Programmer Manual

Tektronix

MTM400
MPEG Transport Stream Monitor

071-1375-02

This document applies to firmware version 2.3.8
and above.

www.tektronix.com

Contacting Tektronix

Tektronix, Inc.
14200 SW Karl Braun Drive
P.O. Box 500
Beaverton, OR 97077
USA

For product information, sales, service, and technical support:

- In North America, call 1-800-833-9200.
- Worldwide, visit www.tektronix.com to find contacts in your area.

Warranty 2

Tektronix warrants that this product will be free from defects in materials and workmanship for a period of one (1) year from the date of shipment. If any such product proves defective during this warranty period, Tektronix, at its option, either will repair the defective product without charge for parts and labor, or will provide a replacement in exchange for the defective product. Parts, modules and replacement products used by Tektronix for warranty work may be new or reconditioned to like new performance. All replaced parts, modules and products become the property of Tektronix.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period and make suitable arrangements for the performance of service. Customer shall be responsible for packaging and shipping the defective product to the service center designated by Tektronix, with shipping charges prepaid. Tektronix shall pay for the return of the product to Customer if the shipment is to a location within the country in which the Tektronix service center is located. Customer shall be responsible for paying all shipping charges, duties, taxes, and any other charges for products returned to any other locations.

This warranty shall not apply to any defect, failure or damage caused by improper use or improper or inadequate maintenance and care. Tektronix shall not be obligated to furnish service under this warranty a) to repair damage resulting from attempts by personnel other than Tektronix representatives to install, repair or service the product; b) to repair damage resulting from improper use or connection to incompatible equipment; c) to repair any damage or malfunction caused by the use of non-Tektronix supplies; or d) to service a product that has been modified or integrated with other products when the effect of such modification or integration increases the time or difficulty of servicing the product.

THIS WARRANTY IS GIVEN BY TEKTRONIX WITH RESPECT TO THE PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX' RESPONSIBILITY TO REPAIR OR REPLACE DEFECTIVE PRODUCTS IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Warranty 9(b)

Tektronix warrants that the media on which this software product is furnished and the encoding of the programs on the media will be free from defects in materials and workmanship for a period of three (3) months from the date of shipment. If any such medium or encoding proves defective during the warranty period, Tektronix will provide a replacement in exchange for the defective medium. Except as to the media on which this software product is furnished, this software product is provided “as is” without warranty of any kind, either express or implied. Tektronix does not warrant that the functions contained in this software product will meet Customer’s requirements or that the operation of the programs will be uninterrupted or error-free.

In order to obtain service under this warranty, Customer must notify Tektronix of the defect before the expiration of the warranty period. If Tektronix is unable to provide a replacement that is free from defects in materials and workmanship within a reasonable time thereafter, Customer may terminate the license for this software product and return this software product and any associated materials for credit or refund.

THIS WARRANTY IS GIVEN BY TEKTRONIX WITH RESPECT TO THE PRODUCT IN LIEU OF ANY OTHER WARRANTIES, EXPRESS OR IMPLIED. TEKTRONIX AND ITS VENDORS DISCLAIM ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. TEKTRONIX’ RESPONSIBILITY TO REPLACE DEFECTIVE MEDIA OR REFUND CUSTOMER’S PAYMENT IS THE SOLE AND EXCLUSIVE REMEDY PROVIDED TO THE CUSTOMER FOR BREACH OF THIS WARRANTY. TEKTRONIX AND ITS VENDORS WILL NOT BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES IRRESPECTIVE OF WHETHER TEKTRONIX OR THE VENDOR HAS ADVANCE NOTICE OF THE POSSIBILITY OF SUCH DAMAGES.

Table of Contents

List of Figures

Preface

This document specifies the MTM400 MPEG Transport Stream Monitor remote control and status monitoring interfaces available to a Management application. Two interfaces are provided; SNMP and an HTTP Web-based interface.

The manual is organized into the following sections:

- Introduction
- MTM400 MIB
- MIB Group Overview
- System Structure
- MPEG Structure
- Web Server URLs

The following documents are also available on the Tektronix Web site (www.tektronix.com):

- MTM400 MPEG Transport Stream Monitor User Manual (Tektronix part number: 071-1224-xx)

  This manual provides operational information for the MTM400.

- MTM400 MPEG Transport Stream Monitor Technical Reference (Tektronix part number: 071-1560-xx)

  This manual provides product specifications, test parameters, configuration file syntax, and hardware maintenance procedures.

Related Material

The following URLs access the Web sites for the standards organizations listed (the URLs listed were valid at the time of writing):

- MPEG−2 standards (International Organization for Standards) http://www.iso.ch/
- DVB standards (European Technical Standards Institute) http://www.etsi.org/
- ATSC standards (Advanced Television Systems Committee) http://www.atsc.org/

- ISDB/ARIB standards (Association of Radio Industries and Businesses)
  http://www.arib.or.jp/english/
- SCTE Society of Cable Television Engineers
  http://www.scte.org/

Introduction

Introduction

This document specifies the MTM400 MPEG Transport Stream Monitor remote control and status monitoring interfaces available to a Management application. Two interfaces are provided; SNMP and an HTTP Web-based interface.

***NOTE**. The MTM400 Programmer Interface MIB file accompanying this document contains entries not described in the manual. These entries should not be used.*

This document should be read in conjunction with the MTM400 User Manual. The reader must be thoroughly familiar with the operation of the MTM400 and have detailed knowledge of SNMP and HTTP.

Do not use multiple variable binding SET requests. Only single variable binding SET requests should be used.

