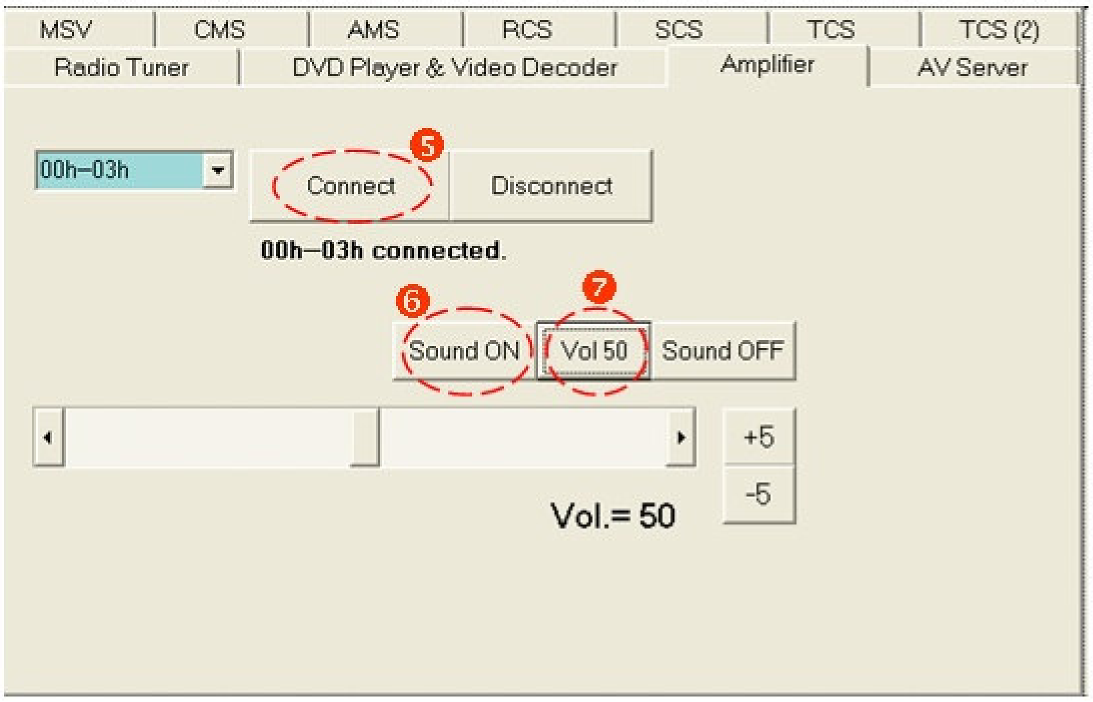


Figure 4.43 Screenshot of Application 1, 1 of 2

Table 4.8 Instructions for Application 1, 1 of 2

Step	Description
1	Come to “Radio Tuner” page.
2	Allocate synchronous channels for the audio stream.

3	Control the Radio Tuner via the control panel. (e.g. switch the waveband to FM)
4	Go to “Amplifier” page.



00h–03h connected.

Figure 4.44 Screenshot of Application 1, 2 of 2

Table 4.9 Instructions for Application 1, 2 of 2

Step	Description
5	Connect the Amplifier to the allocated channels.
6	Turn on the sound.
7	Set the volume to 50.

4.4.2 Application 2: Watch DVD movies

Source: DVDPlayer4MOST

Sink: Amplifier4MOST

- Listen to an audio CD on the MOST network.
- Watch VCD/DVD movies.

(Video is output from the DVD player while audio is output from the amplifier.)

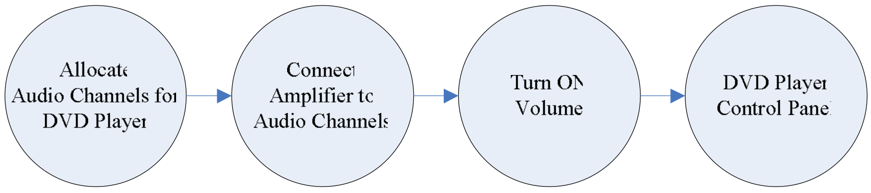

Allocate Audio Channels for DVD Player
Connect Amplifier to Audio Channels
Turn ON Volume
DVD Player Control Panel

Figure 4.45 Flow Diagram of Application 2

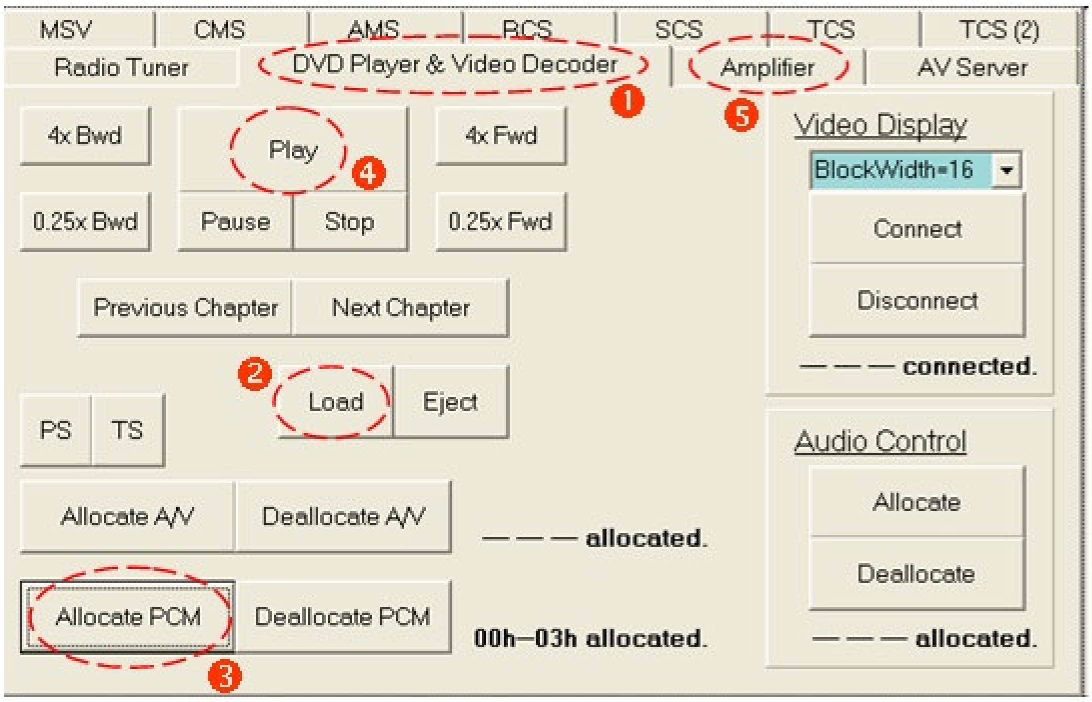
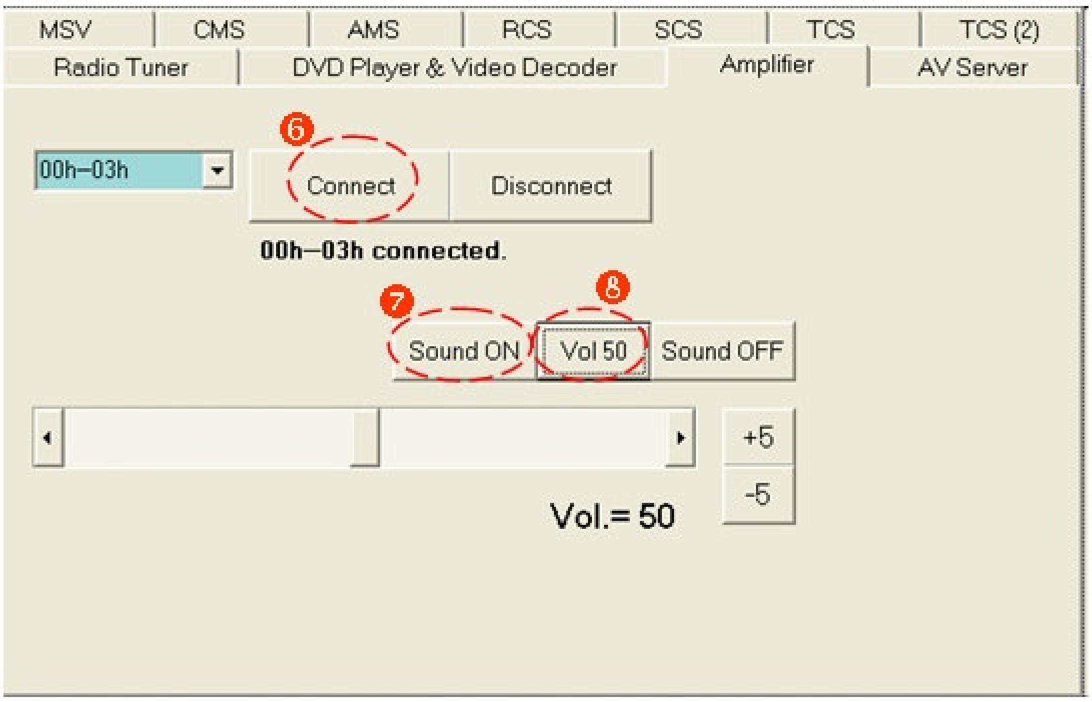


Figure 4.46 Screenshot of Application 2, 1 of 2

Table 4.10 Instructions for Application 2, 1 of 2

Step	Description
1	Come to “DVD Player & Video Decoder” page.
2	Load the disc if needed.
3	Allocate synchronous channels for the audio stream.
4	Control the DVD Player via the control panel. (e.g. start playing the disc)
5	Go to “Amplifier” page



00h–03h connected.

Vol.= 50

Figure 4.47 Screenshot of Application 2, 2 of 2

Table 4.11 Instructions for Application 2, 2 of 2

Step	Description
6	Connect the Amplifier to the allocated channels.
7	Turn on the sound.
8	Set the volume to 50.

4.4.3 Application 3: Watch DVD movies (with Video Decoder)

Source: DVDPlayer4MOST, VideoDecoder4MOST

Sink: VideoDecoder4MOST, Amplifier4MOST

- Watch VCD/DVD movies.

(A/V stream from the DVD player is decoded. Video is output from the decoder while audio is output from the amplifier.)

