NIS D333-1

Digital Video Learning Network V2 Interface Specification

Northern Lights the Way

northern telecom

™

NIS D333-1

Digital Video Learning Network V2 Interface Specification

Distribution Centre:
1-800-347-4850
P.O. Box 13010
Research Triangle Park, North Carolina
27709
Dept. 6611

Issue 1: May 27, 1993, first issue

northern telecom

Digital Video Learning Network V2 Interface Specification

NIS D333-1

Issue 1: May 27, 1993, first issue

NOTICE PAGE

Distribution Centre:
1-800-347-4850
P.O. Box 13010
Research Triangle Park, North Carolina
27709
Dept. 6611

REVISION HISTORY

Date (yy/mm/dd)	Issue information
93/05/17	first issue

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

SECTION 1
Introduction & Overview

This document describes the Digital Video Learning Network (DVLN) protocols, the coding parameters and the multiplexing structure to be used for the DVLN applications.

1.1 DVLN V2 Overview

DVLN is an application used to establish audio and video communication between up to 56 classrooms (assuming the cell is equipped with 2 quad split units). DVLN supports a single-cell where the classrooms are linked to a controller. DVLN V2 supports features such as multiple session scheduling, continuous presence class, multi-point class and broadcast class.

DVLN V2 does not support remote scheduling from the Classroom Interface. However, it is possible to perform scheduling from a different site. Remote and local machine control (for audio and video equipment) are not supported in DVLN V2 software.

Section 2 presents the basic interface and wiring diagram in a typical DVLN V2 application. Detailed physical interface for audio, video and data are described in a companion document. Section 3 presents the DVLN V2 protocol used between the Cell Controller and the Classroom Interface. Section 4 presents how the scheduling functions can be expanded from the Cell Controller to another site.

1.2 Video Codec Overview

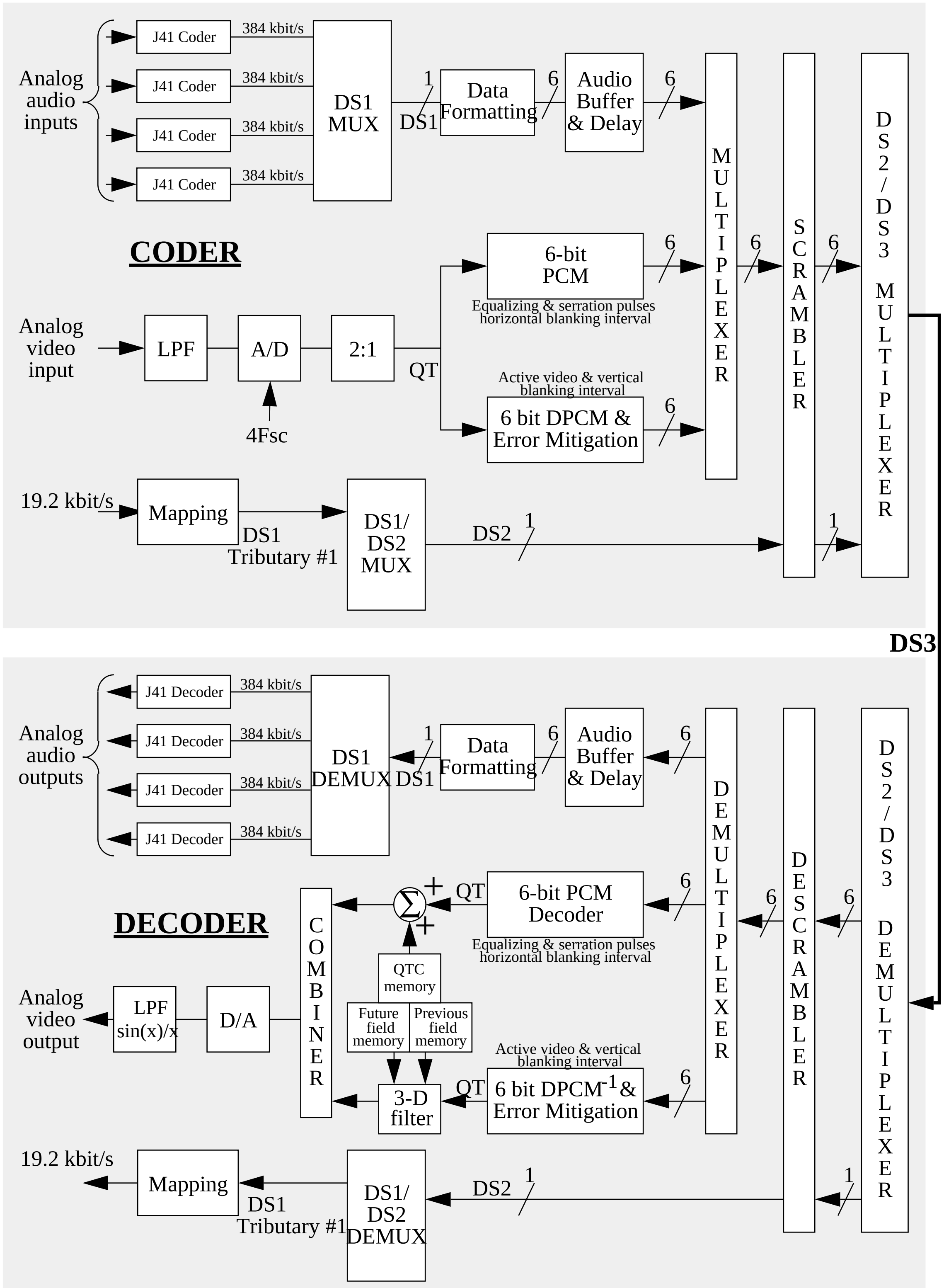
The video signal, audio signals, and the data channel are combined into a DS3 stream (44.736 Mbit/s). The basic architecture is presented in Figure 1.1:

- The analog video signal is filtered, clamped and sampled at 4 f_{sc} (f_{sc} = sub-carrier frequency, i.e. $\approx$ 3.58MHz) to generate an orthogonal signal. The orthogonal signal is subsampled at 2 f_{sc} using a field quincunx (QT) pattern and the QT signal is encoded using predictive/DPCM technique with channel adaptation to compensate for transmission errors. (Section 5)
- The coding of the audio signals as well as the generation of a DS1 which combines up to 4 audio signals are encoded using the CCITT J41 standard [1]. For each audio word (11-bit), a parity bit is included for transmission error detection. Up to 4 audio signals are combined together to generate a DS1 signal. The generation of the DS1 signal is fully described in [1] and [2]. The DS1 signal is converted into 6-bit words for multiplexing reasons. The 6-bit DS1 signal is buffered and delayed to ensure lip-sync. (Section 6)
- A 19.2 kbit/s data channel for connectivity between the DVLN controller and the classroom Macintosh is provided through the DS2 group 1 of the multiplexing structure. (Section 7)
- The 6-bit audio words and 6-bit video words are multiplexed together with framing information using one of the four possible “time-slot” formats. The 6-bit time-slot information stream is combined to the DS2 which contains the 19.2 kbit/s channel to create a 7-bit stream. The 7-bit stream is fed to a scrambler to provide immunity against transmission errors and is multiplexed with a M2-3 multiplexer in such a way that the DS2 which carries the data signal is mapped to DS2 group 1 of the M2-3 mux while the combined audio/video 6-bit time-slot is mapped to DS2 groups 2 to 7. (Section 8)

- X-bits in the DS3 stream are used by the coder to carry information relative to the alarms and controls of the DV45 boards. (Section 9)

At the receiver, the reverse processes are performed to recover the proper signals.

Figure 1.1 - Basic Architecture of the System

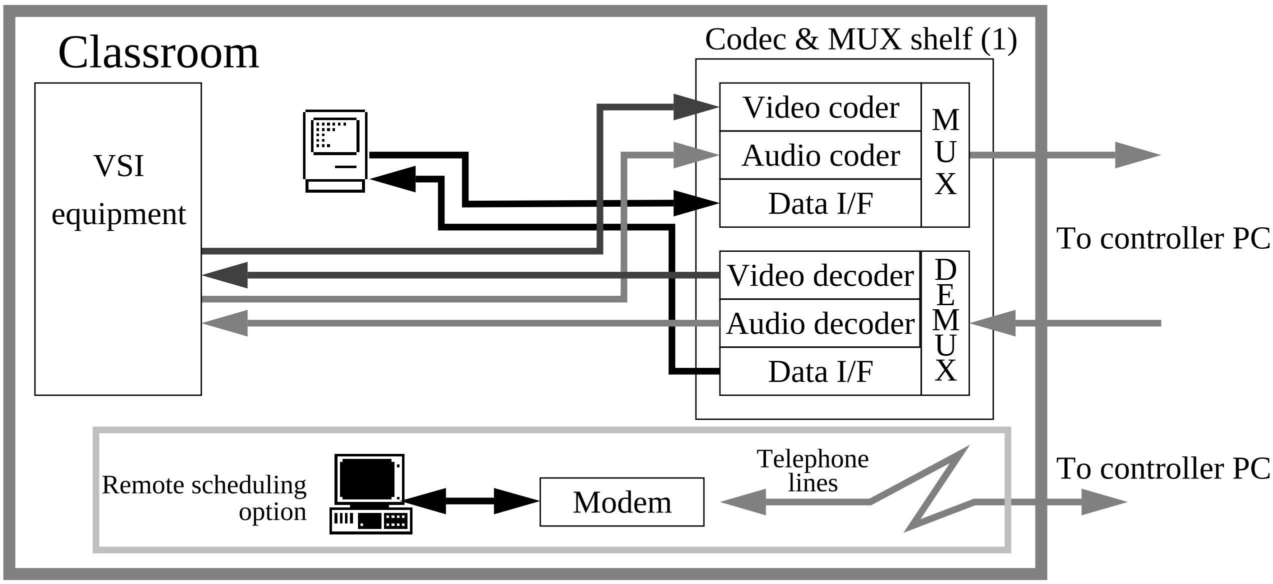


SECTION 2
DVLN V2 Basic Interface

This section provides a description of the various signals used in DVLN V2.