SNMP and MIBs

This document specifies the facilities provided by the MTM400 Simple Network Management Protocol (SNMP) agent, which allows various parameters within the MTM400 monitor to be viewed and set. This will allow you to develop management applications that can control the MTM400 units across a network using SNMP.

The MTM400 SNMP agent has been implemented as an extensible agent under Nucleus, and as such conforms to SNMP v1. Nucleus SNMP is an embedded implementation of SNMP.

The Simple Network Management Protocol (SNMP) is an Internet standard protocol for remote management of entities on a network. It is defined in Internet documents STD-15 (RFC1157) and STD-16 (RFC1155 and RFC1212). STD-15 defines the protocol operations; STD-16 defines the way in which information is structured under SNMP (SMI - Structure of Management Information).

SNMP defines a way of structuring information in a hierarchy of objects supporting both single objects and tables of objects, and making the information available through a network protocol.

Each object can be one of four types, namely:

- **Integer**. Represents numerical values.
- **OctetString**. Represents byte streams.

- **DisplayString**. Represents printable strings.
- **Object Identifier** (OID). References other objects within SNMP.

There are essentially three types of operations that can be performed on each object:

- **Get**. Retrieves the value of an object.
- **GetNext**. Retrieves the value of an object along with the OID of the next object available.
- **Set**. Sets the value of an object.

The complete set of objects accessible through an SNMP agent is called the Management Information Base (MIB). The MIB is a tree structure with MIB objects at the leaves of the tree. Every branch and leaf of the tree is numbered according to a scheme ultimately under the administration of either ISO or the CCITT (or the ITU-T as they are now called). (The root of the tree has three branches: branch 0 is owned by the CCITT, 1 by ISO and 2 is jointly owned by ISO and the CCITT.) These organizations have delegated various branches of this tree to other authorities. Everything of interest to SNMP is under the control of the IANA (the Internet Assigned Numbers Authority), which owns the branch named:

iso (1).org (3).dod (6).internet (1)

The strings of numbers identifying parts of the MIB tree are called Object Identifiers (OIDs).

The Internet standard management sub-trees are all under

iso (1).org (3).dod (6).internet (1).mgmt (2).

However the IANA also allocates numbers to other organizations. Companies can obtain their own sub-trees under

iso (1).org (3).dod (6).internet (1).private (4).enterprises (1).

This entire tree structure is called the MIB. A MIB module is a set of sub-sections of this tree that form some coherent function or set of functions, usually described in a single document and qualified with some other title, such as RMON MIB.

***NOTE**. A MIB module is sometimes referred to as the MIB.*

A MIB Module is defined in a text file using ASN.1 (Abstract Syntax Notation One). This is a language defined by OSI for describing arbitrary data structures.

For more detailed explanations of network management using SNMP, you can refer to *The Simple Book* (Marshall T. Rose, Prentice Hall, ISBN 0-13-451659-1).

MTM400 SNMP Community

SNMP provides a simple mechanism for security, there are community strings to govern read and write to the MIB; these function as passwords.

For the MTM400 the community string “public” is used for read and write access.

MTM400 SNMP Traps

SNMP provides a mechanism for a device to send a notification message to the management system when an event occurs. This means that the management system can poll the device less often and so reduce network traffic.

The important point to note here is that it does not mean that the management system can stop polling the device. Traps are sent using the UDP network protocol. This mechanism does not guarantee arrival of all packets; a trap message can be lost.

Trap messages may be lost not only in the UDP transport layer, but inside the device. The MTM400 takes steps to avoid flooding the network with traps; this means some traps are discarded when there are a burst of error in a stream. A trap should be thought of as a prompt to visit the device to discover status rather than a mechanism to completely know the status.

You should not use trap messages alone to continually monitor the status of the device. This will fail because although traps are issued when the device detects an error and when the error condition clears, they are not issued when tests move from an error state to not applicable, such as when the stream presentation is changed.

To prevent a flood of trap messages on a network, the MTM400 has a throttling mechanism. A flood of trap messages is to be avoided since this could hamper the operator’s ability to use the network to understand and contain an error condition. In the extreme case a flood of trap messages could cause the management system to fail.

On the MTM400, a maximum number of trap messages per second is defined. This is in total, so, if a limit of 10 per second is set, this will yield 5 per second if two managements are subscribed. Internally, there is a buffer for 100 traps so a

short burst can be accommodated without loosing messages. If the buffer overflows, trap messages are discarded.

The implication of the preceding information is that network bandwidth, or trap handling capability, is treated as a limited resource. To avoid wasting this resource, steps are taken to ensure that any management system subscribed for trap messages still requires these messages. So when a management system subscribes to trap messages, this is only for a few minutes. The management system must repeatedly resubscribe in order to continue to receive trap messages. This provides protection in the case of a management system exiting improperly.

MTM400 Web Server

The MTM400 has a Web server interface on HTTP port 80. A number of URL's are supported and are used primarily for transferring bulk data, unsuited to SNMP, to and from the MTM400.

A full list of supported Web server URLs is given in this manual (see *Web Server URLs* section).

MTM400 MIB

MTM400 MIB

Tektronix has been assigned the following root OID:

iso.org.dod.internet.private.enterprises.128

Under this OID Tektronix can define its own MIB for various products.

The MIB subtree for MTM400 is under the following OID:

iso.org.dod.internet.private.enterprises.tek(128).tvt(5).tvtproducts(1).

The tree is specified in the two ASN.1 text files: ADSYS.MIB defines the structure of device specific elements and ADMPEG.MIB defines the structure of the MPEG Interface specific elements.