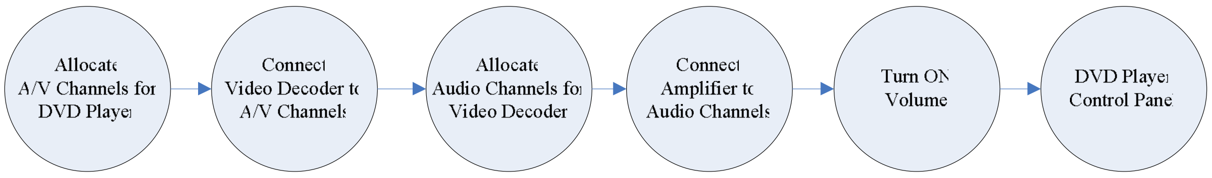
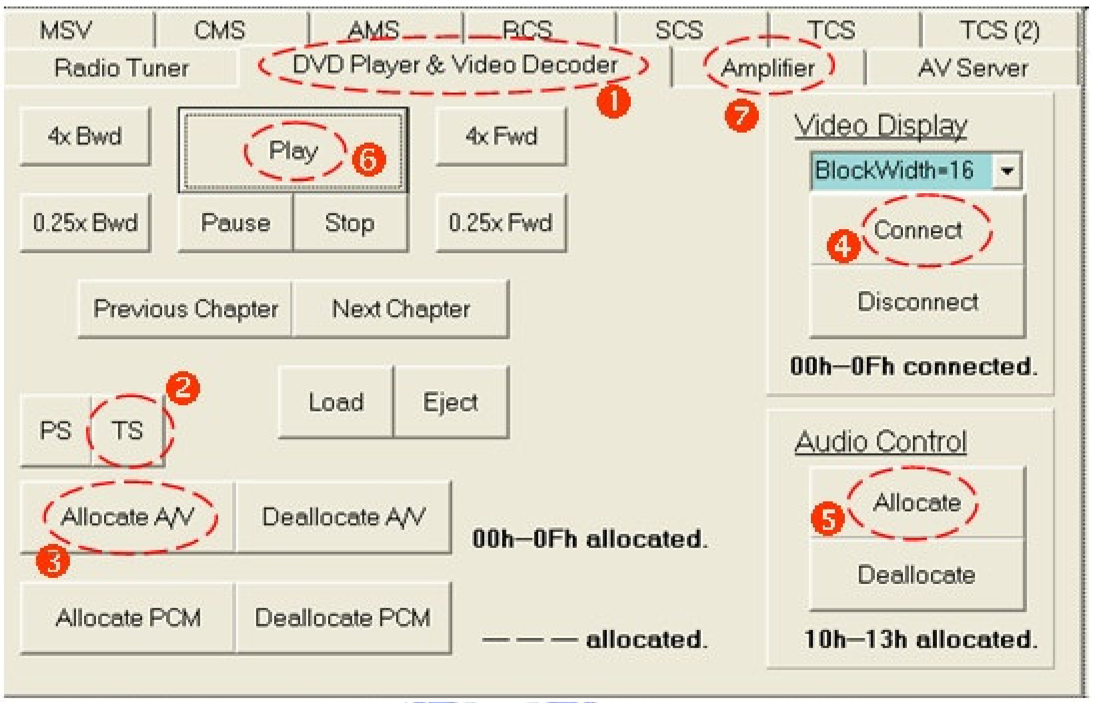

Allocate A/V Channels for DVD Player
Connect Video Decoder to A/V Channels
Allocate Audio Channels for Video Decoder
Connect Amplifier to Audio Channels
Turn ON Volume
DVD Player Control Panel

Figure 4.48 Flow Diagram of Application 3



00h–0Fh connected.

00h–0Fh allocated.

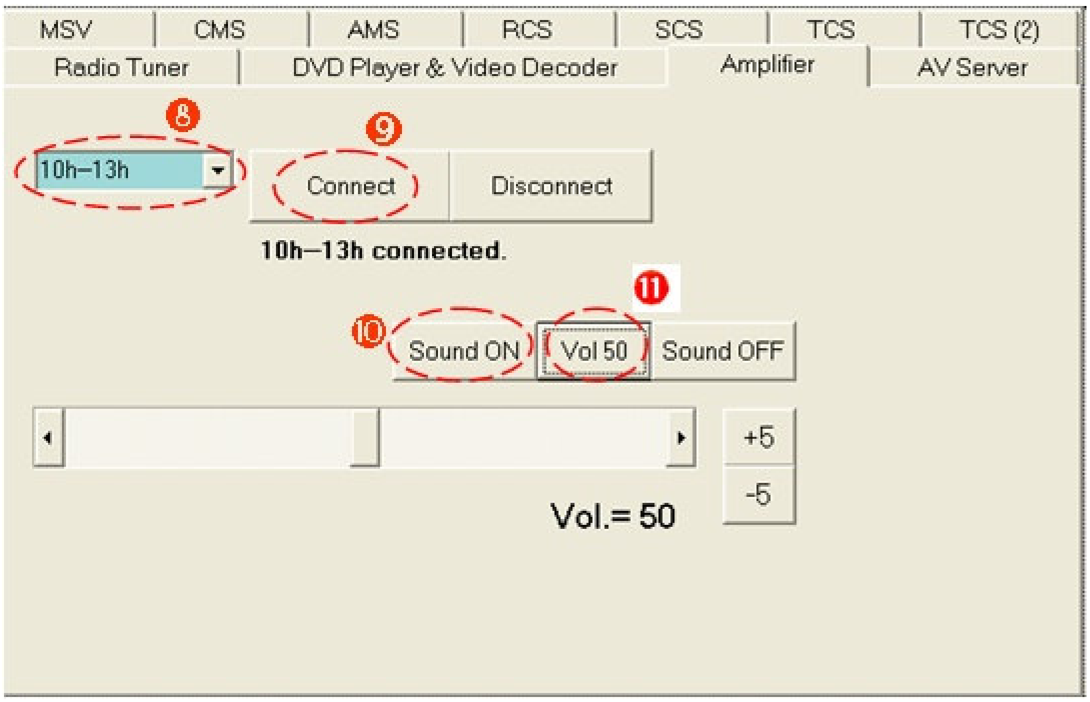
– – – – allocated.

10h–13h allocated.

Figure 4.49 Screenshot of Application 3, 1 of 2

Table 4.12 Instructions for Application 3, 1 of 2

Step	Description
1	Come to "DVD Player & Video Decoder" page.
2	Set the A/V stream source as MPEG-2 TS (Transport Stream).
3	Allocate synchronous channels for the A/V stream.
4	Connect the Video Decoder to the allocated channels.
5	Allocate synchronous channels for the audio stream decoded from the A/V stream.
6	Control the DVD Player via the control panel. (e.g. start playing the disc)
7	Go to "Amplifier" page.



10h–13h connected.

Vol.= 50

Figure 4.50 Screenshot of Application 3, 2 of 2

Table 4.13 Instructions for Application 3, 2 of 2

Step	**Description**
8	Choose the allocated channels for the audio stream. (10h--13h)
9	Connect the Amplifier to the allocated channels.
10	Turn on the sound.
11	Set the volume to 50.

4.4.4 Application 4: The MOST PCI Board as a Sound Card

Source: MOST PCI Board

Sink: Amplifier4MOST

- The MOST PCI Board as a sound card.

  (Sounds played from PC are routed to the amplifier for output)

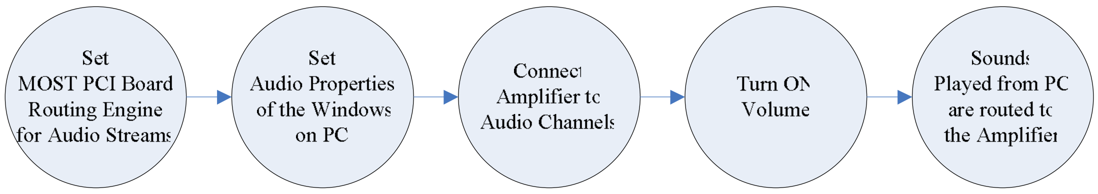

Set MOST PCI Board Routing Engine for Audio Streams
Set Audio Properties of the Windows on PC
Connect Amplifier to Audio Channels
Turn ON Volume
Sounds Played from PC are routed to the Amplifier

Figure 4.51 Flow Diagram of Application 4

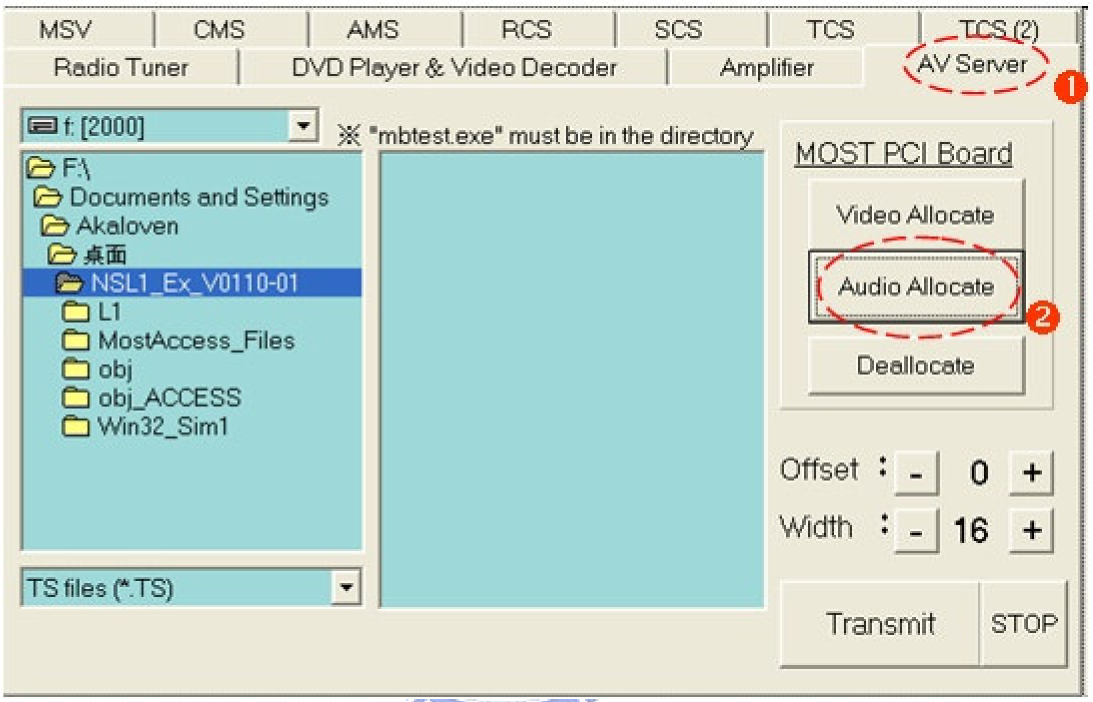
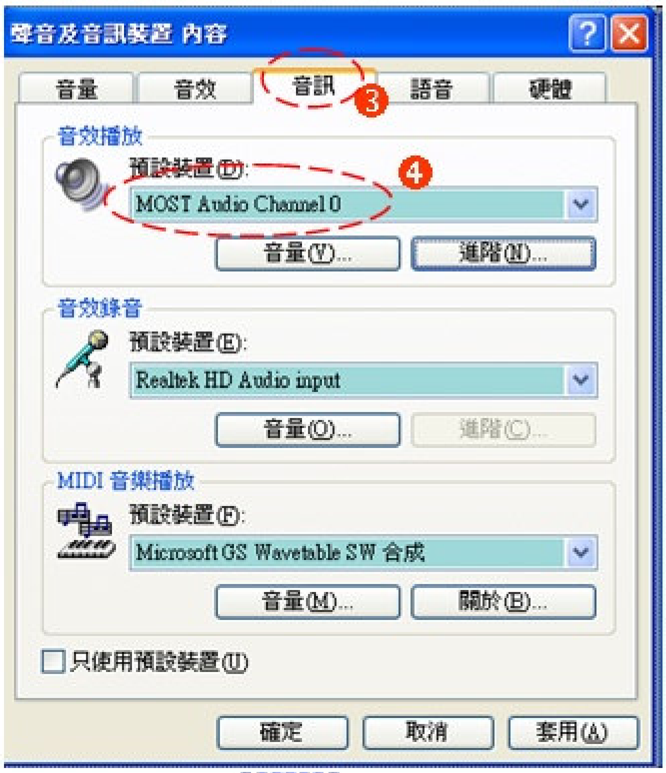


Figure 4.52 Screenshot of Application 4, 1 of 3

Table 4.14 Instructions for Application 4, 1 of 3

Step	Description
1	Come to "AV Server" page.
2	Set the MOST PCI Board routing engine for audio streams.