The following legend is used in the drawings

Figure 2.1 - Typical DVLN classroom equipment



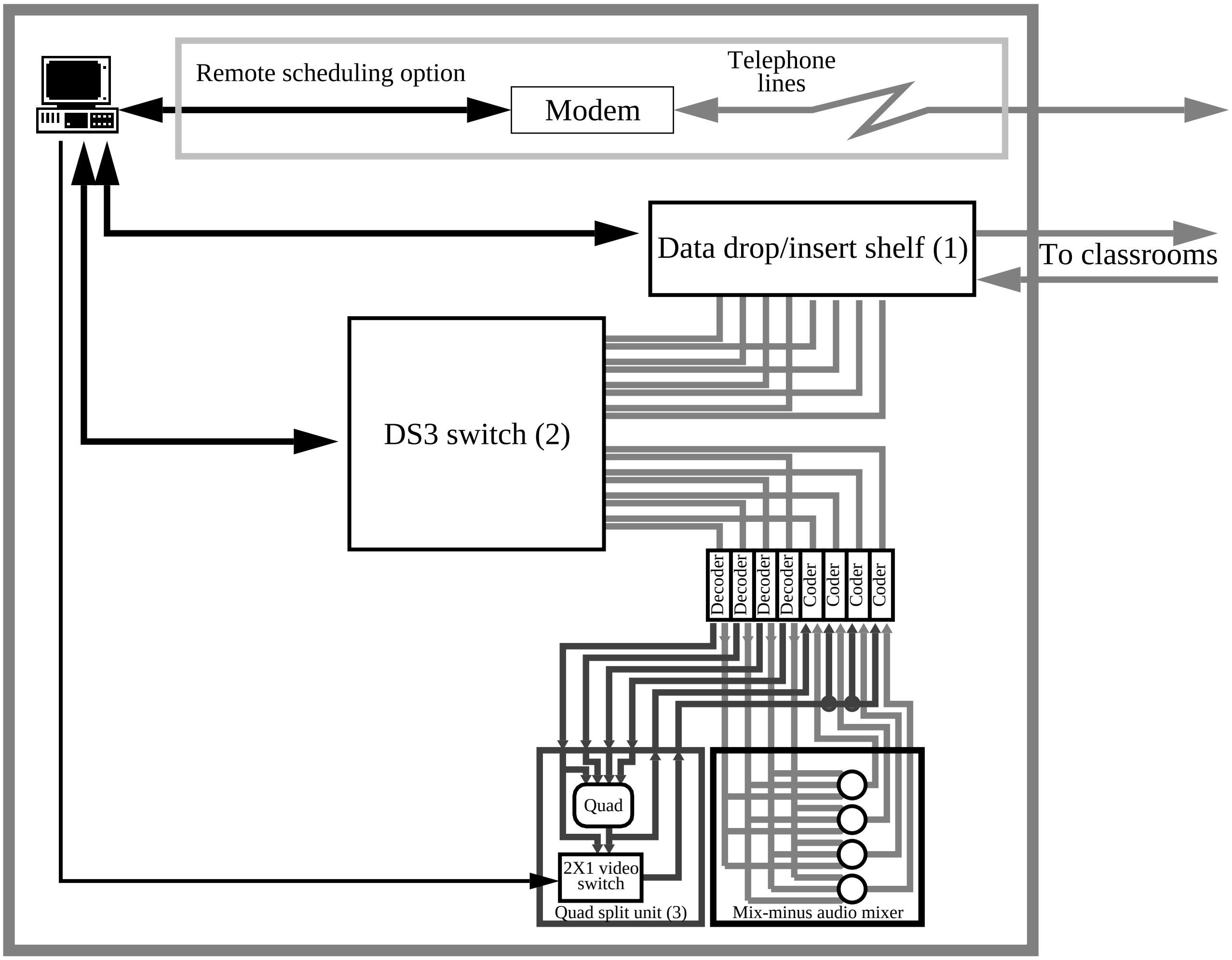
Macintosh to codec and MUX shelf
- Speed & format: 19.2kbaud, 8 bits, 1 stop, odd parity
- Proprietary protocol

Video input/output
- M/NTSC interface meets the EIA RS-170-A standard
- 75Ω unbalanced, BNC connector

Audio input/output
- 600Ω balanced
- Absolute gain 0dBm ±0.1dBm @ 1kHz nominal, 18dBm maximum with overload at 21dBm
- Bandwidth: 40Hz to 15kHz (+0.3dBm @ -1dBm)
- Compression algorithm meets CCITT J41 specifications

DS3 interface:
- 44.763 Mbit/s ± 20ppm
- 75Ω unbalanced
- B3ZS
- Compatible with DS3 frame construction
- Meets DSX3 interconnection specifications

Figure 2.2 - Typical DVLN switching site equipment



Controller to data drop/insert, to codec & MUX shelf and to DS3 switch

- Speed & format: 19.2kbaud, 8 bits, 1 stop, odd parity
- Proprietary protocol

Notes: (1) Typically Northern Telecom's DV45 equipment

(2) Typically di-tech DS3 switch

(3) Panasonic WJ-450 quad split unit

SECTION 3
DVLN V2 Controller to Classroom Interface Protocol

3.1 Data Link

The DVLN V2 data connection between the Controller and the Classroom Interface is described in the Section 7 of this document.

3.2 Data Format

All DVLN Protocol messages are encoded in ASCII. They begin with an "A" and are terminated with one <CR> character (0x0D) followed by one <LF> character (0x0A). Message parameters are always at a fixed offset from the start of the message.

Several basic types are used to encode data on the DVLN communication channels. The most basic type used is char. A char is a single ASCII character.

Integers are always encoded as ASCII characters. Two general purpose integer formats are defined by the DVLN protocol. Integer4 and integer8 are hexadecimal numbers. They are transmitted as their corresponding ASCII character or characters. Integer4 is a single character in length and encodes a 4-bit binary value ("0"-"F"). Integer8 is two characters in length and encodes an 8-bit binary value ("00"-"FF").

The special integer format "dvln_address" is used in the message header to encode the DVLN Address of network elements. dvln_address is four ASCII characters in length (e.g.: "1F0D").

The first two characters encode the DVLN cell number (00-FF), the last two characters encode the DV45 address of one DV45 shelf. Note that all switching site DV45 shelves have DV45 address 00.

dvln_address.cell and dvln_address.dv45 fields may each take the special values "--" or "**". "--" is interpreted by the data interface of the codec and the Controller as meaning "local". It is overwritten by the local cell number or DV45 address of the sender. "**" is interpreted as "broadcast".

In addition, the two special purpose time integer formats "times" and "timel" are defined. The short time type "times" is a 4 character ASCII coded decimal number of the form HHMM (e.g. "1303" encodes the time 13:03). The long time type "timel" is a 6 character ASCII coded decimal number of the form HHMMSS (e.g. "090359" encodes the time 9:03:59).

The bit array type provides a concise means of transmitting arrays of boolean values. A bit array looks identical to a sequence of integer4's on the serial link (e.g.: "7A1F"). A bit array is, however, an array of hexadecimal coded bits. The length of most bit arrays used in the V2 DVLN messages depends on the number of classrooms in the cell (bit arrays are generally of size 2 characters with 8 classrooms, 4 characters with 16 classrooms, 8 characters with 32 classrooms or 16 characters with 64 classrooms). The sender and receiver must know to interpret the bit array as an array of booleans instead of as an array of integer4's (or integer8's). Note that each character of the bit array is one of the ASCII characters 0, 1, 2,... 9, A, B, C,... F.

Strings are always sent as ASCII text strings of fixed length. The length of the each string used in the DVLN messages is indicated with a number following the word "string" (e.g. string9 is a string of total length 9 characters). If the information to be transmitted is shorter than the total size of the string field, it is padded with <NULL> characters. At least one <NULL> character always occurs at the end of a string. Thus the type string9 is large enough to contain a string of 8 characters at most (with one following <NULL>).

Booleans are transmitted as “T” or “F”.

All Application and Codec messages begin with five adjacent fields as follows:

“type dest source slot id” where:

char	type	“A” for Application Level messages
dvln_address	dest	destination (to) room number
dvln_address	source	source (from) room number
integer4	slot	the DCS slot number (0-7) on which the destination classroom is connected, or “*” for broadcast.
2 char	id	a two letter message id

The remainder of each message may contain message parameters (as required) followed by <CR><LF>.

3.3 Message Routing and Error Detection

Message rerouting is performed by the Controller to permit Mac<->Mac communication as will be required in V3.0. The Controller will retransmit any “A” message it receives whose “dest” field is not “00” on the appropriate DCS channel.

The data interface error correction is described in Section 7 of this document.

As an additional measure of protection, any time a Classroom Interface or the Controller receives a message, the following checks must be performed to determine if the message was received intact: the message must start with “A” and end with <LF>; the id of the message must be recognizable; and the length of the message (from “A” to <LF>) must match that expected for a message of the type determined by id (and, for xR and CN messages, by the integer8 which immediately follows the id field).

The Controller will update the Classroom Interface application regularly or when a screen change occurs. After each screen update message and depending on the “ack code” included in the message, the Classroom Interface will reply with a “RA” message. If 5 unsuccessful update attempts are made by the Controller (about 45 seconds of timeout), the room is declared “Not Responding” and the event is recorded in the Error log file. The Controller will continue to send screen update messages regularly to the room. If the Classroom Interface did not respond in the start of a class session and it did not receive the room names, the controller will continue to send the room names along with the screen update until the room responds with the proper “RA” reply. When the room is responding again (i.e. when it replies to the Controller with a “RA” message), it is declared “Responding” and the event is again recorded in the Error log file.

On the other side, if the Classroom Interface did not receive any screen update from the controller within 45 seconds, an error message is displayed on the screen telling the user that the “Controller is not responding”.

3.4 Protocol Messages

The message id field and the contents of each of the “Application Level” messages exchanged between the Macs and the Controller are defined below. In all the examples, “02” is the teacher classroom, “03” is one of the student classrooms and “00” is the controller. The 12 byte header (which includes the message id) and <CR><LF> are included in the “Total length” quoted for each message.

LogInRefresh **(id = “LR”)** **Teacher Mac <- Controller**

Parameters:

integer8	number of classrooms in the class (02-3F)
integer4	ack code: 0 = Mac should not respond with a RefreshAck 1 = “CN” messages follow - hold off RefreshAck any nonzero value must be returned in a RefreshAck
times (hhmm)	time at which class is scheduled to begin
times (hhmm)	time at which class is scheduled to end
timel (hhmmss)	current time
string19	title of the class
Total length	**50** bytes

Use:

LR is sent by the Controller to the Teacher Mac before the beginning of the class. The Teacher must log in to begin the class. Note that this message, like all xR messages, must be acknowledged by the Mac with an “RA” message if the ack code sent in the “LR” is nonzero (and the message was received without errors). If the ack code is “1” then the acknowledgment must be delayed until after all “CN” messages have been successfully received. If they are not all received, then no “RA” is sent. Any other nonzero ack code must be returned immediately in a “RA”.

Example:

```
A000200004LR04213001400112333Astrophysics-
304<NULL><NULL><NULL><CR><LF>
```

LogInRequest **(id = “LQ”)** **Teacher Mac -> Controller**

Parameters:

string9	access code
Total length	**23** bytes

Use:

LQ is sent by the Teacher Mac to the Controller when the teacher logs in.

Example:

```
A00000002*LQ1234567<NULL><NULL><CR><LF>
```

LogInFailed **(id = “LF”)** **Teacher Mac <- Controller**

Parameters:

None	
Total length	**14** bytes

Use:

LF is sent by the Controller to the Teacher Mac if the access code sent in an LQ was incorrect.

Example:

```
A000200004LF<CR><LF>
```

TeacherClassRefresh	**(id = “TR”)**	**Teacher Mac <- Controller**

Parameters:

integer8	number of classrooms in the class (as for “LR”)	
integer4	ack code (as for “LR”)	
bit array	classrooms responding (Size: 1-16 characters): 0 = classroom not responding 1 = classroom responding	
bit array	classrooms with a question pending (Size: 1-16 characters): 0 = classroom has no question pending 1 = classroom has question pending	
array of 8 integer8	order of the 8 first received questions: classroom number of 1st question (01-FF or 00 if no question) classroom number of 2nd question (01-FF or 00 if no question) … classroom number of 8th question (01-FF or 00 if no question)	
integer8	classroom currently previewed:	(01-FF or 00 if none)
integer8	classroom currently ready:	(01-FF or 00 if none)
integer8	classroom currently on air:	(01-FF or 00 if none)
times (hhmm)	time at which class is scheduled to end	
timel (hhmmss)	current time	
boolean	class overrun will be allowed	
boolean	class is currently overrunning	
boolean	DS3 switch is not responding	
Total length	**56** bytes for 8 classrooms, **60** bytes for 16 classrooms, **68** bytes for 32 classrooms, **84** bytes for 64 classrooms.	