The supplied MIB includes some items that do not apply to the MTM400, because the MIB is common to several products.

MIB Types

The MTM400 MIB defines the following extra MIB types.

EVID This type defines events that can occur within the MTM400. It is essentially a WORD, where values 0x1xxx represent events that are generated by the MTM400 box such as Clock and Battery errors. Values over and including 0x2000 represent events that are generated by specific MPEG Interfaces such as Sync Lock or Continuity errors. The full list of these events can be found in the MTM400 Technical Reference (Tektronix part number: 071-1560-xx).

EvState This type represents the state of a given event which can be Green, Yellow or Red. Green indicates that there is no error, yellow indicates that there has been an error since this event was last reset, and red indicates that there is a persistent error.

This is essentially a WORD. Green is defined as 0x1000, yellow as 0x2000 and red as 0x3xxx, where xxx is the specific error number. A value of 0x0000 means that the state is unknown (for example, during the settling time of a test), and 0x4000 means that the event is disabled. Two final values are also possible: 0x5000 is the maintenance state and 0x6000 is N/A (for example, SFN testing when there is no SFN data).

AlmValue This specifies which alarms are activated when an event occurs. It is an integer type and can take combinations of the following values:

0x00000001 = Audible Alarm
0x00000100, 0x00000200, .. , 0x00001000= Relay1, Relay2, .., Relay 5
0x00010000, 0x00020000, .. , 0x00040000= TTL1, TTL2, TTL3
0x00100000, 0x00200000 = Send Trap on; Raise, Clear

Simple Boolean This enumerated type is used to represent a Boolean value.

Log Index This type represents an integer index into a log.

Time Stamp Time stamps are used in several MIB items to specify the time of events. Each time stamp is stored as an eight-byte structure, which consists of an 11-bit signed integer representing the UTC offset and a 53 bit signed integer representing the UTC time. The UTC offset is the number of minutes that must be added to UTC time to obtain the local time on the MTM400 unit. The UTC time is the number of microseconds since midnight Greenwich Mean Time (GMT) January 1, 1970.

Figure 2–1 shows that the timestamp is actually stored with the UTC offset, followed by the UTC Time in MSB format. However, the bytes are reversed when the timestamp is presented as part of an Octet String through SNMP so that the numbers are in LSB format. Care should be taken with byte 6 because it contains both the UTC offset and UTC time.

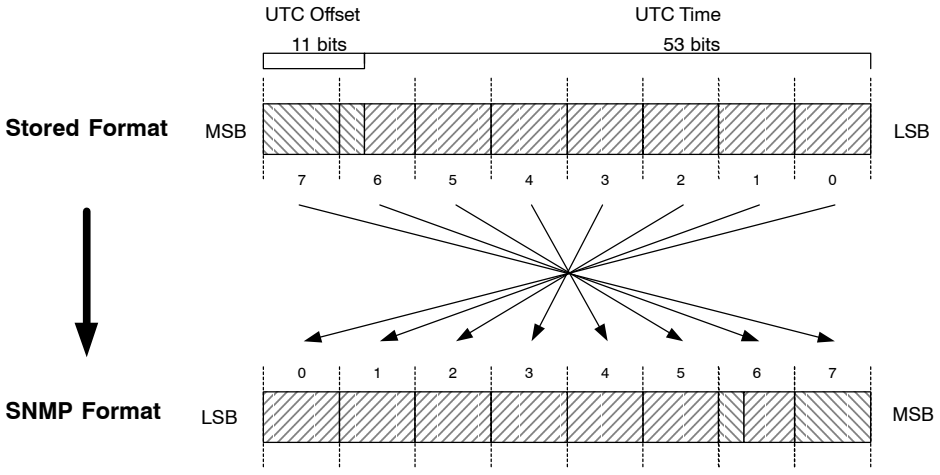


Figure 2–1: Time stamp storage

Accessing MIB Objects

This section describes how to access objects within the MTM400 MIB.

SNMP Access Operations

The MTM400 SNMP agent fully supports the standard SNMP GetRequest, GetNextRequest, and SetRequest PDU operations. This document specifies the access permissions for each object within the MTM400 MIB using the following conventions:

- 'Get' indicates that the GetRequest and GetNextRequest can be used.
- 'Set' indicates that the SetRequest can be used.

Single Leaf Objects

Single Leaf Objects are single-value elements whose values can be accessed using the standard SNMP access operations by appending '0' to the appropriate OID specified in the MIB. For example, in order to access the program name within the System Information Group, use the following OID:

'…adsysProductName.0'.

Tables

The MTM400 MIB defines a number of tables. Tables normally contain objects that can have multiple values, each referenced by appending the required row number to the OID of the object specified in the MIB. Management applications typically access values of objects within tables by first performing a GetNextRequest-PDU on the OID object that will return the OID of the first value. Subsequent calls to the GetNextRequest operation will obtain the values for this object within the table. When the operation returns the 'No Such Name' error, this indicates that the last value has been reached.

Some tables within the MTM400 MIB are indexed by two or more values, so accessing object values becomes a little more complex. For example, the Event State Table is indexed by stream number and event id, so in order to reference a specific value, the OID should be created by appending the stream number and the event id to the OID specified for this object in the MIB. Consequently, in order to access the EventState for an event on a specific stream, use the following OID:

'…mivevtEventState.<interface_no>.<eventid>'.

The GetNextRequest-PDU operation will return the OID of the next eventid, until they have all been exhausted for that stream. At this point it will return the next interface_no, and the first event_id on that interface (or 'No Such Error' if no more interfaces exist to indicate that the end of the table has been reached).