音效播放
預設裝置(D): MOST Audio Channel 0

音效錄音
預設裝置(E): Realtek HD Audio input

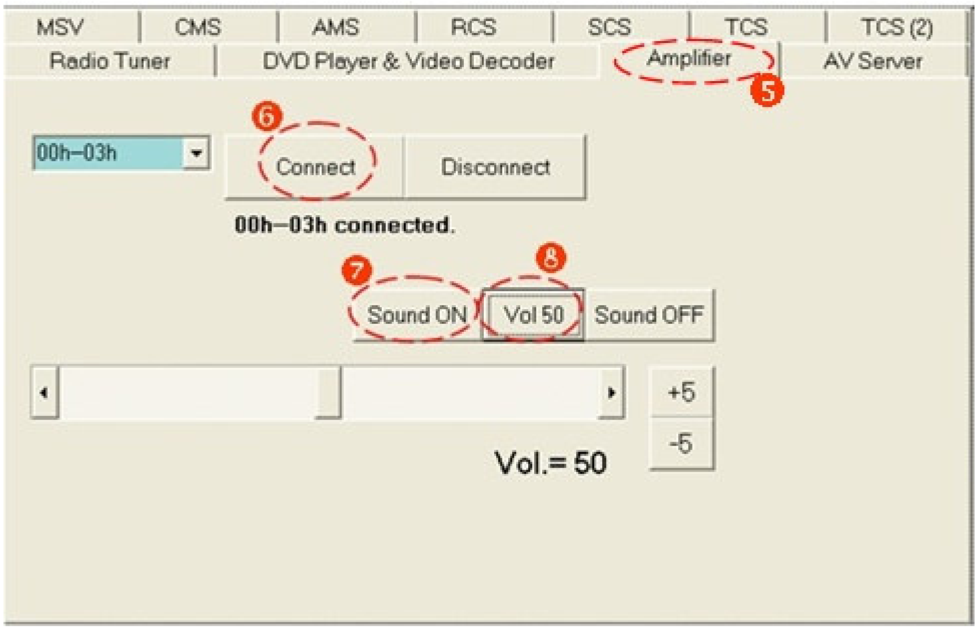
MIDI 音樂播放
預設裝置(F): Microsoft GS Wavetable SW 合成

☐ 只使用預設裝置(U)

Figure 4.53 Screenshot of Application 4, 2 of 3

Table 4.15 Instructions for Application 4, 2 of 3

Step	**Description**
3	Open the window of "Audio Properties" on PC. Switch to "Audio Devices" page.
4	Choose "MOST Audio Channel 0" as default device for sound playback.



00h–03h connected.

Vol.= 50

Figure 4.54 Screenshot of Application 4, 3 of 3

Table 4.16 Instructions for Application 4, 3 of 3

Step	**Description**
5	Come to "Amplifier" page for volume adjustment.
6	Connect the Amplifier to the allocated channels (which are connected by MOST Audio Channel 0).
7	Turn on the sound.
8	Set the volume to 50.

4.4.5 Application 5: Audio Streams be captured and played on PC

Source: RadioTuner4MOST or DVDPlayer4MOST

Sink: MOST PCI Board

- Listen to the radio on PC. (taken for example as follows)
- Listen to an audio CD on PC.

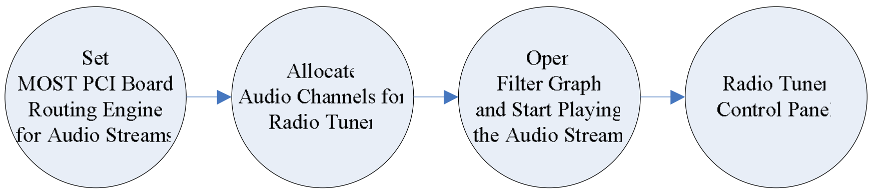

Set MOST PCI Board Routing Engine for Audio Streams
Allocate Audio Channels for Radio Tuner
Open Filter Graph and Start Playing the Audio Stream
Radio Tuner Control Panel

Figure 4.55 Flow Diagram of Application 5

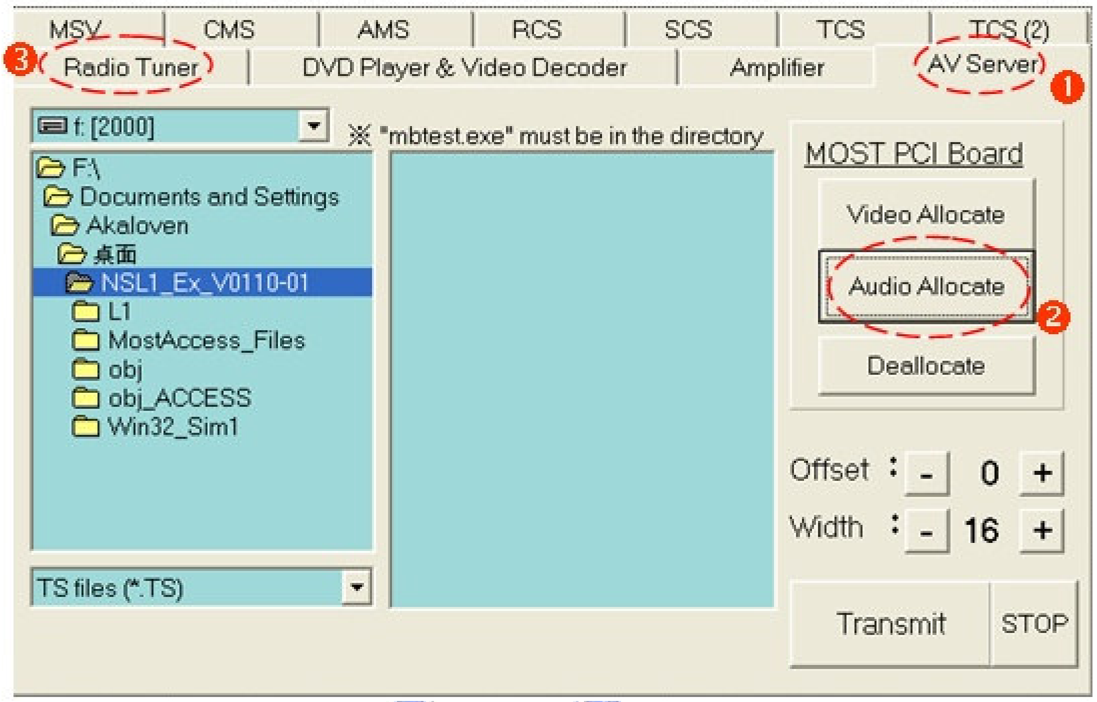


Figure 4.56 Screenshot of Application 5, 1 of 3

Table 4.17 Instructions for Application 5, 1 of 3

Step	Description
1	Come to "AV Server" page.
2	Set the MOST PCI Board routing engine for audio streams.
3	Go to "Radio Tuner" page.

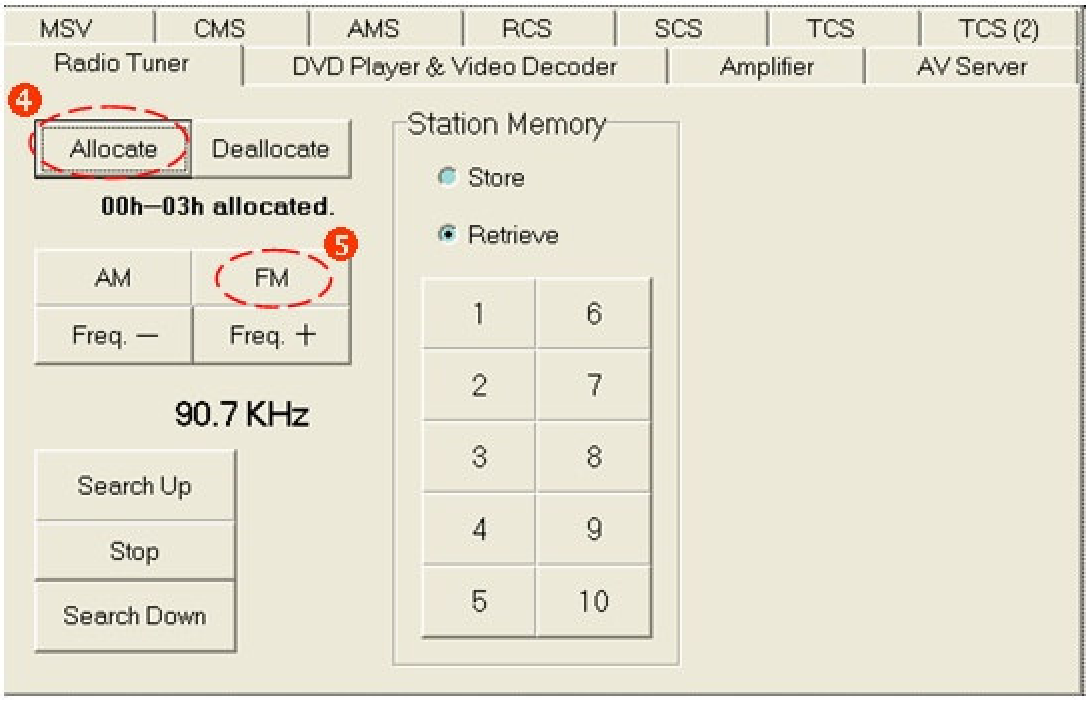


Figure 4.57 Screenshot of Application 5, 2 of 3

Table 4.18 Instructions for Application 5, 2 of 3

Step	Description
4	Allocate synchronous channels for the audio stream.
5	Control the Radio Tuner via the control panel. (e.g. switch the waveband to FM)

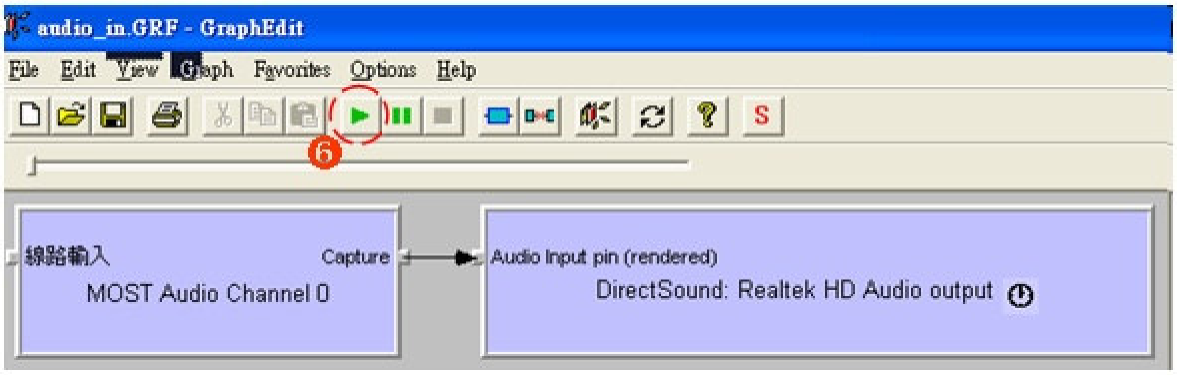


Figure 4.58 Screenshot of Application 5, 3 of 3

Table 4.19 Instructions for Application 5, 3 of 3

Step	Description
6	Open the pre-built filter graph file “audio_in.GRF” in GraphEdit, as shown in Figure

	4.58. Click the button to play the graph, and the audio stream from the MOST network will be rendered by the audio device on PC.