Use:

TR is sent by the Controller to the Teacher Mac periodically as well as immediately as required in response to a change in the state of the system. Since more than one class can be given at any one time, a TR message must be sent to each teacher currently logged in.

Example:

```
A000200004TR1020080000F02010403000000000A0A022230215948FFF<CR><LF>
```

IdleRefresh	**(id = “IR”)**	**Unused Mac <- Controller**

Parameters:

integer8	number of classrooms in the class (as for “LR”)
integer4	ack code (as for “LR”)
times (hhmm)	next class start time
times (hhmm)	next class end time
timel (hhmmss)	current time
boolean	next class will be taught from this room
string19	title of the next class
Total length	**51** bytes

Use:

IR is sent by the Controller to Unused Classroom Macs periodically and immediately when a classroom Mac transitions to the Unused state. A classroom is considered to be Unused when it is not participating in a class (as either the Teacher or as a Student).

Example:

```
A000200002IR10211001200104556TAstrophysics-
304<NULL><NULL><NULL><CR><LF>
```

StudentClassRefresh **(id = "SR")** **Student Mac <- Controller**

Parameters:

integer8	number of classrooms in the class (as for "LR")
integer4	ack code (as for "LR")
integer4	cue state: 0 = no cue 1 = previewed 2 = ready 3 = on air 4 = Quad Split
integer4	port to mute on quad split unit (used in Q41 configuration)
times (hhmm)	time at which class is scheduled to end
timel (hhmmss)	current time
boolean	question requested
boolean	teacher classroom responding {for future use}
boolean	teacher logged in
Total length	**32** bytes

Use:

SR is sent by the Controller to the Student Macs periodically and as required in response to a change in the state of the system.

Example:

```
A000300003SR042102100204523TTT<CR><LF>
```

QuadSplitRefresh **(id = "QR")** **Teacher Mac <- Controller**

Parameters:

integer8	number of classrooms in the class (02, 03, or 04)
integer4	ack code (as for "LR")
bit array	classrooms responding (as for "TR", but Size = 1 character)
array of 4 integer8	order of any received questions (as for "TR" except 4 questions)
integer8	classroom currently previewed (as for "TR")
integer8	classroom currently ready (as for "TR")
integer8	classroom currently on air (as for "TR")
times (hhmm)	time at which class is scheduled to end
timel (hhmmss)	current time
boolean	class overrun will be allowed
boolean	class is currently overrunning
boolean	DS3 switch is not responding
Total length	**43** bytes

Use:

Since more than one class can be given at any time, a QR message must be sent to each teacher currently logged in and running a Quad Split (i.e. Continuous Presence) class session. QR is sent by the Controller to the Teacher Mac periodically as well as immediately as required in response to a change in the state of the system.

The Panasonic WJ-450 Quad Split device is used to perform the room zoom feature (Interact request type 1 in a Continuous Presence class session).When the teacher is zooming a class, a ground level voltage must be supplied to the connectors "4/1" & "1" of the "Remote IN" block located in the back of the WJ-450 unit to zoom the requested room. A voltage greater than 0.2 V or no connection should be supplied to restore the quad split screen to all the rooms.

Example:

```
A000200004QR042F01030000010112301159 48FFF<CR><LF>
```

ClassroomName **(id = “CN”)** **Mac <- Controller**

Parameters:

	integer8	number of classrooms in the class (as for “LR”)
	integer4	number of names in message (i.e. to follow) (1-8)
	integer8	first classroom number
	dvln_address	first classroom DVLN address
	string13	first classroom name
{	dvln_address	second classroom DVLN address
	string13	second classroom name
	dvln_address	third classroom DVLN address
	string13	third classroom name
	dvln_address	fourth classroom DVLN address
	string13	fourth classroom name
… }		
	Total length	**155** bytes for 8 names in message
		87 bytes for 4 names in message
		36 bytes for 1 name in message

Use:

Several CN messages are sent by the Controller to the classroom Mac (teacher, student or idle after an xR message with ack code = 1). The Mac must acknowledge (with RA) once messages containing names for all the classrooms in the class have been received successfully. If any names were not successfully received then no acknowledgment is to be sent.

Example:

```
A000200004CN044010001Clarksdale<NULL><NULL><NULL>
0003Corinth<NULL><NULL><NULL><NULL><NULL><NULL>
0005Philadelphia<NULL>
0006Boston<NULL><NULL><NULL><NULL><NULL><NULL><NULL><CR><LF>
```

RefreshAck **(id = “RA”)** **Refreshed Mac -> Controller**

Parameters:

integer4	(nonzero) ack code as received in last refresh
Total length	**15** bytes

Use:

RA is sent by all Macs to the Controller in response to a TR, SR, or IR message with nonzero ack code.

Example:

```
A00000002*RA2<CR><LF>
```

QuestionRequest **(id = “QQ”)** **Student Mac -> Controller**

Parameters:

boolean	on
Total length	**15** bytes

Use:

QQ is sent by the Student Classroom Macs to the Controller when the operator raises or cancels a question request.

Example:

```
A00000003*QQT<CR><LF>
```

CancelQuestionRequest **(id = “CQ”)** **Teacher Mac -> Controller**

Parameters:

integer8	classroom number at which pending question is to be cancelled (or 00 to cancel all questions)
Total length	**16** bytes

Use:

CQ is sent by the Teacher Mac to the Controller when the teacher cancels one or all of the currently pending questions.

Example:

```
A00000002*CQ03<CR><LF>
```

EndClassRequest **(id = “EQ”)** **Teacher Mac -> Controller**

Parameters: None

Total length	**14** bytes

Use:

EQ is sent by the Teacher Mac to the Controller when the Teacher requests that the current class be ended.

Example:

```
A00000002*EQ<CR><LF>
```

InteractRequest **(id = “IQ”)** **Teacher Mac -> Controller**

Parameters:

integer4	type of interaction: 0 = preview 1 = speak to student (may be used for teacher only zoom) 2 = listen to student (also used for everybody zoom) 3 = end interaction (also used to end zoom)
integer8	classroom number
Total length	**17** bytes

Use:

IQ is sent by the Teacher Mac to the Controller when the user previews or interacts with a room.

Example:

```
A00000002*IQ103<CR><LF>
```

SECTION 4
DVLN V2 Remote Scheduling & File Transfer Protocol

The DVLN Controller scheduling feature can be controlled remotely by using a standard VT100 terminal or a computer running a VT100 terminal emulation software (like Crosstalk XVI on a IBM-PC type computer or Smartcom II on a Macintosh) and a pair of dialup modems. The references of VT100 terminal emulation is described in [3].

The software needed on the Controller is Remote2. It's developed by Digital Communications Associates (DCA) and it is installed as a TSR (Terminate and Stay Resident) software running with DVLN. Remote2 will simply port the content of the Controller screen to the remote terminal and the remote terminal keyboard entries will be routed to the Controller. This will allow a person located anywhere to simply call the Controller's modem, enter a security password and monitor the system, perform backup and do master schedule management.

4.1 Configuration File Format

```
#===========================================================================
#
# DVLN Configuration File - Format Description
# --------------------------------------------
#
# Any line beginning with "#" is considered a comment.  Comments may
# appear anywhere in the file
#
# Data entries are tagged with a four letter code starting in the
# first column.  The following is a listing of all legitimate tags
# and their arguments.
#
#---------------------------------------------------------------------------
#
# ACCC string
#   |    +-> string of up to 8 digits or may be empty
#   +------> teacher access code tag
#
# IDLE room
#   |    +-> room number (0..; `X' - LOOPBACK)
#   +------> between class input tag
#
# SLOG value
#   |    +-> two bit indicator (bit0: error log, bit1: class log)
#   +------> screen logger tag
#
# RMAP room port shortname longname
#   |   |    |       |        +----> long room name of <= 13 chars
#   |   |    |       +-------------> short room name of <= 5 chars
#   |   |    |                       (Note: `_' represents a blank)
#   |   |    |
#   |   |    +---------------------> ds3 switch port (0..)
#   |   +--------------------------> dv45 room number/address
#   +------------------------------> room mapping tag
#
# QM41 quad# port1 port2 port3 port4
#   |    |     |     |     |     +-> fourth port for quad device
#   |    |     |     |     +-------> third port for quad device
#   |    |     |     +-------------> second port for quad device
#   |    |     +-------------------> first port for quad device (teacher)
#   |    +-------------------------> quad split device unit number
#   +------------------------------> quad 4x1 device id config tag
#   (Note: Manual editing required for initial device configuration)
#
# QM44 quad# port1 port2 port3 port4
#   |    |     |     |     |     +-> fourth port for quad device
#   |    |     |     |     +-------> third port for quad device
```

```
#   |    |     |     +-------------> second port for quad device
#   |    |     +-------------------> first port for quad device (teacher)
#   |    +-------------------------> quad split device unit number
#   +------------------------------> quad 4x4 device id config tag
#   (Note: Manual editing required for initial device configuration)
#
# QM45 quad# port1 port2 port3 port4 port5
#   |    |     |     |     |     |     +-> fifth port for quad device
#   |    |     |     |     |     |         (This value must be an "extra"
#   |    |     |     |     |     |          input port.  See note below.)
#   |    |     |     |     |     +-------> fourth port for quad device
#   |    |     |     |     +-------------> third port for quad device
#   |    |     |     +-------------------> second port for quad device
#   |    |     +-------------------------> first port for quad device
#   |    +-------------------------------> quad split device unit number
#   +------------------------------------> quad 4x4 device id config tag
#   (Note: Manual editing required for initial device configuration)
#
#
# Manual Edit Configuration Tags
# ------------------------------
#
# DGPT type port
#   |    |    +-> comm port to be assigned (1..; `X' - NO ASSIGNMENT)
#   |    +------> comm port assignment: 'R' - network Mac port
#   |                                   'S' - switch port
#   |                                   'L' - logger port
#   |                                   'Q' - quad split port
#   +-----------> digiboard comm port configuration information
#
# SWIP value
#   |    +-> number of input ports on the switch
#   +------> size of switch input tag
#
# SWOP value
#   |    +-> number of output ports on the switch
#   |        (This value may be ajusted automatically.  See note below.)
#   +------> size of switch output tag
#
# SPEW value-> Bit mask for spewing of logger messages (HEXADECIMAL)
#   |             bit[0]:   Tx & Rx poll messages
#   |             bit[1]:   Switch messages
#   |             bit[2]:   QS shelf messages
#   |             bit[3]:   Tx RMIF messages
#   |             bit[4]:   Rx RMIF messages
#   |             bit[5-7]: unused
#   +---------> Logger spew tag
#
#
# Note concerning the switch size and the quad split devices
# ----------------------------------------------------------
#
# The fifth port of a QM45 unit must be connected to one of the "extra"
# inputs of the switch.  The SWIP and SWOP determine the logical size of
# the switch.  The difference between the SWIP and the SWOP value determine
# the number of "extra" inputs of the switch.  One "extra" input is
# required for each QM45 unit.  The value of the SWOP will be ajusted
# (reduced) to the proper value if not enough "extra" inputs are available.
#
# It is NOT recommended to have many types of quad split unit in a system
# (the system should have only QM44, QM41 or QM45 units).  If many types
# are used, the quad split conflict error messages may not be appropriate
# to the given situation.
#
#===========================================================================
#
```