When a table is defined within the MIB, each table leaf object is represented by the following OID:

'…<table_oid>.<table_entry_oid>.<table_leaf_object_oid>'.

The 'table_entry_oid's within the MTM400 MIB are always given the value 1, and are not shown on the structure charts within this document because it would complicate the diagrams. However, it should be recognized that these must be included in the OIDs when referencing objects.

MIB Group Overview

MIB Group Overview

The following sections define the groups of the MIB modules that make up the MTM400 SNMP interface. There is a split between MPEG-related and non-MPEG-related objects, and so the groups have been separated into two MIB modules. The System MIB module contains all non-MPEG-specific groups; MPEG-specific groups are found in the MPEG MIB module. Figures 3–1 to 3–3 show the overall structure of the MTM400 MIB subtree.

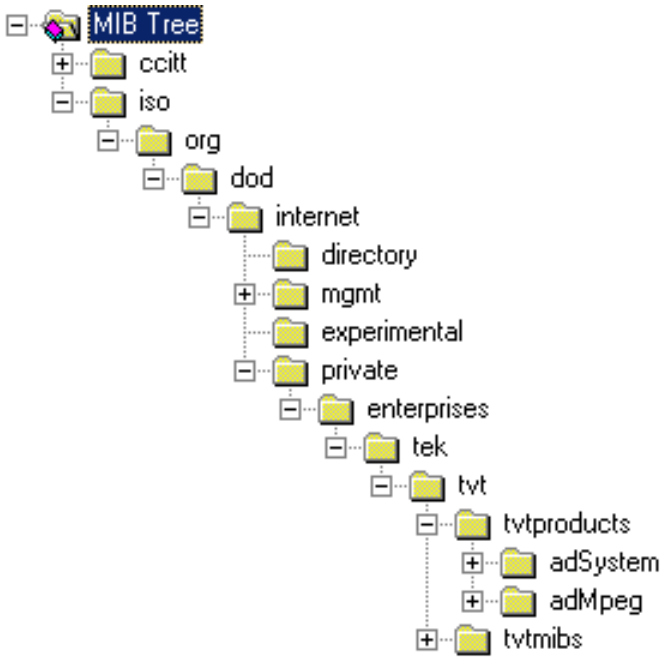


Figure 3–1: Overall MIB structure

The system OID is:

iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).tek(128).tvt(5).tvtproducts(1).adSystem(16)

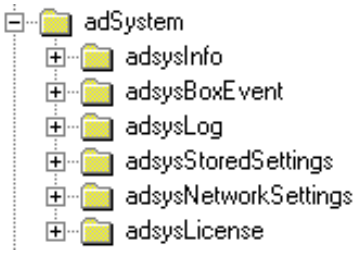


Figure 3–2: System structure

For a complete description of the system structure, refer to the *System Structure* section of this manual.

The MPEG OID is:

iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).tek(128).tvt(5).tvtproducts(1).adMpeg(17)

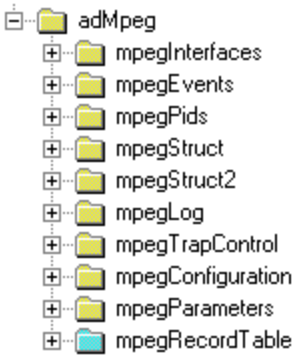


Figure 3–3: MPEG structure

For a complete description of the MPEG structure, refer to the *MPEG Structure* section of this manual.

System Structure

System Structure

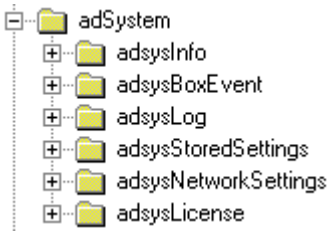


Figure 4–1: System structure

System Information Group

Figure 4–2 shows the structure of the System Information Group, which provides access to attributes of the most general nature, such as the product name and the installed software .

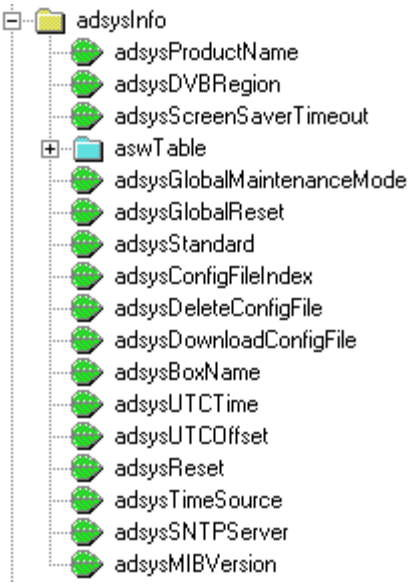


Figure 4–2: System information group structure

Product Name

For the MTM400 this is fixed as “MTM400”. This may be used to positively identify an MTM400.

The format of this item is defined as:

Name:	adsysProductName
OID:	1.3.6.1.4.1.128.5.1.16.1.1
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).tek(128).tvt(5).tvtproducts(1).adSystem(16).adsysInfo(1).adsysProductName(1)
Module:	AD-SYSTEM-MIB
Parent:	adsysInfo
Numerical syntax:	Octets
Base syntax:	OCTET STRING
Composed syntax:	OCTET STRING
Status:	mandatory
Max access:	read-only
Description:	A textual name unique to this product type

DVB Region

Obsolete - see *MPEG Structure, MPEG Interfaces Table*.

Screen Saver Timeout

N/A

Software Components

A list of software components and performance metrics is present on this entity.