4.4.6 Application 6: MPEG-2 Streams be captured and played on PC

Source: DVDPlayer4MOST

Sink: MOST PCI Board

- Watch VCD\DVD movies on PC.

  (By using DirectShow filters, capture MPEG-2 Program Streams from the DVD player.)

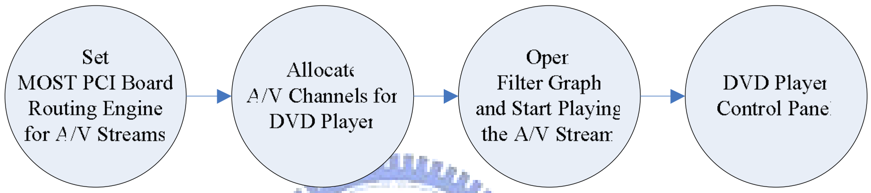

Set MOST PCI Board Routing Engine for A/V Streams
Allocate A/V Channels for DVD Player
Open Filter Graph and Start Playing the A/V Stream
DVD Player Control Panel

Figure 4.59 Flow Diagram of Application 6

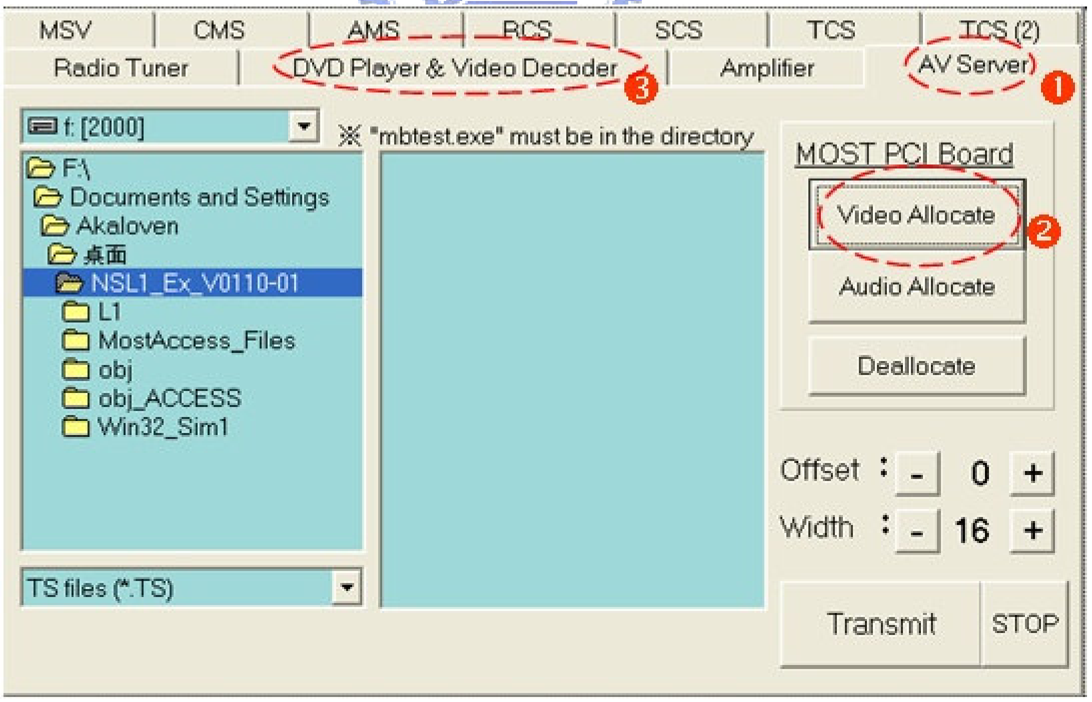
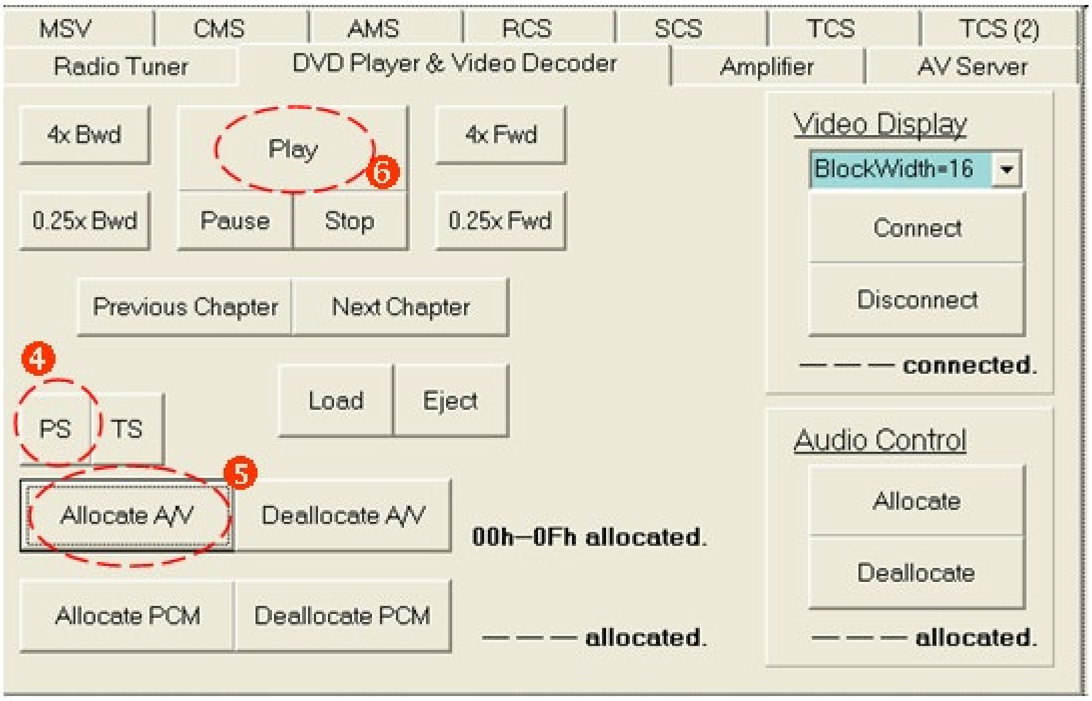


Figure 4.60 Screenshot of Application 6, 1 of 3

Table 4.20 Instructions for Application 6, 1 of 3

Step	Description
1	Come to "AV Server" page.
2	Set the MOST PCI Board routing engine for A/V streams.
3	Go to "DVD Player & Video Decoder" page.



00h–0Fh allocated.

——— allocated.

——— connected.

——— allocated.

Figure 4.61 Screenshot of Application 6, 2 of 3

Table 4.21 Instructions for Application 6, 2 of 3

Step	Description
4	Set the A/V stream source as MPEG-2 PS (Program Stream).
5	Allocate synchronous channels for the A/V stream.
6	Control the DVD Player via the control panel. (e.g. start playing the disc)

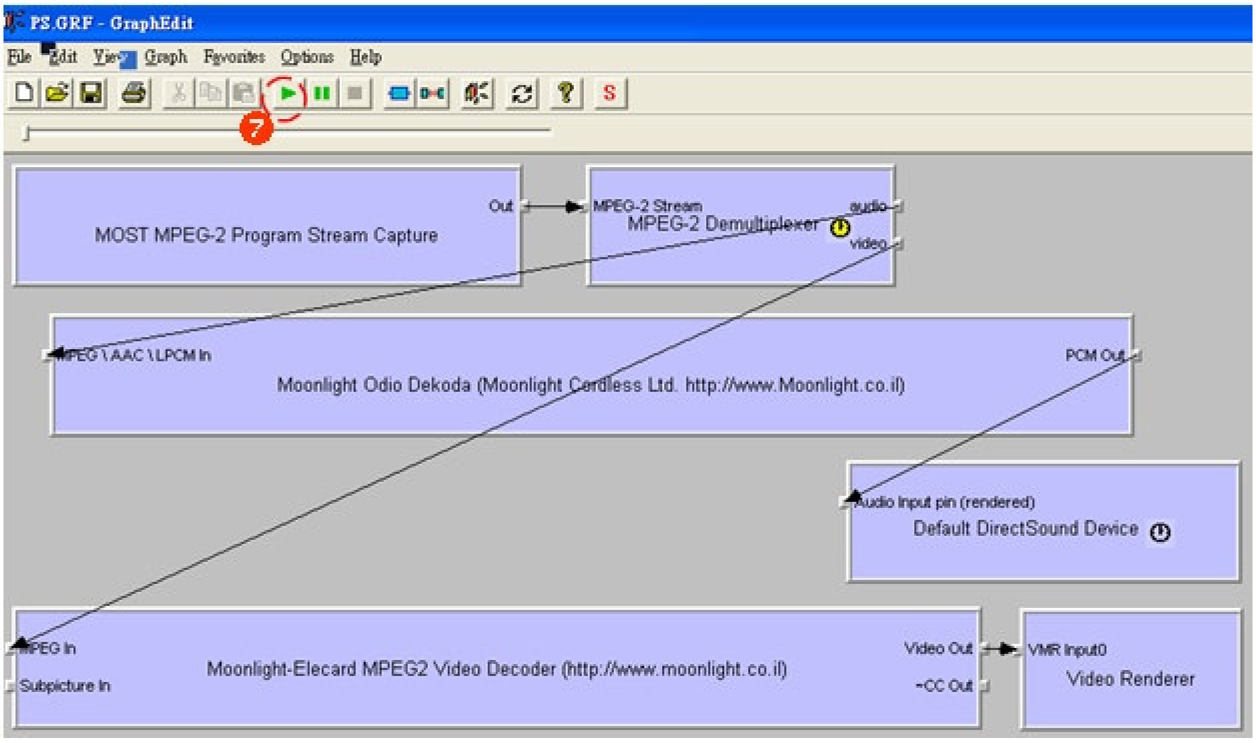
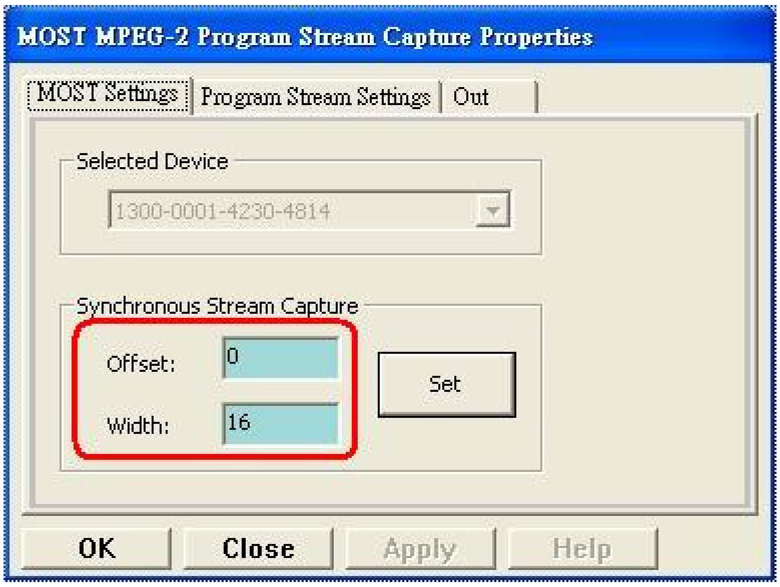