4.2 Master Schedule File Format

```
#======================================================================
#
# DVLN Master Schedule - Format Description
```

```
# ----------------------------------------
#
# Any line beginning with '#' is considered a comment.  Comments may
# appear anywhere in the file, except within a class entry.  Any
# comments placed before a class entry are considered associated with
# that entry.
#
# Data entries are tagged with a four letter code starting in the
# first column.  The following is a listing of all legitimate tags
# and their arguments.
#
#--------------------------------------------------------------------
#
# CRSE title-> string of up to 18 characters
# SEMR title->
#  +---------> class assignment: course/seminar type tag
#
# STAT status-> condition: `ok',`conflict',`deleted',`quad_conflict'
#  +----------> class entry status tag
#
# TCHR cell dv45-> dv45 address: 1..254
# STUD cell dv45-> dv45 address: 1..254, `ALL' : broadcast class
#  |    +--------> cell address: 1..254,
#  +-------------> teacher and student room tags
#
# DATE stt end-> end date for a course entry only: YYMMDD
#  |    +------> start date for a course entry only: YYMMDD
#  +-----------> date assignment tag for a course entry
#
# PERD date/dow stt end Q#-> qs device assignment: (1..)/'?'-qconflict
#  |    |    |   |   |  +--> quad split period tag
#  |    |    |   |   +-----> end time for the period   : HHMM
#  |    |    |   +---------> start time for the period : HHMM
#  |    |    +-------------> day of week (COURSE)      : (0-Sun..6-Sat)
#  |    +------------------> date for period (SEMINAR) : YYMMDD
#  +-----------------------> period designation tag
#
# QUAD-> quad split class designation tag
#
# END-> end of class entry tag
#
#--------------------------------------------------------------------
#
# File Support Tags
# -----------------
#
# LAST_VERIFIED-> Date and time of last verification of this file
#                   (eg. LAST_VERIFIED MM/DD/YY HH:MM:SS)
#
# END_OF_HEADER-> End of header tag
#
#====================================================================
#
#
END_OF_HEADER
```

SECTION 5
Video Signal Processing

5.1 Analog Video Interface to Digital Mapping

5.1.1 Analog Input/Output

The analog video input (at the coder) and output (at the decoder) correspond to the EIA RS-170-A [4] for M-NTSC (i.e. frame rate = 29.97 Hz), 525 lines, 2:1 interlace system.

5.1.2 Low Pass Filtering

Prior to digitization, the analog video input is filtered to remove high frequencies that might cause aliasing during the A/D process. A typical filter will have the following characteristics:

Passband: 4.2 MHz

Stopband: > 45 dB from 6 MHz to 100 MHz.

At the receiver an equivalent filter with sin(x)/x compensation is required after the D/A conversion to retain the information in the 0 to 7.16 MHz band.

5.1.3 Sampling Parameters

The basic sampling parameters of the analog video input are:

Sampling frequency:	$4 f_{sc}$
Sampling structure:	Orthogonal
Sampling phase	Locked to the burst at 0° phase
Number of samples per line:	910
Number of samples per active line:	758 (centered in the active video area)
Number of samples in the HBI:	152

5.1.4 Analog to Digital Conversion

The relationship between video signal levels (nominal values) and the 9-bit quantization levels are:

Sync level (-40 IRE):	8
Blanking level (0 IRE):	120
White level (100 IRE):	400

Other levels can be obtained by using Equation (EQ 1).

$$\text{9-bit level} = 2.8 \times \text{IRE level} + 120 \qquad \text{(EQ 1)}$$

5.1.5 Time-Code Bit Generation

A time-code bit is generated at the coder for accurate clock recovery at the receiver. A counter is counting every $2 f_{sc}$ clock pulse. Whenever the count reaches 454 [1], another free-running counter clocked at half of the DS3-clock rate is reset. At the moment when the half-DS3

1. The counter is assumed to count starting from 0.

counter is reset, the negation of the LSB of this counter is the time-code bit (i.e. if the count=1420, then the time-code bit is represented by 1. If the count=1421, it is represented by 0.

Bit 1 of the very last word of every line is used to convey this bit information (Figure 8.1). The repetition rate of this time-code bit is the video line rate (about 15.734 kHz).

5.2 Video Coding Algorithm

The coder functions are described in details in this Section. Further details are described in [5]. Functionality of the decoder corresponds to the inverse functions.

The coding algorithm compresses efficiently the video information by segmenting the video signal into the following components which have different characteristics:

- the active video (758 pixels/line, lines 22 to 262 and the equivalent lines in the other fields) where one can assume temporal and spatial correlation;
- the Horizontal Blanking Interval (HBI, 152 pixels/line, lines 10 to 262 and the equivalent lines in the other fields) which is easily predictable;
- the pre-equalizing pulses, serration pulses, and post-equalizing pulses (910 pixels/line, lines 1 to 9 and the equivalent lines in the other fields) which are deterministic;
- VITS, Teletext and the like (758 pixels/line, lines 10 to 21 and the corresponding lines in the other fields) where only horizontal correlation can be assumed. In the DVLN application none of these signals are used. Information is provided to ensure compatibility with the current hardware implementation without the need to establish the different method for coding these intervals.

Table 5.1 as well as Figure 5.1 show how each segment is processed.

5.2.1 QT Decimation and the Reconstructor

The orthogonal video signal is 2:1 subsampled by using a field quincunx (QT) pattern. The subsampling pattern is presented in Figure 5.2.

Figure 5.1 - Subsampling Pattern

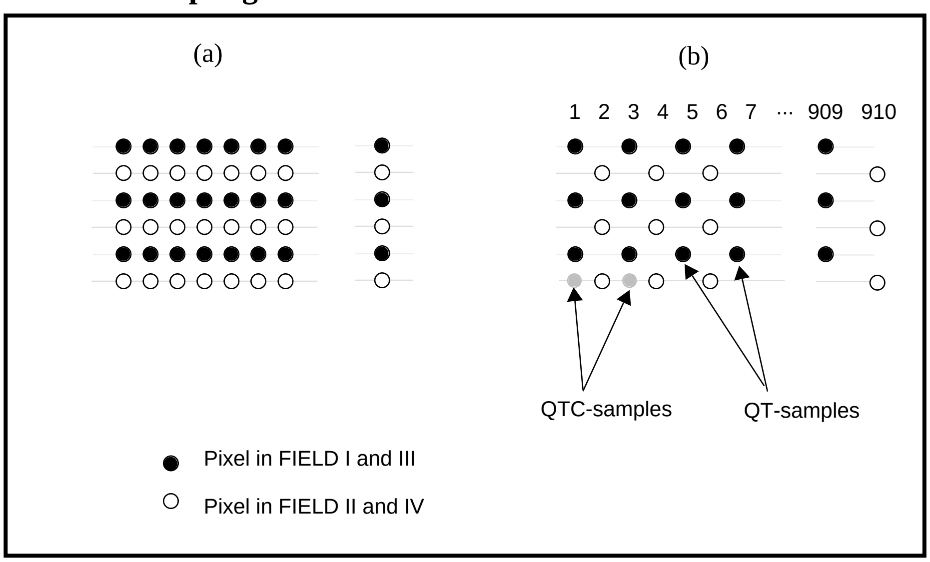
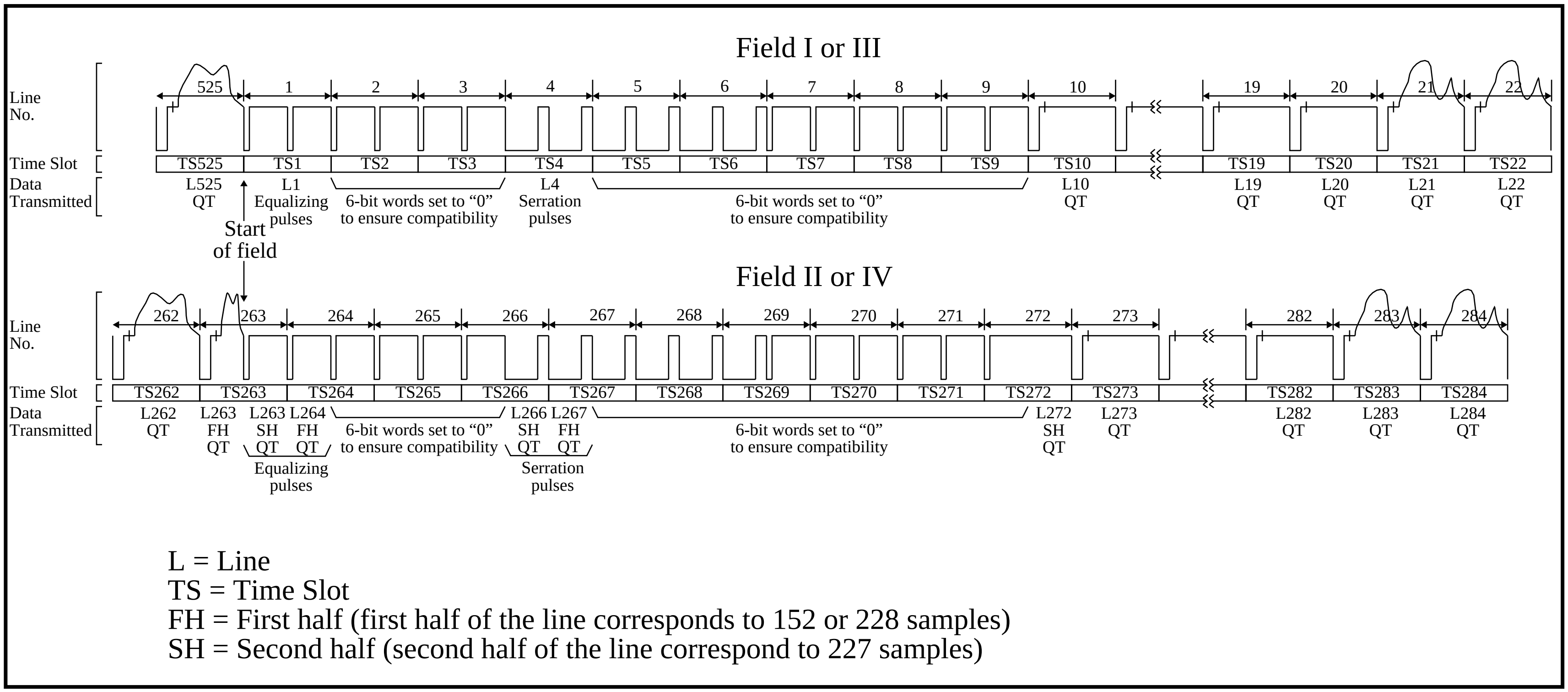