The format of this item is defined as:

Variable	Type	Use	Access
aswIndex(1)	Integer	Table index.	Get
aswName(2)	Octet string	Component name.	Get
aswVersion(3)	Octet string	Component version.	Get

Global Maintenance Mode

The format of this item is defined as:

Name:	adsysGlobalMaintenanceMode
OID:	1.3.6.1.4.1.128.5.1.16.1.5
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).tek(128).tvt(5).tvtproducts(1).adSystem(16).adsysInfo(1).adsysGlobalMaintenanceMode(5)
Module:	AD-SYSTEM-MIB
Parent:	adsysInfo
Numerical syntax:	Integer (32 bit)
Base syntax:	INTEGER
Composed syntax:	SimpleBoolean
Status:	mandatory
Max access:	read-write
Description:	Setting this variable to true sets the whole box into global maintenance mode. In this state, processing of events continues, but no alarms are raised.

Standard

Obsolete

Config File Index

N/A

Delete Config File

N/A

Download Config File

N/A

Box Name

This value contains a configurable name for the box.

Name:	adsysBoxName
OID:	1.3.6.1.4.1.128.5.1.16.1.11
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adSystem(16).adsysInfo(1).adsysBoxName(11)
Module:	AD-SYSTEM-MIB
Parent:	adsysInfo
Numerical syntax:	Octets
Base syntax:	OCTET STRING
Composed syntax:	OCTET STRING
Status:	mandatory
Max access:	read-write
Description:	The name of the box

UTC Time

The UTC time of the box; that is, the number of seconds since midnight 1st January 1970.

Variable	Type	Use	Access
adsysUTCTime (12)	Integer	The UTC time of the box.	Get/Set

UTC Offset

Number of minutes to add to UTC time to get to local time frame - this may be negative.

Variable	Type	Use	Access
adsysUTCOffset (13)	Integer	The UTC offset of the box.	Get/Set

Reset

Setting this value to a hex value DE5B12A resets the device

Variable	Type	Use	Access
adsysReset (14)	Integer	Device reset. Get has no meaning in this context.	Get/Set

Time Source

Specifies the system time source.

Variable	Type	Use	Access
adsysTimeSource (15)	Integer	0 = RTC (Real Time Clock on the device). 1 = LTC (Longitudinal Time Code). 2 = SNTP (Simple Network Time Protocol).	Get/Set

SNTP Service

The IP Address of an SNTP server.

Variable	Type	Use	Access
adsysSNTPServer (16)	IP Address	SNTP server IP address.	Get/Set

MIB Version

SNMP interface version.

Variable	Type	Use	Access
adsysMIBVersion(17)	Octet string	SNMP interface version.	Get

Box Event Group

The MTM400 may generate several box-specific events. Normally, an event may be in one of five states:

- 'Red' (0x3xxx) indicates that there is currently an error condition.
- 'Yellow' (0x2000) indicates that there is currently no error condition, but that one has occurred since this event was last reset.
- 'Green' (0x1000) indicates that there is no error condition.
- 'Gray' (0x0000) indicates the state is unknown (or that the link is lost).
- 'White' (0x4000) indicates that the event is disabled.

Each event also has an alarm value associated with it, which indicates the type of alarm that will be triggered (such as audible or relay), if the event goes into error. The full list of box events is specified in the MTM400 Technical Reference (Tektronix part number: 071-1560-xx).

The following diagram shows the structure of the Box Event Group, which contains information on the states and alarm values for all box events that can be generated by the MTM400.

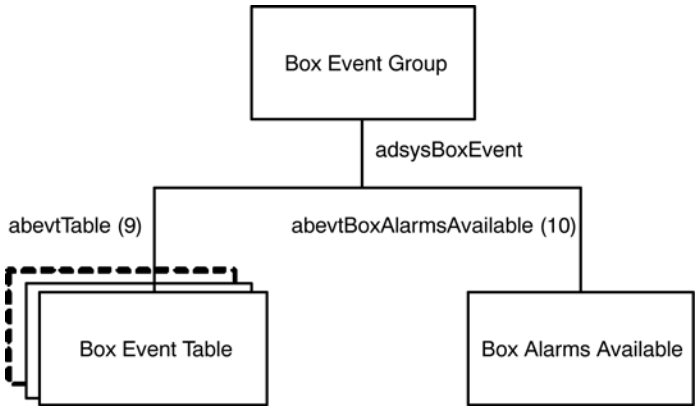


Figure 4–3: Box event group structure

The following table describes the objects within the Box Events Group.

Box Events

The Box Alarm table contains the state and alarm value for each box-wide event as specified in the MTM400 Technical Reference (Tektronix part number: 071-1560-xx).

Variable	Type	Use	Access
abevtIndex(1)	EvId	An index identifying the event id as defined in Appendix A.	N/A
abevtEventName(2)	Octet string	A short name for this event.	Get
abevtEventDescription(3)	Octet string	A brief description of the meaning of this alarm.	Get
abevtEventState(4)	EvState	The state of this event.	Get/Set
abevtAlarmValue(5)	AlmValue	The alarms that will be triggered for this event.	Get/Set
abevtEventEnable(6)	Simple Boolean	Specifies whether the event is enabled (0 = disabled, 1 = enabled).	Get/Set

Indexing. The table is indexed by EvId; for example in order to reference the name of event 0x1000 (4096), use the following OID:

'…abevtEventName.4096'.