PS.GRF - GraphEdit
File Edit View Graph Favorites Options Help
7
MOST MPEG-2 Program Stream Capture
Out
MPEG-2 Stream
MPEG-2 Demultiplexer
audio
video
MPEG \ AAC \ LPCM In
Moonlight Odio Dekoda (Moonlight Cordless Ltd. http://www.Moonlight.co.il)
PCM Out
Audio Input pin (rendered)
Default DirectSound Device
MPEG In
Subpicture In
Moonlight-Elecard MPEG2 Video Decoder (http://www.moonlight.co.il)
Video Out
~CC Out
VMR Input0
Video Renderer

Figure 4.62 Screenshot of Application 6, 3 of 3

Table 4.22 Instructions for Application 6, 3 of 3

Step	Description
7	Open the pre-built filter graph file "PS.GRF" in GraphEdit, as shown in Figure 4.62. Click the button to play the graph. The A/V stream from the DVD Player on the MOST network will be demuxed into audio and video streams which are transformed, and finally rendered by the audio device and video renderer on PC, respectively.

Please note: When constructing the filter graph used in this application, properties of certain filters need to be set, as shown in the following figures on the next page. For "MOST MPEG-2 Program Stream Capture" properties, we should make sure that "Offset" and "Width" settings for the synchronous stream fit our case. For "MPEG-2 Demultiplexer" properties, the output pins, video and audio, need to be added manually; moreover, "stream_id" and "Content" of the output pins should follow MPEG-2 standard.



MOST MPEG-2 Program Stream Capture Properties

MOST Settings | Program Stream Settings | Out

Selected Device: 1300-0001-4230-4814

Synchronous Stream Capture

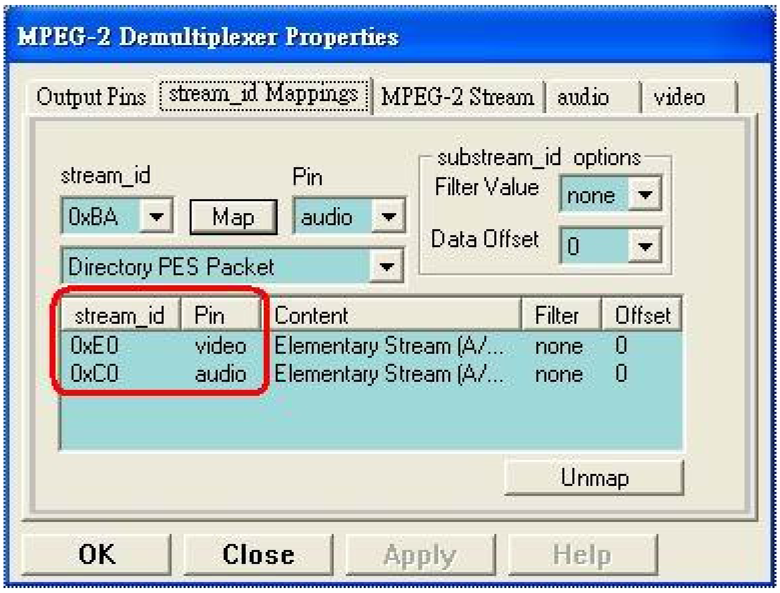
Offset: 0

Width: 16

Set

OK | Close | Apply | Help

Figure 4.63 MOST MPEG-2 PS Capture Property Setting



MPEG-2 Demultiplexer Properties

Output Pins | stream_id Mappings | MPEG-2 Stream | audio | video

stream_id: 0xBA   Map   Pin: audio

Directory PES Packet

substream_id options — Filter Value: none, Data Offset: 0

stream_id	Pin	Content	Filter	Offset
0xE0	video	Elementary Stream (A/...	none	0
0xC0	audio	Elementary Stream (A/...	none	0

Unmap

OK | Close | Apply | Help

Figure 4.64 MPEG-2 Demultiplexer Property Setting

4.4.7 Application 7: A/V Server under MOST Network

A/V Server under MOST network is called “Application 7” in this thesis and is introduced separately in Chapter 5. Please go on to the next chapter.

CHAPTER 5

Audio/Video Server under MOST Network

5.1 The Idea and Design

As introduced in Chapter 4, we have carried out several multimedia applications under MOST network, with the utilities from SMSC. Transmission of audio streams between the MOST network and a PC is fulfilled, as well as A/V streams on the MOST network can be captured to a PC and played; however, we'd like to stream the video files which are commonly played on a PC, to the MOST network to be displayed. In other words, it is the idea of an A/V server under MOST network. With the VideoDecoder4MOST and the MOST PCI Board we have, it is possible to realize this idea.

5.1.1 VideoDecoder4MOST

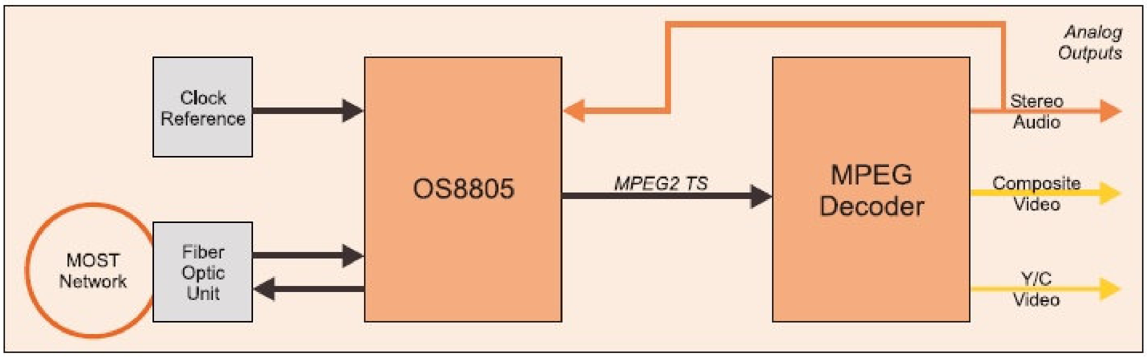


Figure 5.1 VideoDecoder4MOST Block Diagram [25]

With the block diagram shown in Figure 5.1, we'll illustrate how VideoDecoder4MOST works. MPEG-2 Transport Stream data packets are transmitted over the MOST network via synchronous channels as continuously streamed data bytes. The OS8805 receives the broadcasted audio/video content and rebuilds the MPEG2 packets from the synchronous data stream. These packets are passed to the MPEG Decoder for further processing. As described above, if we can transfer the video file into MPEG-2 TS format and stream the data onto the MOST network, the A/V stream will be received by the VideoDecoder4MOST and then decoded for analog out.

5.1.2 MOST PCI Board

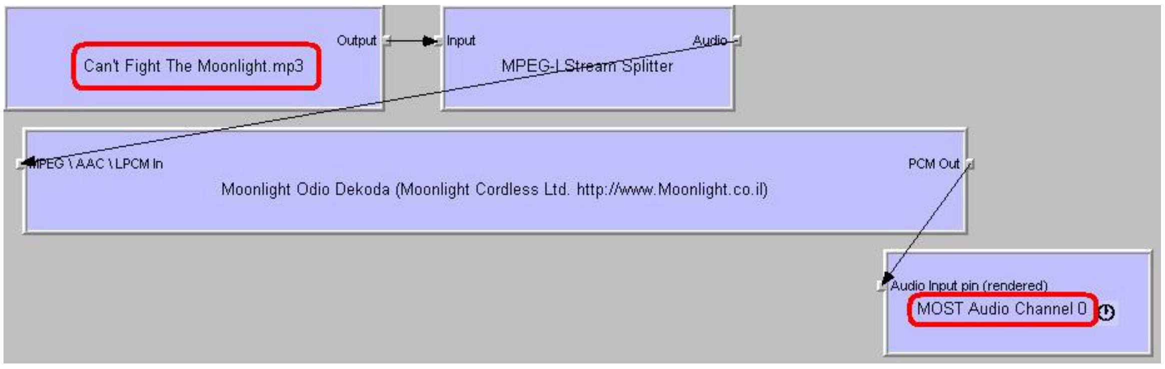

Can't Fight The Moonlight.mp3
Output
Input
Audio
MPEG-I Stream Splitter
MPEG \ AAC \ LPCM In
Moonlight Odio Dekoda (Moonlight Cordless Ltd. http://www.Moonlight.co.il)
PCM Out
Audio Input pin (rendered)
MOST Audio Channel 0

Figure 5.2 Rendering an Audio File by MOST Audio Channel 0

Through the MOST PCI Board, as shown by the application in Section 4.4.4, real-time audio data can be transferred onto the MOST network. Disregarding the audio device settings of the Windows operating system, we can also send audio streams to "MOST Audio Channel 0" in GraphEdit, as shown in Figure 5.2. After allocating the synchronous channels and connecting the Amplifier4MOST as those in Section 4.4.4, the sound can be heard from the Amplifier4MOST.

By the means of GraphEdit and the respective DirectShow drivers for MOST, it is easy to send streaming data to the MOST network; but unfortunately a DirectShow filter for video rendering via the MOST PCI Board is not available from SMSC, since they did not develop the drivers for that use. However, combined with the MOST50 drivers v1.4.4 for MOST PC Interfaces, there is a small tool application that might solve our problem in another way, introduced as follows:

• ***mbtest.exe*** **[26]**

"***mbtest***" is a small DOS mode based reference application designated to test the transmission over the MOST network. With the command "**synchtx**", it sends a continuous synchronous data stream to the MOST network, and with "**synchrx**", it receives a continuous synchronous data stream from the MOST network. Channel offset and width are specified as parameters of the command. Explanation of its usage with the commands, arguments and options will be shown by executing "*mbtest -h*".