Table 5.1 - Method for Encoding the Video Segments

Time slot	Flag code	Encoding of active pels	Data transmitted	Line no.	Reconstruction of (QTC) pels
1	SOF	7-bit PCM quality	QT Line 1	1	QTC restored from past TS263SH and TS264FH
2	X	-	6-bit word “0” generated (video)	2	Copy of reconstructed equalizing pulses
3	X	-	6-bit word “0” generated (video)	3	Copy of reconstructed equalizing pulses
4	SYNC	7-bit PCM quality	QT Line 4	4	QCT restored from past TS266SH and TS267FH
5	X	-	6-bit word “0” generated (video)	5	Copy of reconstructed equalizing pulses
…	…	…	…	…	…
9	X	-	6-bit word “0” generated (video)	9	Copy of reconstructed equalizing pulses
10	SYNC	6-bit DPCM	QT Line 10	10	Black line generated at the decoder
11	SYNC	6-bit DPCM	QT Line 11	11	Black line generated at the decoder
12	X	6-bit DPCM	QT Line 12	12	Black line generated at the decoder
…	…	…	…	…	…
21	X	6-bit DPCM	QT Line 21	21	Black line generated at the decoder
22	X	6-bit DPCM	QT Line 22	22	3-D Interpolation filter
…	…	…	…	…	…
262	X	6-bit DPCM	QT Line 262	262	3-D Interpolation filter
263	X	6-bit DPCM	QT Line 263FH	263FH	3-D Interpolation filter
	-	7-bit PCM quality	QT Line 263SH	263SH	QTC restored from past TS1
264	SYNC		QT Line 264FH	264FH	
		-	6-bit work “0” generated (video)	264SH	Copy of reconstructed equalizing pulses
265	X			265FH	
		-	6-bit work “0” generated (video)	265SH	Copy of reconstructed equalizing pulses
266	X			266FH	
		7-bit PCM quality	QT Line 266SH	266SH	QTC restored from past TS4
267	SYNC		QT Line 267FH	267FH	
		-	6-bit work “0” generated (video)	267SH	Copy of reconstructed equalizing pulses
268	X			268FH	
		-	6-bit work “0” generated (video)	268SH	Copy of reconstructed equalizing pulses
269	X			269FH	
		-	6-bit work “0” generated (video)	269SH	Copy of reconstructed equalizing pulses
270	X			270FH	
		-	6-bit work “0” generated (video)	270SH	Copy of reconstructed equalizing pulses
271	X			271FH	
		-	6-bit work “0” generated (video)	271SH	Copy of reconstructed equalizing pulses
272	X			272FH	
		7-bit PCM quality	QT Line 272	272SH	Copy of QT samples
273	SYNC	6-bit DPCM	QT Line 273	273	Black line generated at the decoder
274	SYNC	6-bit DPCM	QT Line 274	274	Black line generated at the decoder
275	X	6-bit DPCM	QT Line 275	275	Black line generated at the decoder
…	…	…	…	…	…
284	X	6-bit DPCM	QT Line 284	284	Black line generated at the decoder
285	X	6-bit DPCM	QT Line 285	285	3-D Interpolation filter
…	…	…	…	…	…
525	X	6-bit DPCM	QT Line 525	525	3-D Interpolation filter

Figure 5.2 - Alignment of the Video Lines in the Time Slots



In the QT pattern, a 2:1 horizontal subsampling takes place by keeping every other pel. Within a field, the pels kept by the decimator are vertically aligned, producing an orthogonal pattern. From field to field, the sampling phase is offset by one pel; thus, the pels kept in one field are aligned with the pels discarded in the previous and the following fields. The sub-sampling phase changes from field to field, generating a field quincunx pattern. The odd pels (1, 3, 5, ... , 909) of each line are kept in fields I and III; the even pels (2, 4, 6 , ... , 910) of each line are kept in fields II and IV. Two 2 f_{sc} streams are generated by the 2:1 sub-sampling. The first stream is the QT information (pels transmitted in the given field) and the second stream is the QT complementary (QTC) information (pels discarded).

At the decoder, different techniques (Table 1) are applied to reconstruct the missing pels in the various signal components depicted in Section 5.2. These techniques might be different from one implementation to the other because they are not required to ensure compatibility. Examples of such techniques are:

- The QTC information associated to the serration and equalizing pulses, and to the HBI can be restored from a memory (the missing information was transmitted during the previous field).
- Black lines can be generated for lines 10 to 21 as they are not used in the DVLN application. However, it is important to generate proper information at the coder to ensure compatibility with the existing hardware.
- At the decoder, a 3D interpolation filter can be used to reconstruct the QTC information during the active part of the video signal. For example, a simple filter is described by the impulse response presented in Equation (EQ 2). The filter applies to the 262nd previous line, the current line, and the 262nd following line as shown in Figure 5.3.

$$h(n) = \frac{1}{16}\begin{bmatrix} 0 & -4 & 0 & 8 & 0 & -4 & 0 \\ 1 & 0 & 7 & 16 & 7 & 0 & 1 \\ 0 & -4 & 0 & 8 & 0 & -4 & 0 \end{bmatrix} \qquad \text{(EQ 2)}$$

5.2.2 PCM Coding

As identified in Table 1, the Horizontal Blanking Interval and the serration/equalizing pulses are coded in 6-bit PCM with a quality equivalent to 7-bit PCM.

Equalizing and serration pulses

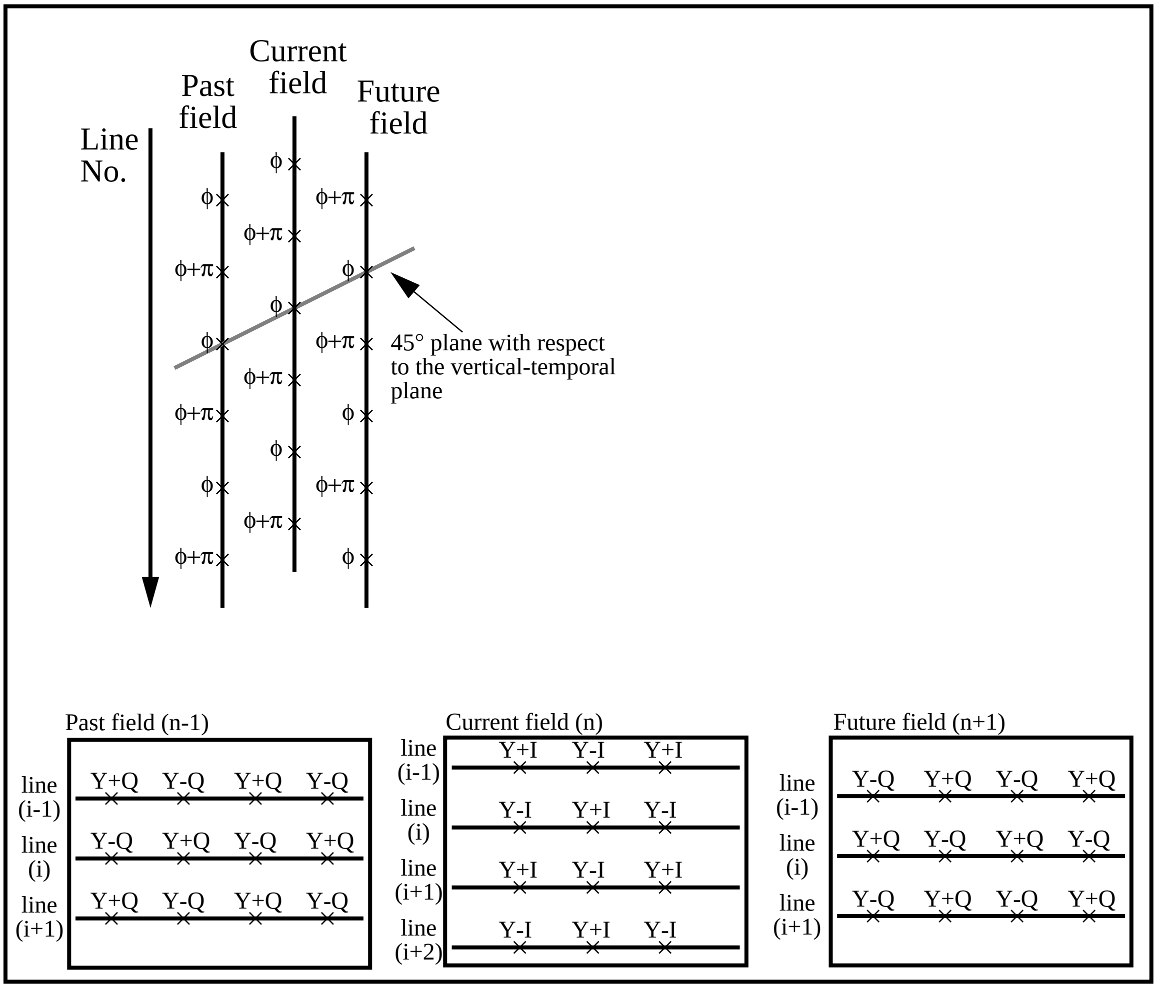
The equalizing pulses (lines 1, 2, 3, 7, 8, 9 in field I or III, and the corresponding lines in field II or IV) as well as the serration pulses (lines 4, 5, 6 in field I or III, and the corresponding lines in II or IV) are processed in the following way. One line of equalizing pulses and one line of serration pulses is sent per field. In field I or III, line 1 is sent during time-slot 1 (time-slot notion is introduced in Section 8.1), while line 4 is sent during time-slot 4. In field II or IV, the second half of line 263 and first half of line 264 are sent during the second half of time-slot 263 and first half of time-slot 264, while the second half of line 266 and first half of line 267 are sent during the second half of time-slot 266 and first half of time-slot 267.

Only the QT information (455 pels) of the equalizing and serration pulses is transmitted. The QT information is encoded 6-bit PCM with a quality equivalent to 7-bit PCM by dropping the

2 lsb's (least significant bit) and the msb (msb is never active during these intervals) from the 9-bit PCM signal.

To allow for the full lines (455 pels) of equalizing and serration pulses, flag code SOF (see Section 8.2) is forced in time-slot 1, and flag code SYNC is forced in time-slots 4, 264, and 267.

Figure 5.3 - Region of Operation of the 3-D Interpolation Filter



Horizontal Blanking Interval

To render the compression algorithm more efficient, the horizontal blanking intervals (HBI) are transmitted a limited number of times per field.