Name and Description. An event name and description are included in this table so that management applications using this MIB can report all events. (This table has been designed so that new event types can be added later. A management application could display all of the event types it knows about in a predetermined manner, but still be able to display events added after it was written.) These textual MIB variables would typically be downloaded once when the management application starts, or not at all if you only want to display some particular fixed set of events.

Unsupported Events. Box events that are not supported for the MTM400 unit will have an event state of 0x0000.

Event States. Reading the event state returns the current event status as described for the EvState type (see *MTM400 MIB* section). Writing any value will reset the event. The effect of resetting is to change a 'yellow' event state to either 'green' or 'unknown'.

Alarm Values. An alarm value specifies which alarms will be triggered when the corresponding event indicates an error. A value is a combination of those specified AlmValue (see page *Box Events*), for example, 0x00020401 will set TTL2, Relay3, and Audible alarms to be triggered.

Available Box Alarms

Variable	Type	Use	Access
abevtBoxAlarmsAvailable(10)	AlmValue	The result of 'ORing' the types of alarms that can be triggered for box events. This is determined by the hardware available on the addressed box.	Get

Box Log Group

Figure 4–4 shows the structure of the Box Log Group, which provides access to the box specific log items.

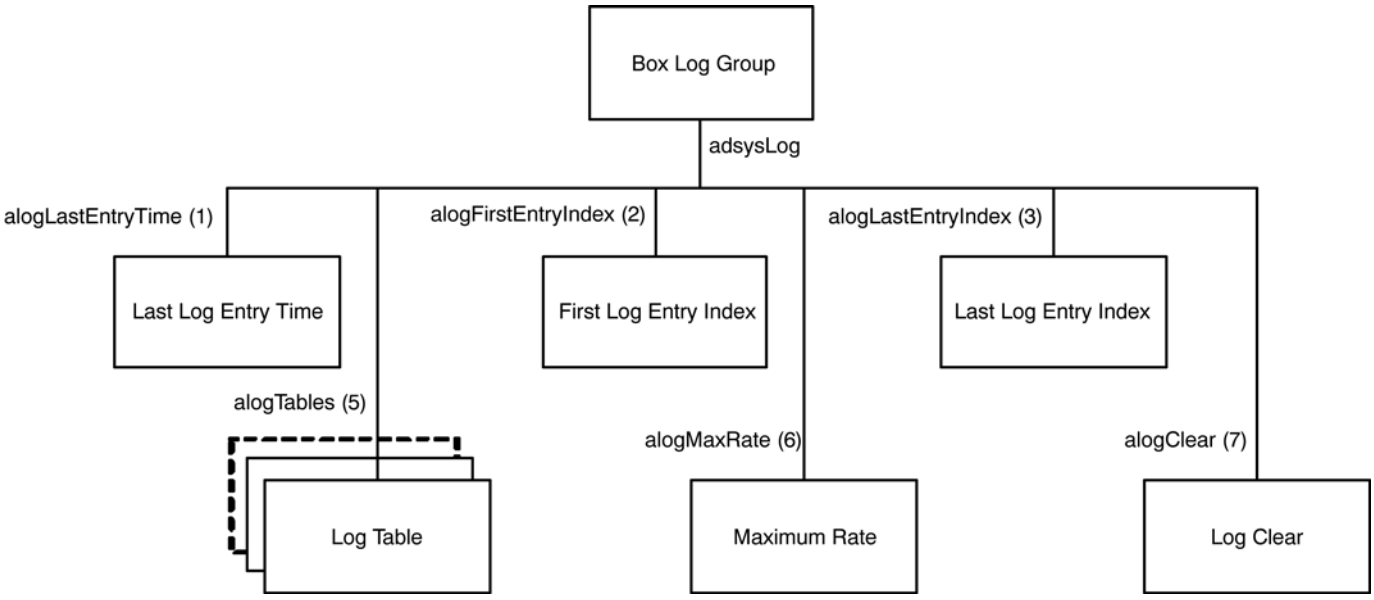


Figure 4–4: Box log group structure

The following table describes the single leaf objects within the Box Log Group. It should be noted that some of these values also apply to the stream logs.

Variable	Type	Use	Access
alogLastEntryTime(1)	Time Ticks	The value of sysUpTime at which the most recent entry was added to a box log or any stream log.	Get
alogFirstEntryIndex(2)	Log Index	The index of the oldest box log entry.	Get
alogLastEntryIndex(3)	Log Index	The index of the most recent box log entry.	Get
alogMaxRate(6)	Integer	This sets the maximum number of entries that will be logged (per second) for both box and stream logs. A value of 0 disables logging, and a value of 10000 specifies that there is no maximum limit.	Get/Set
alogClear(7)	Integer	Setting this value clears the box and stream logs.	Get/Set

The first and last entry indices can be used to access the required elements from the Log Table, which is shown in the following table.

Log Table

The log entry table contains information on the event log generated by the MTM400, and is defined as:

Variable	Type	Use	Access
alogIndex(1)	Log Index	Log entry index.	N/A
alogText(2)	Octet string	Contains a coded representation of the log entry.	Get

Indexing. The table index is an integer, so it may wrap around if the number of entries in the log becomes very large. This implies that the element with the largest index is not necessarily the latest log entry. The index of the last entry can be obtained from the single leaf element alogLastEntryIndex. In order to obtain the required log text from the table, use the following OID:

'…alogText.<index>'.

Log Text Formatting. The alogText will be empty if the index requested is not valid. This occurs if the management application requests an entry that no longer exists, for example, if the log was full and the entry was deleted from the end of the list to make room for new entries. If the log is being filled rapidly, the index returned from alogFirstEntryIndex is likely to be invalid for a call to alogText.