5.1.3 Applicable File Types

The VideoDecoder4MOST supports MPEG-2 Transport Stream (TS) only; hence, before streaming video files on the hard disk to the MOST network, we convert the files into MPEG-2 TS format (the file extension is ".TS") in advance. The conversion can be done by video convert software that supports MPEG-2 TS encoding with certain detailed settings. In general, the stream must be synchronized to the MOST clock frequency before transmitted over MOST. For isochronous streams, e.g. MPEG-1 System Stream, MPEG-2 Program Stream, and MPEG-2 Transport Stream, before the transmission of a Systems Layer multiplex, the stream must be synchronized to a constant MOST bit rate by using a MPEG compliant mechanism (e.g. by applying stuffing bytes). Please refer to [27] for detailed information about MOST-A/V streams; however, by experimenting with different MPEG-2 encoder settings and making comparison with the A/V streams generated by the DVDPlayer4MOST (captured via "*mbtest.exe*"), we conclude the following properties of applicable TS files for our application:

- The video must be PAL (with a Frame rate of 25 fps) to be decoded correctly.
- For A/V streams with BlockWidth = 16 (mostly applied), the video must have a Constant Bit Rate (CBR) of 5178 Kbps. According to the MPEG-2 TS streams generated by the

DVDPlayer4MOST, the video bit rates for other possible BlockWidths are: 3800 Kbps for BlockWidth = 12, 2422 Kbps for BlockWidth = 8, and 1044 Kbps for BlockWidth = 4.

- The audio must be MP2 (MPEG-1 Audio Layer 2), while AC3 or other formats can't be decoded by the MPEG decoder.
- Default bit rate of the audio is 192 Kbps, which may cause a slight lag when the audio is output from the Amplifier4MOST; therefore, we set it to 224 Kbps which seems to make the audio synchronized better for our case.

Table 5.1 Information of a converted TS file

General	Video #1E1	Audio #1E2
Format : MPEG-2 Transport	Codec : MPEG-2 Video	Codec : MPEG-1 Audio layer 2
Format/Family : MPEG-2	Codec profile : Main@Main	PlayTime : 1mn 36s
File size : 64.7 MiB	Codec settings/Matrix : Standard	Bit rate : 224 Kbps
PlayTime : 1mn 36s	PlayTime : 1mn 36s	Bit rate mode : CBR
Bit rate : 5643 Kbps	Bit rate : 5178 Kbps	Channel(s) : 2 channels
	Bit rate mode : CBR	Sampling rate : 48 KHz
	Width : 720 pixels	Resolution : 16 bits
	Height : 576 pixels	
	Aspect ratio : 4/3	
	Frame rate : 25.000 fps	
	Standard : PAL	
	Chroma : 4:2:0	
	Interlacement : Top Field First	
	Bits/(Pixel*Frame) : 0.499	

Table 5.1 lists the information of a TS file converted for our needs. The information is gathered and displayed by the "KMPlayer", a multimedia player on the Windows platform.

5.2 Implementation

5.2.1 The User Interface

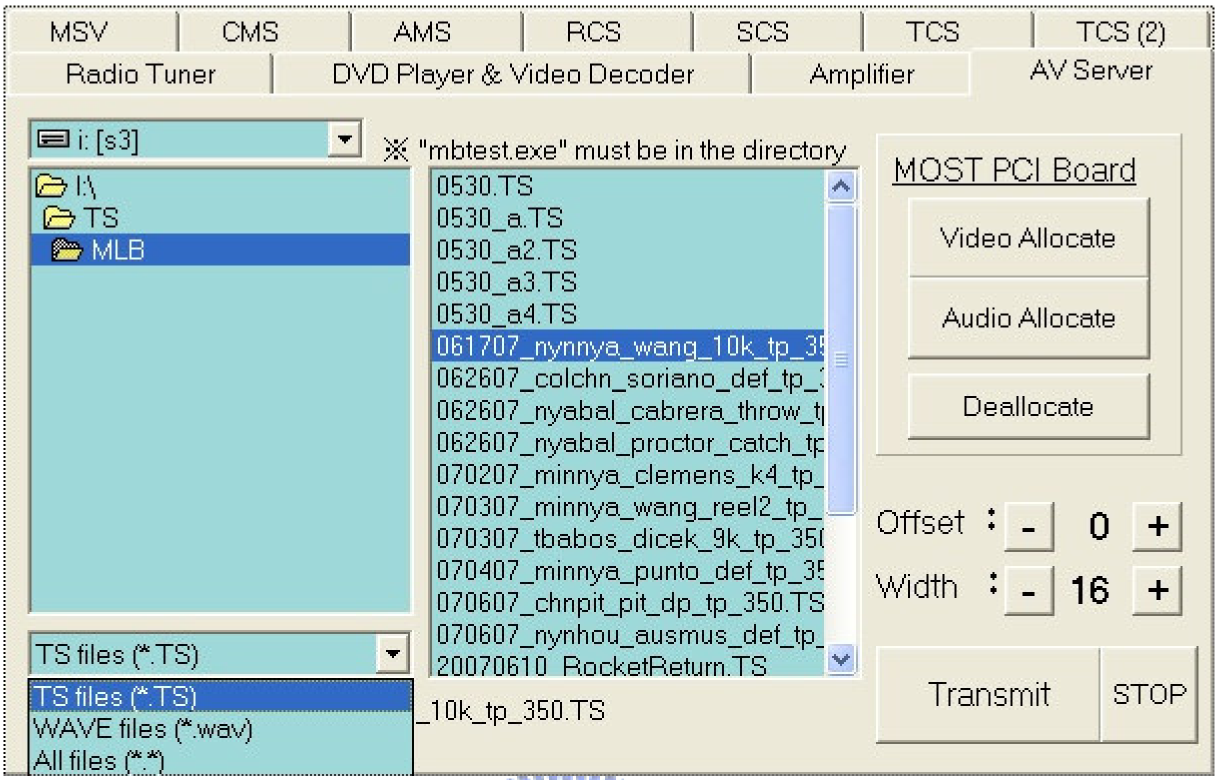

MSV
CMS
AMS
RCS
SCS
TCS
TCS (2)
Radio Tuner
DVD Player & Video Decoder
Amplifier
AV Server
i: [s3]
I:\
TS
MLB
※ "mbtest.exe" must be in the directory
0530.TS
0530_a.TS
0530_a2.TS
0530_a3.TS
0530_a4.TS
061707_nynnya_wang_10k_tp_35
062607_colchn_soriano_def_tp_
062607_nyabal_cabrera_throw_t
062607_nyabal_proctor_catch_tp
070207_minnya_clemens_k4_tp_
070307_minnya_wang_reel2_tp_
070307_tbabos_dicek_9k_tp_350
070407_minnya_punto_def_tp_35
070607_chnpit_pit_dp_tp_350.TS
070607_nynhou_ausmus_def_tp_
20070610_RocketReturn.TS
TS files (*.TS)
TS files (*.TS)
WAVE files (*.wav)
All files (*.*)
_10k_tp_350.TS
MOST PCI Board
Video Allocate
Audio Allocate
Deallocate
Offset : - 0 +
Width : - 16 +
Transmit
STOP

Figure 5.3 A/V Server User Interface

Figure 5.3 is a screenshot of the "AV Server" page in our control program, NSL1_EX. The control program for MOST PCI Board is included, on the top right of this page. On the left side is a file browser which consists of a DriveComboBox, a DirectoryListBox, a FileListBox, and a FilterComboBox. In the file browser, we can select a file to transmit. Below the file browser is a label representing the current selected file.

Please note: "*mbtest.exe*" must be in the directory, together with the file we want to stream, or it can not be executed when we click the "Transmit" button.

On the bottom right side, we can adjust the Offset and BlockWidth parameters for the A/V stream. The button "Transmit" is used for starting the transmission while "STOP" is used for terminating the transmitting process.

"*mbtest.exe*" plays an important role in implementing this control program. With the button "Transmit" clicked, the function *ShellExecute* is called to execute an external application, "*cmd.exe*" (the command line interpreter). Then we set a *mbtest* command as the argument of "*cmd.exe*", and the complete command for streaming the file to the MOST network is:

mbtest synchtx <channel offset> <channel width> -r <file name>

Here <file name> is given by the label representing the current selected file, with full path. As for the "STOP" button, it sends a message to close an "opened handle" which represents the "*cmd.exe*" window when it pops up; as a result, the transmission will be terminated.

5.2.2 Step-by-Step Instructions

Source: MOST PCI Board, VideoDecoder4MOST

Sink: VideoDecoder4MOST, Amplifier4MOST

- Application 7: A/V Server under MOST network

  (The selected MPEG-2 TS file is streamed onto the MOST network via the MOST PCI Board, and the stream is decoded by the Video Decoder. Video is output from the decoder while audio is output from the amplifier.)

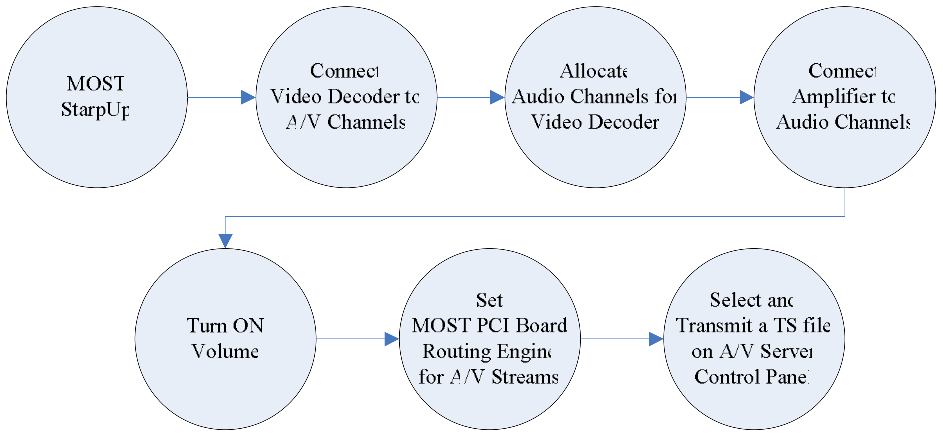

MOST StarpUp
Connect Video Decoder to A/V Channels
Allocate Audio Channels for Video Decoder
Connect Amplifier to Audio Channels
Turn ON Volume
Set MOST PCI Board Routing Engine for A/V Streams
Select and Transmit a TS file on A/V Server Control Panel

Figure 5.4 Flow Diagram of Application 7

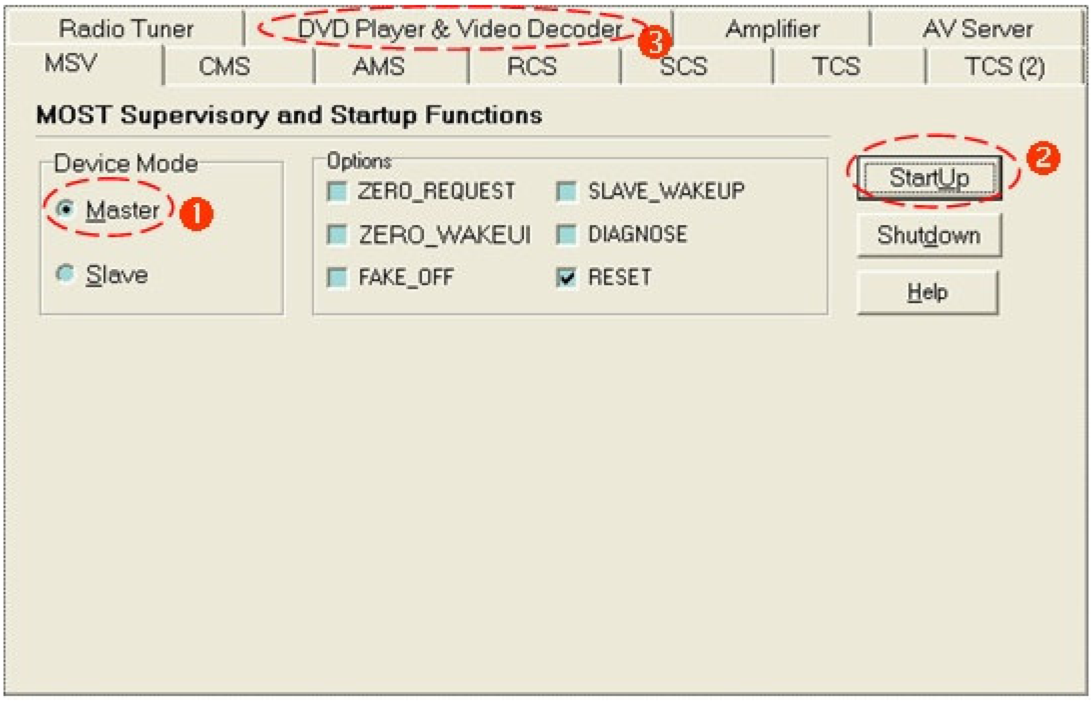


Figure 5.5 Screenshot of Application 7, 1 of 5

Table 5.2 Instructions for Application 7, 1 of 5

Step	Description
0	Start MICtrl (connecting to the MOST PCI Board) and NSL1_EX (connecting to the Optolyzer Box on COM 1).
1	Set the Optolyzer Box as Timing Master.
2	Start up the MOST network.
3	Go to "DVD Player & Video Decoder" page.