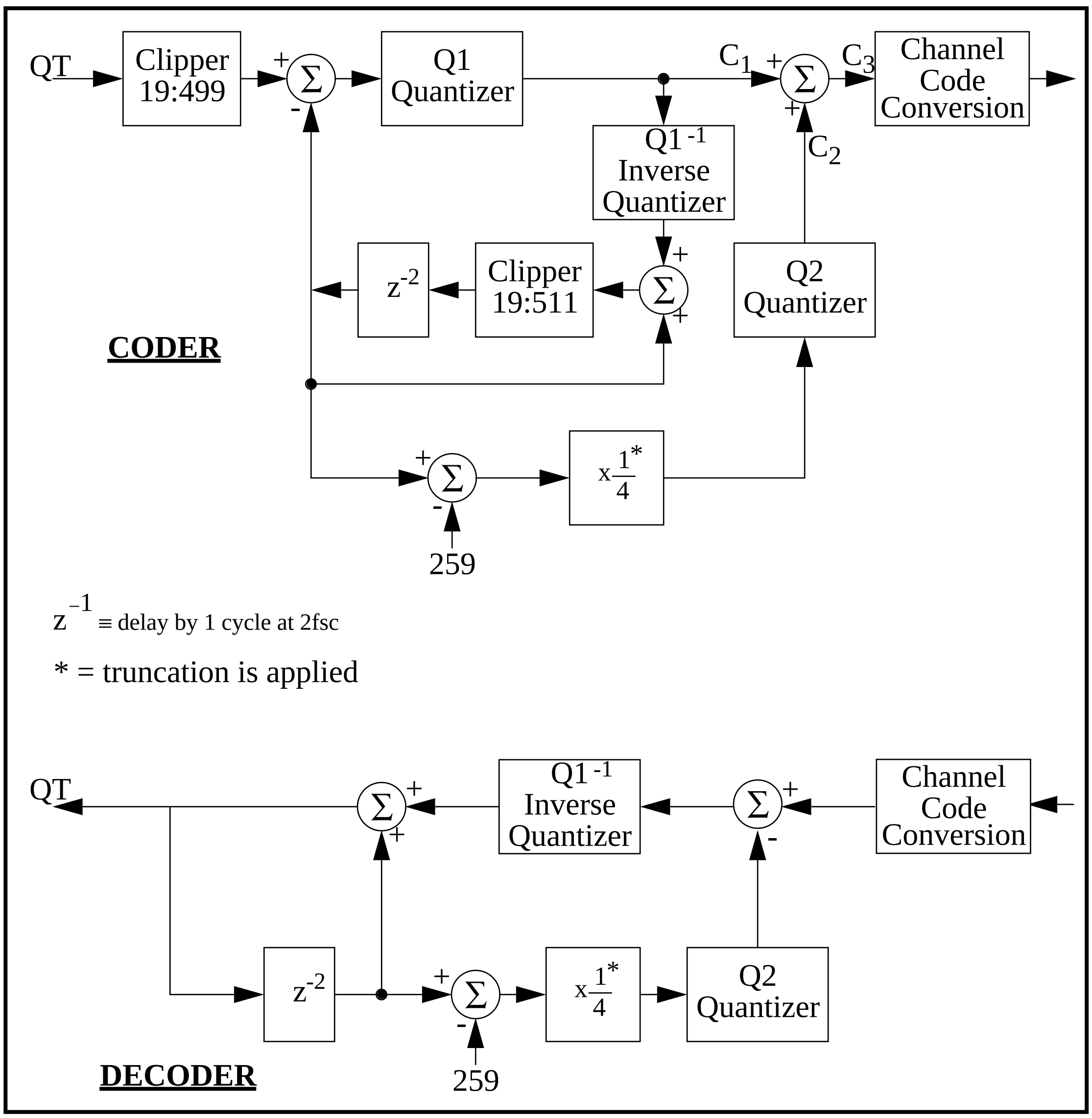
The QT information corresponding to the HBI represents 76 pels. The QT information is encoded 6-bit PCM with a quality equivalent to 7-bit PCM by dropping the 2 lsb's and the msb from the 9-bit PCM signal.

In addition to line 1, 4 (and the corresponding lines in the other fields) where the HBI is transmitted as part of the serration and equalizing pulses, the HBI is always transmitted for lines 10, 11 and the corresponding lines in the other fields as these lines are the two first HBI of the field that contain the two phases of the color burst. Other HBI are transmitted to regularize the buffer content (rules are defined in Section 8.2).

5.2.3 DPCM Coding with Channel Adaptation

The 379 QT pixels representing the VBI (vertical blanking interval) and active video information (lines 10 to 262 and the equivalent lines in the following fields) are coded in 6-bit DPCM mode incorporating circuitry for transmission error mitigation. The block diagram is shown in Figure 5.4. The DPCM predictor is the 4th previous reconstructed pel (when the signal is sampled at 4 f_{sc}). The DPCM prediction errors are non-uniformly quantized with 6 bits using Table 5.2. Only the positive halves are tabulated. The negative halves are the mirror images of the corresponding positive ones and are represented in 2's complement format. For robustness against transmission errors, a reset is done at the beginning of each active line by setting to "128" the prediction value of the two first samples of the line to be DPCM encoded.

Figure 5.4 - Details of the Video Coding Algorithm



The second loop adds another channel code C_2 (Table 5.3) to C_1. The Q1 and Q2 tables are designed such as the summation of C_1 and C_2 is also included between -31 and 31. If the system works in an error free environment, the second loop has no effect. If transmission errors occur, the effect is that the error will decay and will be invisible after a few pels.

Table 5.2 - Q1 Table (channel code and representative value) †.

Input range of prediction error from	to	6-bit channel code (C_1)	representative value of prediction error
0	0	0	0
1	1	1	1
2	2	2	2
3	4	3	3
5	6	4	5
7	8	5	7
9	11	6	10
12	14	7	13
15	18	8	16
19	22	9	20
23	27	10	25
28	32	11	30
33	38	12	35
39	44	13	41
45	51	14	48
52	59	15	55
60	67	16	63
68	77	17	72
78	87	18	82
88	98	19	93
99	110	20	104
111	123	21	117
124	137	22	130
138	153	23	145
154	169	24	161
170	187	25	178
188	206	26	196
207	226	27	216
227	248	28	237
249	272	29	260
273	298	30	285
299	511	31	311

† Note: Only the positive half is listed. The negative half is identical to the positive half except for the sign.

5.2.4 Channel Code Conversion

The 6-bit PCM and 6-bit DPCM channel code (i.e. the summation of C_1 and C_2) might emulate the flag-sign value (hexadecimal value “3F”, see Section 8.2). To prevent this emulation as much as possible, all channel codes (part of the video information in either in 6-bit PCM or 6-bit DPCM mode) with a sign bit set will have their amplitude bits inverted at the coder (Figure 5.4). The reverse process is done at the decoder.

Table 5.3 - Q2 Table (channel code) ‡.

Input range of prediction value		6-bit channel code (C2)
from	to	
0	0	0
1	3	1
4	6	2
7	9	3
10	12	4
13	15	5
16	18	6
19	21	7
22	25	8
26	29	9
30	32	10
33	35	11
36	38	12
39	40	13
41	43	14
44	45	15
46	47	16
48	48	17
49	50	18
51	51	19
52	53	20
54	54	21
55	55	22
56	56	23
57	57	24
58	58	26
59	59	28
60	63	31

‡ Note: Only the positive half is listed. The negative half is identical to the positive half except for the sign.

SECTION 6
Audio Signal Processing

6.1 Audio Coding

The audio signals are coded using the CCITT J41 algorithm. A parity bit is added to each 11-bit sample for protection against transmission errors. The 12-bit samples are produced at a sampling rate of 32 kHz to generate a 384 kbit/s stream per audio channel. Four audio channels are combined together to create a 1.544 Mbit/s stream. The audio coding and generation of the DS1 signal is fully described in [2]. For applications where only one audio channel is required, DS0 groups 1 to 6 are used; if two audio signals are required, the second channel will use DS0 groups 7 to 12; etc.

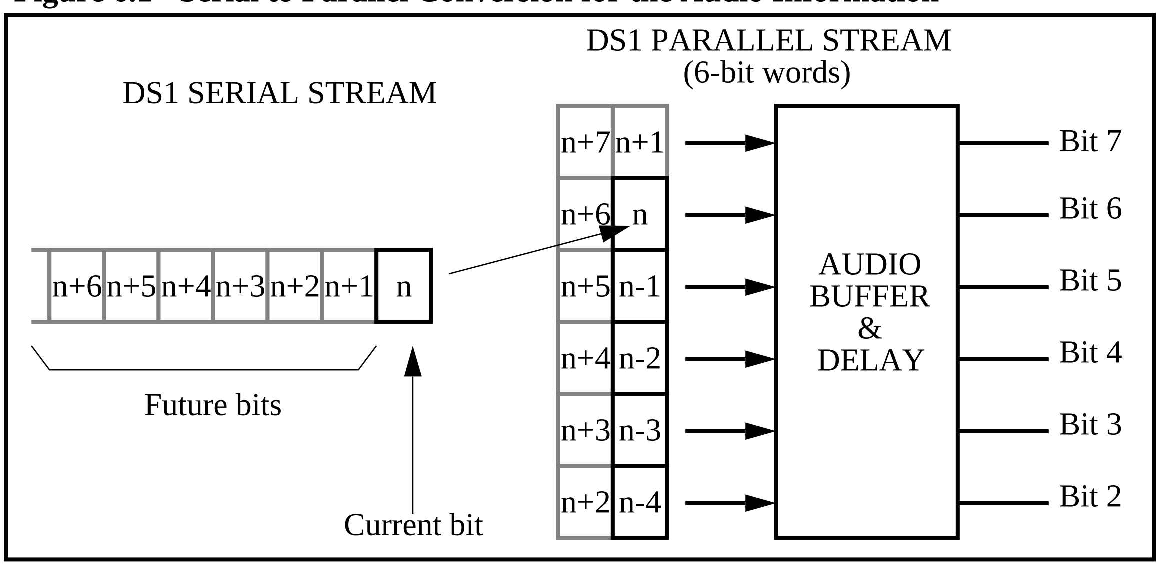
6.2 DS1 Serial to Parallel Conversion

For multiplexing reasons, the DS1 is converted to a six-bit parallel stream as depicted in Figure 6.1.

6.3 Audio Buffer

The 6-bit words are stored in a buffer, and recalled and combined to other signals in group of 24 six-bit words when slots are made available (see Section 8.2). The audio buffer should also perform delaying functions to ensure "lip-sync" with the video signal. The video processing delay is approximately 17 msec. The audio delay must be programmable (in steps of 2 msec or less) to ensure that "lip-sync" could be established within 1 msec.

Figure 6.1 - Serial to Parallel Conversion for the Audio Information



SECTION 7
Data Signal Processing

7.1 DVLN Data Format

The DVLN Classroom Interface (Macintosh) communicates with the DVLN Controller (PC) at 19.2 kbaud. The DVLN connection is described by the following parameters:

- Asynchronous, 19.2 kbaud,
- 1 start bit,
- 8 data bits,
- 1 stop bit,
- no flow control.

The packet size varies between 14 bytes and 256 bytes (see Section 3.2). There is no cyclic redundancy check (CRC) done at that stage.

7.2 HDLC Protocol

The 19.2 kbaud information (data and flag codes) is transferred using the HDLC protocol. HDLC frames are shown in Figure 7.1, note that the address and control fields are not used (HDLC option). The HDLC channel is running at 384 kbit/s and CRC is fully described in [6] and is as per Equation (EQ 3). The start & stop bits of the DVLN protocol are not included in the information part of the HDLC packet.

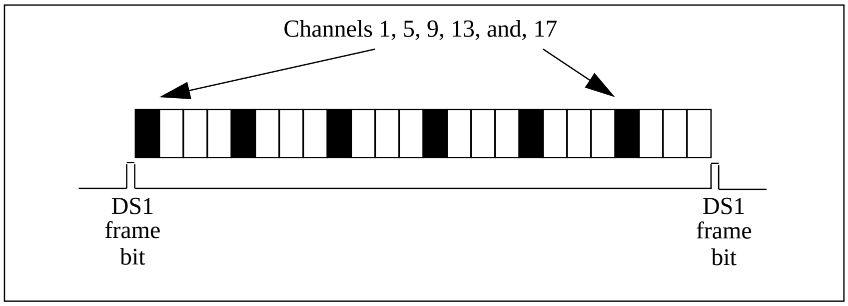
Figure 7.1 - Modified HDLC Format

Opening Flag	Information	CRC	Closing Flag

7.3 384 kbit/s Channel to DS1 Mapping

The 384 kbit/s connection used by the HDLC channel is established by using time-slots 1, 5, 9, 13, 17, and 21 within the DS0/DS1 multiplexer using the normal DS1 framing (Figure 7.2). Data is transmitted LSB first.