If alogText is not empty, the format of the octet string is as follows:

Bytes 0..7 : Public timestamp structure.

Bytes 8..9: The ID of the event.

Bytes 10..11 : Extension ID.

Bytes 12..13 : The state of the event.

Bytes 14 - onwards : Text description (UTF8, not NULL terminated).

***NOTE**. All numeric values are coded with the LSB first.*

Network Settings

The network settings table provides information on the device's network settings. The information available is defined as:

Variable	Type	Use	Access
aNetIpAddress(1)	IP address	The IP address of the device.	Get/Set
aNetGatewayAddress(2)	IP address	The IP address of the gateway for the device.	Get/Set
aNetSubnetMask(3)	IP address	The subnet mask.	Get/Set
aNetCommunityRead(4)	Display string	Alternate SNMP community string used to read.	Get/Set
aNetCommunityWrite(5)	Display string	Alternate SNMP community string used to write.	Get/Set
aNetCommunityTrap(6)	Display string	SNMP target community for all traps.	Get/Set

The read and write community strings in this table are alternates to support management systems with fixed communities. The default 'public' community will always work.

Changing the network information will have no effect until the MTM400 is reset.

License Table

This field is an octet string containing a variable length bit field enumerating the licensed capabilities of the unit.

Variable	Type	Use	Access
alicCapabilities(1)	Octet string	The licensed capabilities of the device.	Get

The current bit definitions are:

0	Structure View	16	Reduced I/O card
1	Repetition Graphs	17	QAM A Select *
2	Bitrate Limits	18	QAM B Select *
3	Pid Groups	19	QAM C Select *
4	Templates	20	Reserved
5	Template Tree View	21	Reserved
6	Recording	22	Reserved
7	PCR Graphs	23	Reserved
8	SFN	24	Reserved
9	Service Log	25	Reserved
10	Pid Variability	26	Reserved
11	Scheduling	27	DPI
12	MPE	28	Reserved
13	TMCC	29	RF Tests
14	RF Analysis	30	Reserved
15	Full I/O card		

* - applicable to common design cards only

Each octet has bits numbered from zero for the least significant, to seven for the most significant. The first octet contains the values 0..7, the second contains 8..15, and so on up to the number of required octets.

MPEG Structure

MPEG Structure

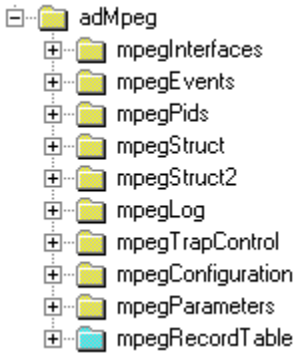


Figure 5–1: MPEG structure

MPEG Interfaces Group

Figure 5–2 shows the structure of the MPEG Interfaces Group, which contains information on each of the MPEG Interfaces connected to the MTM400 unit. The terms ‘Stream’ and ‘Interface’ are used interchangeably.

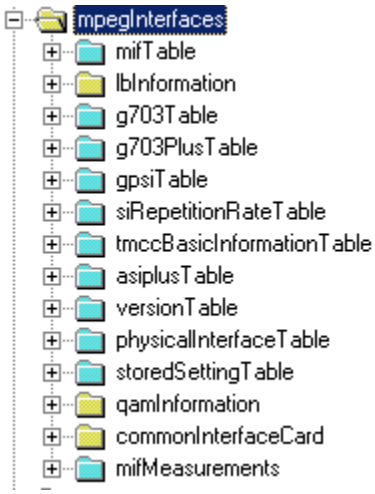


Figure 5–2: MPEG interfaces group structure

MPEG Interfaces Table

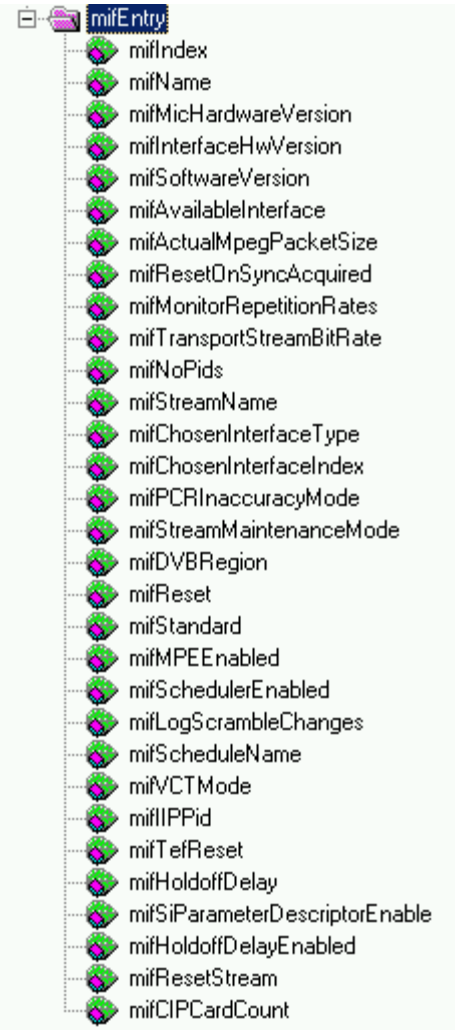


Figure 5–3: MPEG interfaces table structure

The MPEG Interfaces table is similar in concept to the Interfaces Group (ifTable) defined in MIB-II (RFC1213), which provides a list of all network interfaces that are installed in a device supporting network management. As with the ifTable, it allows a common network management mechanism to be used to describe and control MPEG interfaces regardless of the application. Also as with the ifTable, the indices into the MPEG Interface table can be used as cross references from other MIB modules, or even as indices for other tables, enabling these to extend the MPEG Interface table with application-specific information.