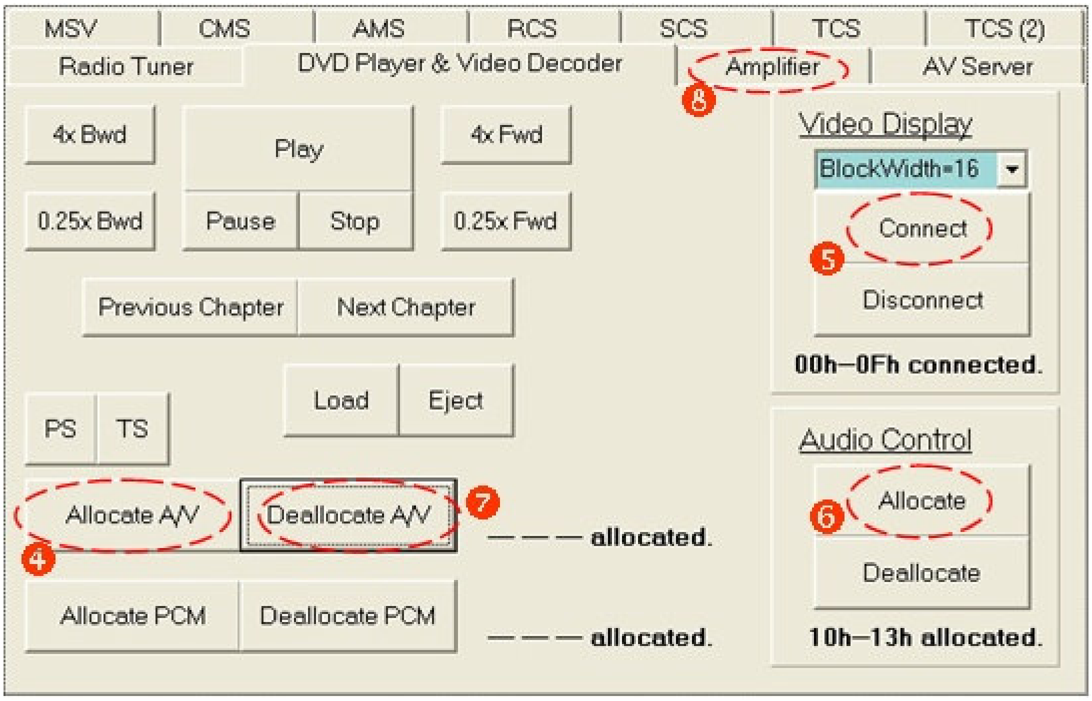
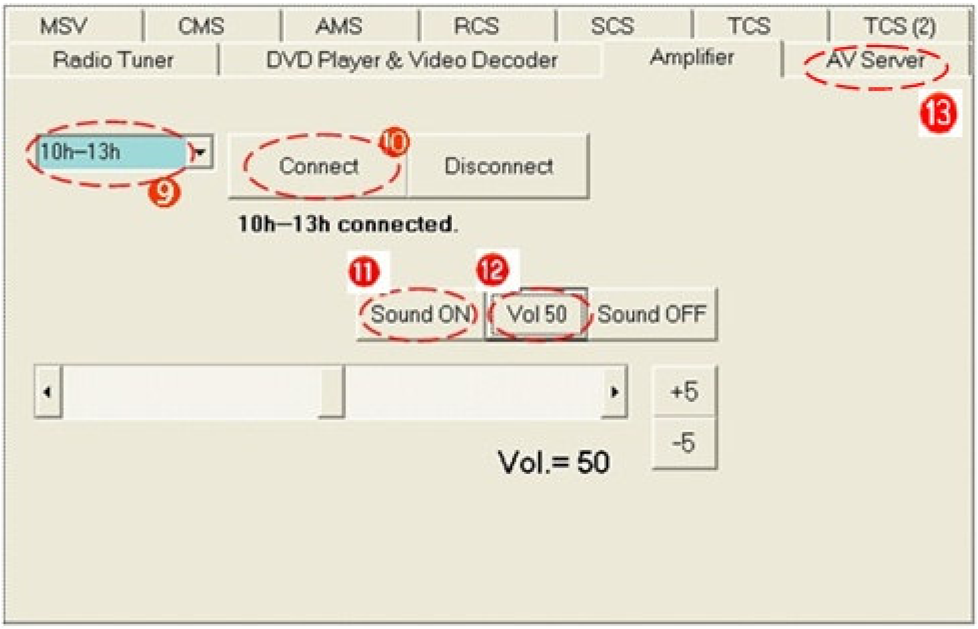


Figure 5.6 Screenshot of Application 7, 2 of 5

Table 5.3 Instructions for Application 7, 2 of 5

Step	Description
4	Allocate synchronous channels for the A/V stream.
5	Connect the Video Decoder to the allocated channels. **Please note:** Without Step 4, 00h--03h will be allocated, which will result in a conflict with the MOST PCI Board routing (00h--0Fh).
6	Allocate synchronous channels for the audio stream decoded from the A/V stream.
7	De-allocate the synchronous channels allocated by the DVD Player
8	Go to “Amplifier” page.



10h--13h connected.

Vol.= 50

Figure 5.7 Screenshot of Application 7, 3 of 5

Table 5.4 Instructions for Application 7, 3 of 5

Step	**Description**
9	Choose the allocated channels for the audio stream. (10h--13h)
10	Connect the Amplifier to the allocated channels.
11	Turn on the sound.
12	Set the volume to 50.
13	Go to "AV Server" page.

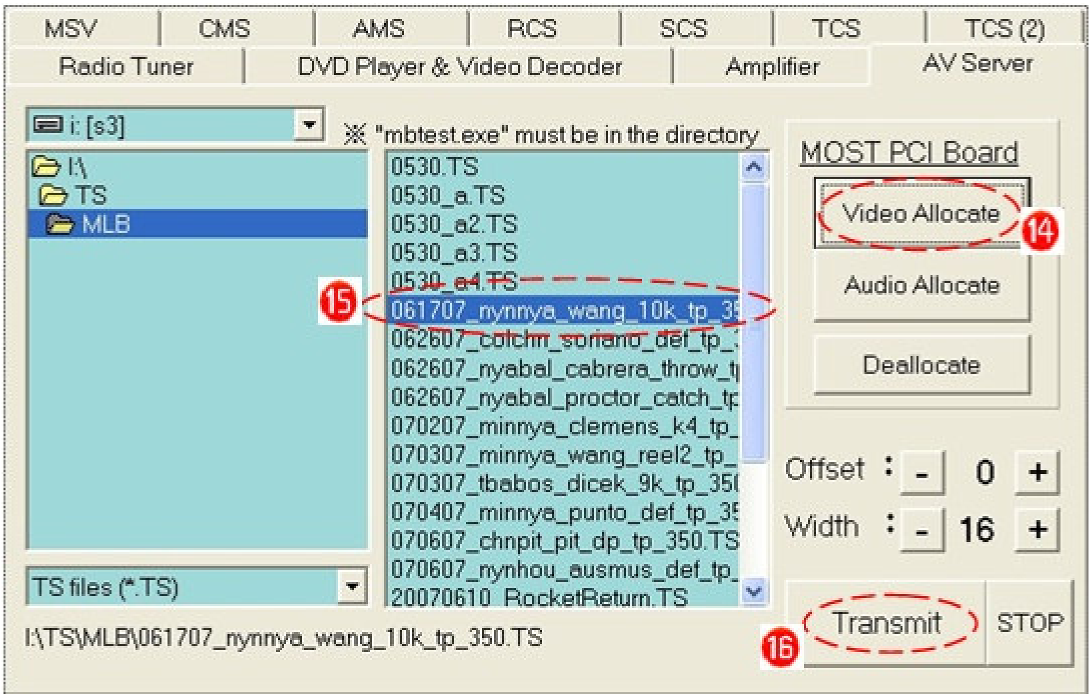
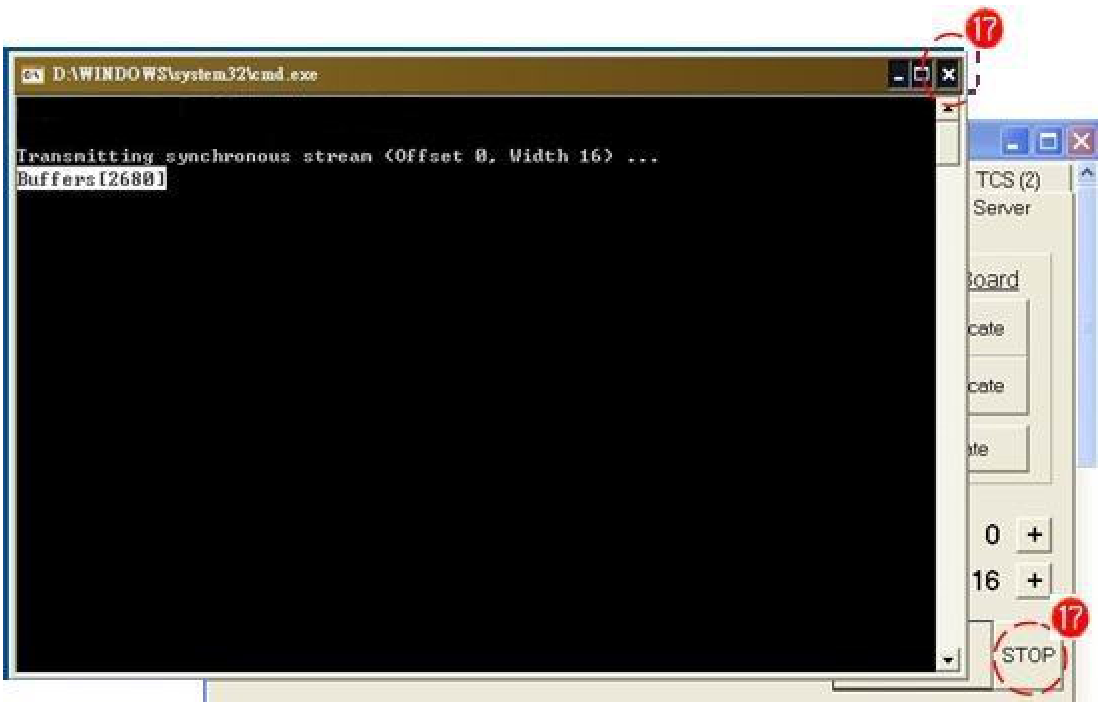