Figure 7.2 - Mapping of the 384 kbit/s Channel



SECTION 8
Multiplexing of the Information

8.1 Time-Slot Formats

Figure 1.1 shows the basic multiplexing structure. A time division multiplex first combines the 6-bit audio stream, the 6-bit video stream, and insert flagging information into time- slots.

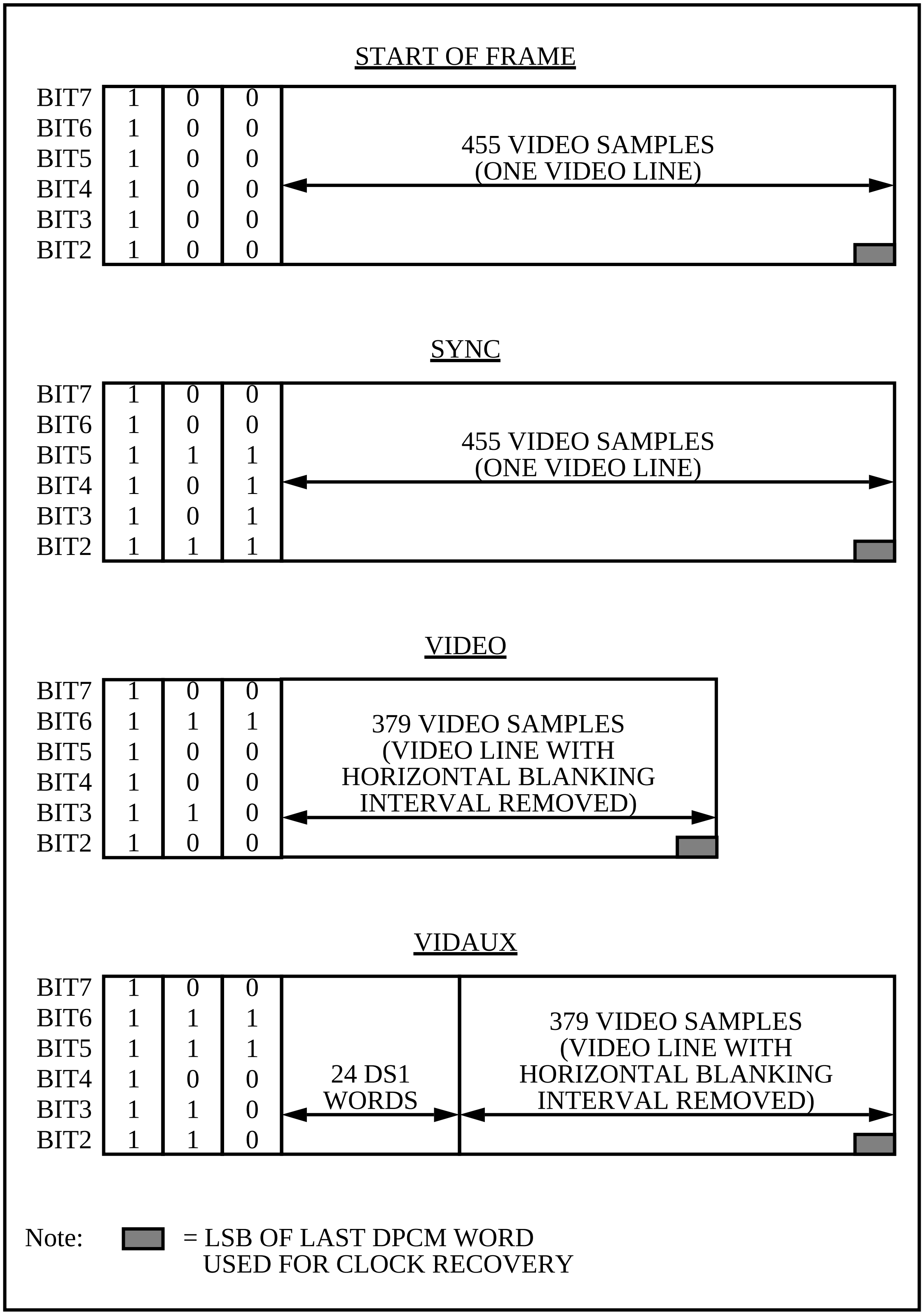
The time-slot formats are presented in Figure 8.1. Four different formats are specified. Each time-slot starts with a flag sign (hex “3F”) and are immediately followed by two others 6-bit words which define the time-slot format being used. The flag sign is a distinctive codeword which cannot be emulated by the coded video information. This is for two reasons: to limit the number of one densities, and, to accelerate the resynchronization during power-up and after a loss of synchronization. The lsb of the last 6-bit word of each line is used to carry the time-code information. The time-slot formats are the following:

- SOF (Start Of Frame): Indicates a new frame (selected only at line 1 of fields I and III), the horizontal blanking interval is transmitted, the corresponding time-slot is 458 words long.
- SYNC: Indicates that the horizontal blanking interval is transmitted, the corresponding time-slot is 458 words long.
- VIDEO: Indicates that the horizontal blanking interval is removed, the corresponding time-slot is 382 words long.
- VIDAUX: Indicates that the horizontal blanking interval is removed, and, that 24 six-bit words of the DS1 information follow, the corresponding time-slot is 406 words long.

The arrangement of the data that follows the three flag words is:

- DS1 audio signal: The 24 six-bit words occupy bits 2 to 7 for 24 consecutive DS2 slots. Bit 2 corresponds to the lsb of the 6-bit word while bit 7 corresponds to the msb.
- HBI: The 76 six-bit PCM words (7-bit PCM quality) occupy bits 2 to 7 for 76 consecutive DS2 slots. Bit 2 corresponds to the lsb of the 6-bit word while bit 7 corresponds to the msb.
- Serration and Equalizing Pulses: The 455 six-bit PCM words (7-bit PCM quality) occupy bits 2 to 7 for 455 consecutive DS2 slots. Bit 2 corresponds to the lsb of the 6-bit word while bit 7 corresponds to the msb.
- Time-slots where encoding is 6-bit DPCM: The 379 six-bit DPCM words occupy bits 2 to 7 for 379 consecutive DS2 slots. Bit 2 corresponds to the lsb of the 7-bit word while bit 7 corresponds to the msb.

Figure 8.1 - Time-Slot Formats



8.2 Rules for Selecting Time-Slot Formats

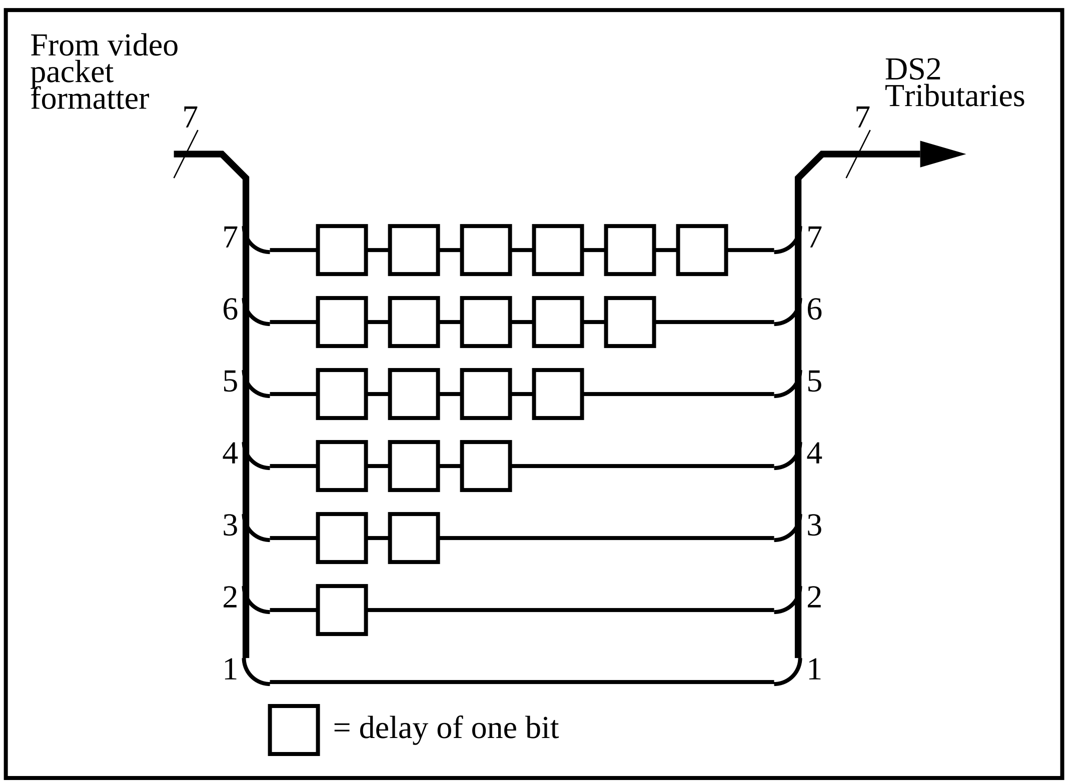
The rules for selecting the appropriate time-slot format are the following (they are listed sequentially):

1. If video line number = 1, then SOF is selected. End.
2. Else, if video line number = 4, 10, 11, 264, 273, or 274 then SYNC is selected. End.
3. Else, if the DS2 FIFO contains less than 128 words, then SYNC is selected. End.
4. Else, if at least 24 six-bit words of the DS1 stream are present in the buffer, then VIDAUX is selected. End.
5. Else, VIDEO is selected. End.

8.3 Bit Scrambler and Descrambler

The 7-bit stream (6-bit time-slot and the 1-bit data circuit) goes through a bit scrambler (Figure 8.2) at the coder before being multiplexed at DS3 level. The purpose of this scrambler is to spread the flagging information in the DS3 serial stream. As a consequence, short transmission burst errors do not corrupt the flagging information. A descrambling process is done at the decoder. The scrambler delays DS2 group 7 by 6 bits, DS2 group 6 by 5 bits and so on.

Figure 8.2 - Bit Scrambler



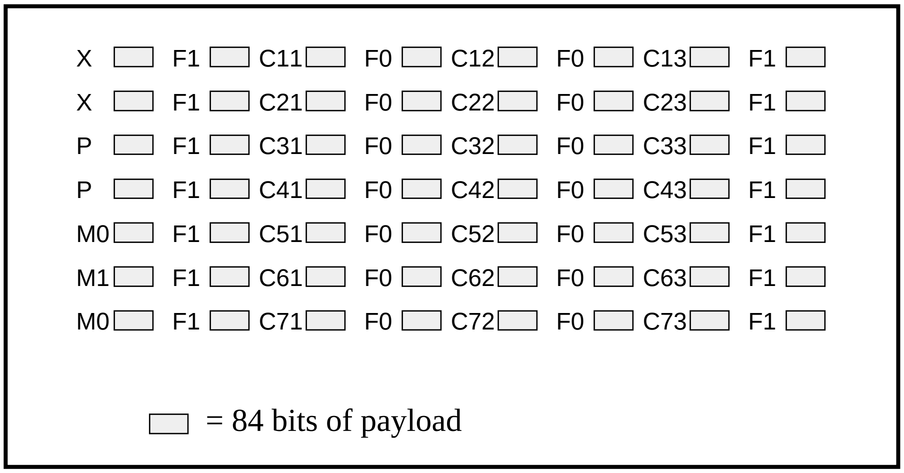
8.4 Multiplexing Structure

The 7-bit stream out of the scrambler is read at the DS2 rate and fed to M2-3 multiplexer. The M2-3 multiplexer combines the seven 6.312 Mbit/s streams using the normal DS3 structure

(Figure 8.3) with the normal DS3 framing.