MSV
CMS
AMS
RCS
SCS
TCS
TCS (2)
Radio Tuner
DVD Player & Video Decoder
Amplifier
AV Server
i: [s3]
※ "mbtest.exe" must be in the directory
I:\
TS
MLB
0530.TS
0530_a.TS
0530_a2.TS
0530_a3.TS
0530_a4.TS
061707_nynnya_wang_10k_tp_35
062607_colchn_soriano_def_tp_
062607_nyabal_cabrera_throw_t
062607_nyabal_proctor_catch_tp
070207_minnya_clemens_k4_tp_
070307_minnya_wang_reel2_tp_
070307_tbabos_dicek_9k_tp_35
070407_minnya_punto_def_tp_35
070607_chnpit_pit_dp_tp_350.TS
070607_nynhou_ausmus_def_tp_
20070610_RocketReturn.TS
TS files (*.TS)
I:\TS\MLB\061707_nynnya_wang_10k_tp_350.TS
MOST PCI Board
Video Allocate
Audio Allocate
Deallocate
Offset : - 0 +
Width : - 16 +
Transmit
STOP
14
15
16

Figure 5.8 Screenshot of Application 7, 4 of 5

Table 5.5 Instructions for Application 7, 4 of 5

Step	Description
14	Set the MOST PCI Board routing engine for A/V streams.
15	Select a TS file to be sent.
16	Start transmitting the selected file to the MOST network as continuous streamed data bytes.



```
Transmitting synchronous stream (Offset 0, Width 16) ...
Buffers[2680]
```

Figure 5.9 Screenshot of Application 7, 5 of 5

Table 5.6 Instructions for Application 7, 5 of 5

Step	**Description**
17	The transmission can be terminated by closing the command window directly or clicking on the "STOP" button.

Please note: This control program can be used to stream audio files as well. The routing and connection should be set up as that of Application 4. However, in our test, only the audio files consisting of raw data, such as WAV encoded in PCM (Pulse-Code Modulation), are successfully processed by the Amplifier4MOST to output the sound. Therefore, for this case, the "wav" extension is added in the FilterComboBox of the file browser.

5.3 Result and Discussion

- **Result**

It is tough to completely meet the requirements of the VideoDecoder4MOST. The result is just acceptable, only few jitters or mosaic frames may occur during the decoding process. The bit rate must be exactly identical, and the file must stream in real-time with the actual bit rate. If this is not the case, a buffer over or under run will happen, and then the decoder must be resynchronized which leads to short stops or interrupts in displaying.

- **Advantages**

An A/V server under MOST network provides convenience for the user to watch the movies or listen to the music without a DVD or a CD disc. Media files are converted into unified formats and are stored on the hard disks, allowing easier management; hence the methods for transferring files to the decoder are simplified. By means of the MOST network, A/V streams are easily broadcasted to every node in the ring, that multiple sink nodes connecting to one source is also possible.

- **Disadvantages**

This application of an A/V server design is kind of defective, since originally it is regarded as a trial to implement the transmission of A/V streams from a PC to the MOST network. At first, the converted MPEG-2 TS files are not always perfectly decoded by the Video Decoder yet. Second, much disk space is taken due to the high bit rate which must be constant. Third, it wastes time converting the video files (up to 20% to 80% of the file's playtime is taken, depending on the output quality and the performance of the PC). Fourth, the movie of the A/V stream is out of control (e.g. to pause or to search forward) during the transmission.

- **Possible Improvements**

In our original design, a DirectShow rendering filter for the MOST PC Interface is essential to transmit the required A/V stream to the MOST network, as shown in Figure 5.10.

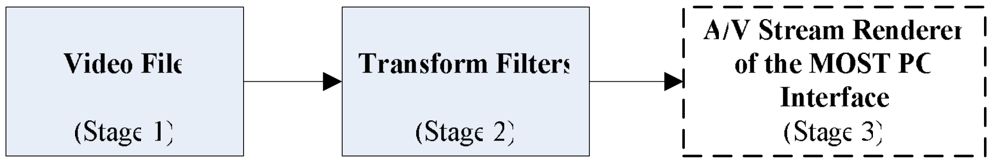


Figure 5.10 Imaginary Filter Graph of A/V Stream Rendering

Figure 5.10 is an imaginary filter graph just like a "Video version" of Figure 5.2. Ideally, the source file in Stage 1 can be of any video format since there are always transform filters in Stage 2 suitable for it. Furthermore, a transform filter with the required output format (e.g. MPEG-2 TS), which connects Stage 2 and Stage 3 is usually available. In fact, every component in Figure 5.9 is attainable nowadays except for the renderer in Stage 3. However, since we can use "*mbtest.exe*" instead, it is possible to create the A/V stream rendering filter with the functions (API) used in "*mbtest.exe*". Similar application using "*mbtest.exe*" to stream a media file to MOST can be found in the Application Note for "MOST NetServices V2 Basic Example " [28], but the same application is not seen in the former examples (built for NetServices V1 which is used in our program). For this reason, I suppose MOST NetServices API V2 is necessary to implement this streaming function. With a multimedia player implementing the filter graph in Figure 5.9, there will be many advantages including the follows:

1. The video is transferred and streamed instantaneously when being played that control of the video is made possible by the media player program (e.g. to pause or to jump to any location of the video).
2. Most video formats will be supported due to the variety codecs installed in the operating system. This saves time because no prior conversion is needed.
3. Similar to the above reason, if we want to change the channel BlockWidth used in the

MOST network, properties (such as data rate) of the A/V stream could be adjusted to fit the Video Decoder, only by some settings of the transform filter (the respective properties may be built in the player for conveniences), instead of re-converting the file with the required properties.

4. Files in compressed formats save more disk space.

CHAPTER 6
Conclusion

The contents of Chapter 4 and Chapter 5, especially the multimedia applications and the ways to implement their functions, are the cores of this thesis. Examples for using AMS, CMS and RCS of the MOST NetServices Layer 1 API are explained within the control program “NSL1_EX”. We also introduced the methods of controlling multimedia MOST devices such as RadioTuner4MOST, Amplifier4MOST, DVDPlayer4MOST and VideoDecoder4MOST. MOST protocols. We not only implemented the ordinary applications but also proposed and tested an unusual way to fulfill an A/V server under MOST network in Chapter 5. As shown in this thesis, with the aids of MOST NetServices API and the prototyping devices, programmers can construct their programs more efficiently to perform versatile applications under the MOST network.

- **Future works**

First, as mentioned at the end of Chapter 5, we may extend the functions for the application with NetServices V2; however, an upgrade of the PC Interface to MOST50 ePHY for performing full features of NetServices V2 [28]. Secondly, with the Asynchronous Data Transmission Services (ADS) which are not concerned in our research yet, the MOST PCI Board acts similar to an Ethernet card, as described in [26]. This feature may be useful in Internet browsing or making communications with a PDA. Thirdly, by adding any applicable MOST devices into the ring, the ability of MOST is extended as well. For example, with a Bluetooth-MOST-Gateway (if available), devices on the MOST network can be used as Hands Free Kits for mobile phones with Bluetooth interfaces. Furthermore, Dial-up networking can be made possible by the Bluetooth Function Block for MOST.

REFERENCES

[1] Jian-Xun Wu, "Exploration of MOST (Media Oriented System Transport) Network with Multimedia Applications," National Chiao Tung University, Master thesis, Nov. 2006.

[2] Oasis SiliconSystems, "MOST NetServices Layer 1 User Manual/Specification Rev. 1.10.x," Jan 2004.

[3] MOST Cooperation website, http://www.mostcooperation.com/home/index.html

[4] SMSC website, http://www.smsc-ais.com/AIS/

[5] MOST Cooperation, "MOST Specification Rev 2.5," Oct. 2006.

[6] MOST Cooperation, "MOST Specification Framework Rev 1.1," 1999.

[7] Oasis SiliconSystems, "VideoCompressor 4 MOST Technical Information Rev. 3.0," July 2002.

[8] Peter Ekström, Fredrik Hoel, "Audio over Bluetooth and MOST," Linköping University, Master thesis, March 2002.

[9] SMSC, "OS8104 MOST Network Transceiver Final Product Data Sheet," Sep. 2006.

[10] SMSC, "Product Flyer: RadioTuner 4 MOST V01 00 XX-9," July 2006.

[11] SMSC, "Product Flyer: Amplifier 4 MOST V01 00 XX-5," Nov. 2006.

[12] SMSC, "Product Flyer: DVDPlayer 4 MOST V01 02 XX-5," July 2006.

[13] Oasis SiliconSystems, "VideoDecoder 4 MOST V1.0.X User Manual," May 2004.

[14] SMSC, "Product Flyer: MOST PC Interfaces V01 00 XX-2," July 2006.

[15] Oasis SiliconSystems, "User Manual For MOST PCI Board Rev. 3.0," Dec. 2001.

[16] SMSC, "Product Flyer: OptoLyzer V02 00 XX-11," July 2006.

[17] Oasis SiliconSystems, "MOST Interface Control User Manual Rev. 1.4," Sep. 2003.

[18] SMSC, "MOST Radar V3.0.x User Manual," May 2006.

[19] Oasis SiliconSystems, "Application Note For MOST PCI Board V1.0-21," Nov. 2000.

[20] Microsoft Corporation, “GraphEdit Help,” 2002.

[21] MSDN webpage, “Introduction to DirectShow”, http://msdn2.microsoft.com/en-us/library/ms786508.aspx

[22] Oasis SiliconSystems, “RadioTuner 4 MOST V1.0.0 User Manual,” August 2003.

[23] Oasis SiliconSystems, “DVDPlayer 4 MOST V1.2.0 User Manual,” Oct. 2003.

[24] Oasis SiliconSystems, “Amplifier 4 MOST Version: 2.0 User Manual,” July 2003.

[25] SMSC, “Product Flyer: VideoDecoder 4 MOST V01 00 XX-3,” July 2006.

[26] SMSC, “MOST PCI Interfaces Cookbook V1.0.x User Manual,” Dec. 2006.

[27] MOST Cooperation, “Guideline of Transmission and Control for DVD-Video/Audio through MOST Bus Version: 1.0,” Nov. 2003.

[28] SMSC, “MOST NetServices V2.1.X Basic Example Application Note,” Dec. 2006.