The stuff bit position for each DS2 never carries valid information and should be ignored at the receiver. The stuff-control bits (Cij where i = {1, 2, ... , 7} and j = {1, 2, 3}) in the DS3 frame are set to 0. The control channel (bit 1) is resident in the DS2 group #1 and the Video/Audio (bit 2-7) are resident in the groups #2 to #7 respectively.

Figure 8.3 - DS3 Format

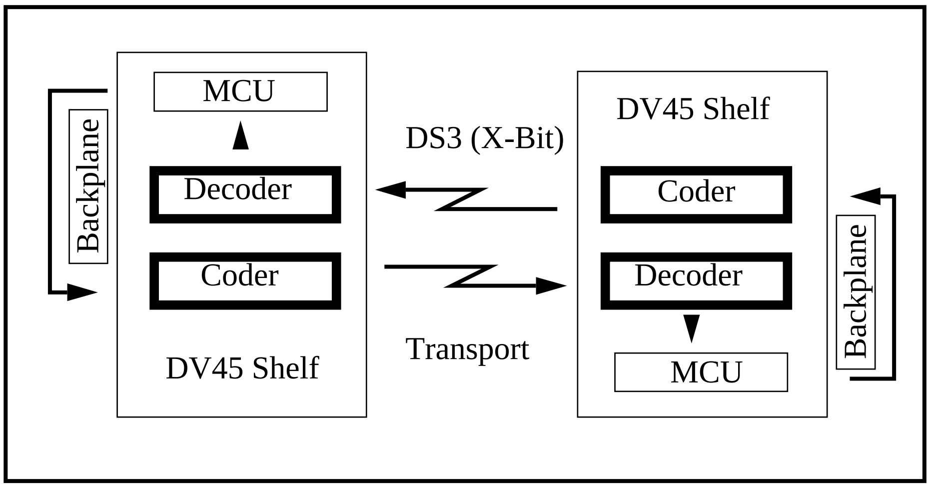


SECTION 9
X-Bit Information

9.1 General Overview

The DV45 uses the X-Bit channel of the DS3 stream to send alarm and status information from the local shelf to the remote shelf (Figure 9.1). This information is used to provide end-to-end status to the monitoring equipment. The hardware interface information for the X-Bit channel is public and well-known. This document will only describe the software level interface used by the DV45 equipment (coder, decoder and data interface card).

Figure 9.1 - X-Bit Channel Data Direction



Since the DS3 stream is unidirectional, going from the coder to the decoder, the MCU transfers to the coder the alarm and status information of the shelf decoder for transmission to the remote shelf. In this way end-to-end status and alarm information can be monitored from both ends.

The situation is the same if there is a data interface unit between adjacent coder and decoder units. But then the MCU forwards the decoder information to the data interface instead of the coder. Other implementation are expected to comply with only the test functions of Audio/ Video loopback. Data base content message, if provided, is specific to each manufacturer.

9.2 Packet Framing

Information on the X-Bit channel is carried on small variable length packets (Figure 9.2). Each packet has a maximum length of 256 bytes of payload. The payload is framed by specific start and end bytes and the integrity is verified by a cyclic redundancy check.

Figure 9.2 - X-Bit Packet Structure

Dle	Stx	Payload	...	...	Crc-1	Crc-2	Dle	Etx

Not to restrict the content of the payload a {Dle} byte occurring in the payload is sent as a pair {Dle,Dle}.The framing bytes are the usual ANSI-defined characters shown in Table 9.1.

Table 9.1 - Framing Bytes

Bytes	Hex value	Meaning
Dle	10	Data Link Escape
Stx	02	Start of text
Etx	03	End of text

The CRC is the 16-bit CCITT defined cyclic redundancy check for the polynomial (Equation (EQ 3)). It is fully described in [6].

$$CRC = x^{16} + x^{12} + x^{5} + 1 \quad \text{(EQ 3)}$$

9.3 Payload content

We will describe packet content in BNF-like (Backaus-Nohr-Form) notation:

- Bytes with specific values are denoted by their hexadecimal representation as 0xNN.
- Database points (e.g. *Audio_chip_failure*) take on only bit values {0,1}. “0” indicates negation of the database point meaning (e.g. *Audio_chip_failure* = 0, means that the audio chip has not failed).
- Sets of possible values are denoted by curly braces with ellipsis indicating obvious sequence range.
- Concatenation is denoted by the comma “,”.
- “Or” is denoted by the “/”.
- Repetition is denoted by start “*”, followed by the repetition count.

There are only 2 different types of packets sent over the X-Bit channel. The loopback request packet and the database packet.

Payload:= Loopback|Database

Loopback := 0xF0,Loopback_request
Database := Alphabetic,Alphabetic,Numeric,Numeric,Unit_Database

Alphabetic := {A,B, ... Z}
Numeric := {0,1, ... 9}
Unit_Database := Decoder|Coder|D ata interface

9.3.1 Decoder Database Content

Decoder := Audio_chip_failure,Audio_ram_failure,Audio_fifo_failure
, Power_failure,Battery_A_failure,Battery_B_failure
, No_video,Broadcast_circuit_failure
, Video_ram_failure,Video_fifo_failure,Video_dram_failure
, Digital_audio_failure,J81_chip_failure,Frame_store_failure
*, DSP_failure,Xilinx_failure,Unused_point * 16,Loopback_enable*
, Freeze_on_video_loss,B8ZS_AMI,LBO2,LBO1,LBO0

, *Digital_analog_audio,Audio_frame_loss,Mcu_dead,Audio_AIS*
, *External_boot,Monitoring_video,Frame_store_equipped*
, *No_external_broadcast,Video_sync_loss_pre_process*
, *Front_end_dead,Video_sync_loss_post_process*
, *Decoding_mode_1,Decoding_mode_0,DS3_clock_loss*
, *DS3_frame_loss,DS3_parity,DS3_parity_div8,DS3_blue*
, *DS3_all_ones,DS3_degraded,Unused_point * 101*
, *Loopback_control,,Digital_analog_control,,Broadcast_control,*
, *B8ZS_AMI_control,Impedance_2,Impedance_1,Impedance_0*
, *Frame_store_control,Disable_black,Unused_point * 62*
, *Unit_fail_alarm,J81_DS3_fail_alarm,J81_Video_fail_alarm*
, *DS3_fail_alarm,DS3_alarm_indication_signal*
, *Video_sync_loss_alarm,Unit_on_status,Black_active_status*
, *Lamptest_status,Audio_channel_active_status*
, *Video_pattern_generation_control,Pattern_1,Pattern_0*
, *Freeze_control,Loopback_control,Unused_point * 7*

9.3.2 Coder Database Content

Coder := *Auidio_failure,Audio_ram_failure,Audio_fifo_failure*
, *Power_failure,Battery_A_failure,Battery_B_failure,No_video*
, *Broadcast_failure,Video_ram_failure,Video_fifo_failure*
, *Unused_point,Dsp_failure,Xilinx_failure,Unused_point * 19*
, *Loopback_sense,Unused_point,B8ZS_AMI,Encoding_mode_1*
, *Encoding_mode_0,Loopback_request,Digital_analog_audio*
, *Audio_frame_loss,Mcu_dead,Audio_ais,External_boot*
, *DS1_loss,Unused_point,Broadcast_equipped,No_video*
, *Front_end_dead,45_meg_lock,Video_overload_check*
, *Unused_point * 110,Digital_analog_control,Calibration_control*
, *45_meg_lock_control,B8ZS_AMI_control,Mode_1,Mode_0*
, *Unused_point *4,No_video_mask,Front_end_dead_mask*
, *Unused_point * 59,Unit_fail,Video_failure,Video_sync_loss*
, *Unit_on,Unused_point *4,Lamptest,Unused_point * 4*
, *Loopback_request,Loopback_status,Unused_point * 8*

9.3.3 Data interface Database Content

Data interface:= *Power_failure,Battery_A_failure,Battery_B_failure*
, *Equipment_DS3_clock_loss,Equipment_DS3_frame_loss*
, *Line_DS3_clock_loss,Line_DS3_frame_loss,Oscillator_failure*
, *External_channel_failure,Unused_point * 14,Xilinx_failure*
, *Unused_point * 8,Equipment_AIS,Equipment_blue*
, *Equipment_parity,Equipment_par_div8*
, *Equipment_DS3_alarm_indication_signal*

, Equipment_DS3_degrade,Line_AIS,Line_blue,Line_parity
, Line_par_div8,Line_DS3_alarm_indication_signal
*, Line_DS3_degrade,Unused_point * 2,Dvln_address[0...7]*
*, Unused_point * 61,Dip_switch_7[0...7],Dip_switch_8[0...7]*
, Dip_switch_9[0...7],Slot_number[0...2],Dip_switch_14[0...7]
*, External_boot,Hybrid_codec_status,Unused_point * 41*
, Channel_1_232_422_control,Channel_1_clock_direction_control
, Channel_2_232_422_control,Channel_2_clock_direction_control
, Channel_1_rts_control,Relay_control[0...7],Broadcast_control
, Frame_slip_clear_control,Channel_2_rts_control
, Pll_clear_control,,Channel_1_appletalk_control
, Channel_2_appletalk_control,C-Bit_transparency_control
, Xilinx_reset_control,DS1_DS0_control,Channel_1_rx_rts_control
*, Channel_2_rx_rts_control,Unused_point * 14,Unit_fail_alarm*
, Equipment_frame_loss_alarm,Line_frame_loss_alarm
*, Communication_channel_failure,Unused_point * 4,Lamptest*
, Unused_point,Communication_channel_transmit_status
, Communication_channel_receive_status,Loopback_request
*, Unused_point * 2*

BIBLIOGRAPHY

[1] Bellcore, Technical Reference TR-TSY-000431, *15 kHz Digital Audio Terminal for Program or Television - Requirements and Objectives*, Issue 1, October, 1987.

[2] CCITT Rec. J41, *Characteristics of Equipments for Coding Analogue Sound Programme Signals*, 1984

[3] Digital Equipment EK-VT100-TM, *VT-100 Technical Manual*

[4] Electrical Industrial Association, *RS-170-A Electrical Performance Standards - Color Television Studio Facilities*, November 1977

[5] S. Sabri, D. Lemay and E. Dubois, *A Modular Digital Video Coding Architecture for Present and Advanced TV Systems*, SMPTE Journal, 504-511, July 1989

[6] International Standards, ISO 3309, *Information processing systems - Data communications- High level data link procedures - Frame structure*, 1984