The table is defined as:

Variable	Type	Use	Access
mifIndex(1)	Integer	The MPEG Interface for which these readings apply. These are used to identify MPEG interfaces elsewhere in the MIB.	N/A
mifName(2)		N/A	
mifMicHardwareVersion(3)		N/A	
mifInterfaceHwVersion(4)		N/A	
mifSoftwareVersion (5)		N/A	
mifAvailableInterface(6)	Integer	The available interface (see *Available Interface* following this table).	Get
mifActualMpegPacketSize(7)	Integer	The actual MPEG Packet size received on this interface. This will be 0, 188, 204 or 208 where 0 indicates unknown.	Get
mifResetOnSyncAcquired(8)		N/A	
mifMonitorRepetitionRates (9)		N/A	
mifTransportStreamBitRate(10)	Integer	Transport rate of the stream in bps.	Get
mifNoPids(11)	Integer	Number of PIDs in the stream with non-zero bit rate and those that have had limits set.	Get
mifStreamName(12)	Octet string	Configurable name for the stream.	Get/Set
mifChosenInterfaceType(13)	Integer	Interface type to use for this stream. If more than one interface of this type is available, the first one will be chosen. To select an interface other than the first one, use 'mifChosenInterfaceIndex' instead. See 'physicalInterfaceType' in the physical interfaces table for more information. Any change to this variable will also affect 'mifChosenInterfaceIndex'.	Get/Set

Variable	Type	Use	Access
mifChosenInterfaceIndex(14)	Integer	Interface index type to use for this stream. The number selected matches the 'physicalInterfaceIndex' in the physical interfaces table. This variable must be used in preference to 'mifChosenInterfaceType' to select an interface other than the first one of a given type. A change to this variable may also affect 'mifChosenInterfaceType'.	Get/Set
mifPCRInaccuracyMode(15)		N/A	
mifStreamMaintenanceMode(16)	Integer	Specifies whether the stream is in maintenance mode. 0 = off, 1 = on.	Get/Set
mifDVBRegion(17)	Integer	Specifies the DVB Region of the stream. 0 = DVB 1 = DTG 2 = Nordic 3 = ISDB 4 = Aus 5 = Reserved 6 = DCII Hybrid	Get/Set
mifReset(18)	Integer	Resets the stream parameters to the factory defaults.	Get/Set
mifStandard(19)	Integer	Specifies the MPEG Standard for the stream. 0 = MPEG 1 = DVB 2 = ATSC 3 = ISDB 4 = China 5 = DigiCipher® II	Get/Set
mifMPEEnabled(20)		N/A	
mifSchedulerEnabled(21)	Integer	Specifies whether scheduler is enabled. 0 = disabled 1 = enabled	Get/Set
mifLogScrambleChanges(22)		N/A	
mifScheduleName (23)	Octet String	The name of the schedule file currently loaded.	Get
mifIIPPid(25)	Integer	The pid on which the ISDB-T Information Packets (IIP) are transmitted.	Get/Set

Variable	Type	Use	Access
mifTefReset(26)	Integer	Setting this to any value resets the TEF Count.	Get/Set
mifHoldoffDelay (27)	Integer	In ISDB mode there is an option to prevent alarms in the period following a PAT/PMT change; the length of the period is set here.	Get/Set
mifSiParameterDescriptorEnable (28)	Integer	In ISDB mode some test parameters can be taken from the stream; setting this parameter enables this mode.	Get/Set
mifHoldoffDelayEnabled (29)	Integer	In ISDB mode there is an option to prevent alarms in the period following a PAT/PMT change; setting this parameter enables this option.	Get/Set
mifResetStream (30)	Integer	Setting this to any value restarts the stream and clears all SI and tests.	Get/Set
mifCIPCardCount (31)	Integer	Returns a count of detected CIP cards.	Get

Indexing. The table is indexed by interface number, for example to reference the name for interface 1, use the following OID:

‘…mifName.1’.

Available Interface. This field indicates which, if any, of the supported interface cards are connected to the MTM400 via the serializer port. The interpretation of the mifAvailableInterface values is as follows:

0x0000 = Unknown
0x0800 = QAM_ANNEX_A
0x2800 = QAM_ANNEX_B
0x1800 = QAM_ANNEX_C
0x4800 = QAM_ANNEX_X
0x3800 = LBAND
0x4000 = Common Interface card, COFDM, 8PSK, 8VSB, QAMB2, GbE
0x6800 = SMPTE
0xE000 = ASI

Standard and Region

There are a number of standards; the region field meaning depends on the standard chosen. For DVB, this field denotes a region; in other cases it is a specialization.

Standard	Region
MPEG (0)	Standard (0)
DVB (1)	Std (0)
	DTG (1)
	Nordic (2)
	Reserved (3) (was ARIB - see ISDB)
	Aus (4)
	Reserved (5)
	DigiCipher® II Hybrid (6)
ATSC (2)	Standard (0)
ISDB (3)	ISDB-S (0) (Japanese standard)
	ISDB-T (1) (Japanese standard)
Chinese (4)	GY/Z 174-2001 (0) (DVB with explicit GB2312 content)
	GB2312 (1) (DVB with implicit GB2312 content)
DigiCipher® II (5)	SCTE57 (0)

LBand Information Group

The following diagram shows the structure of the LBand Information Group, which contains information on the LBand Settings where appropriate.

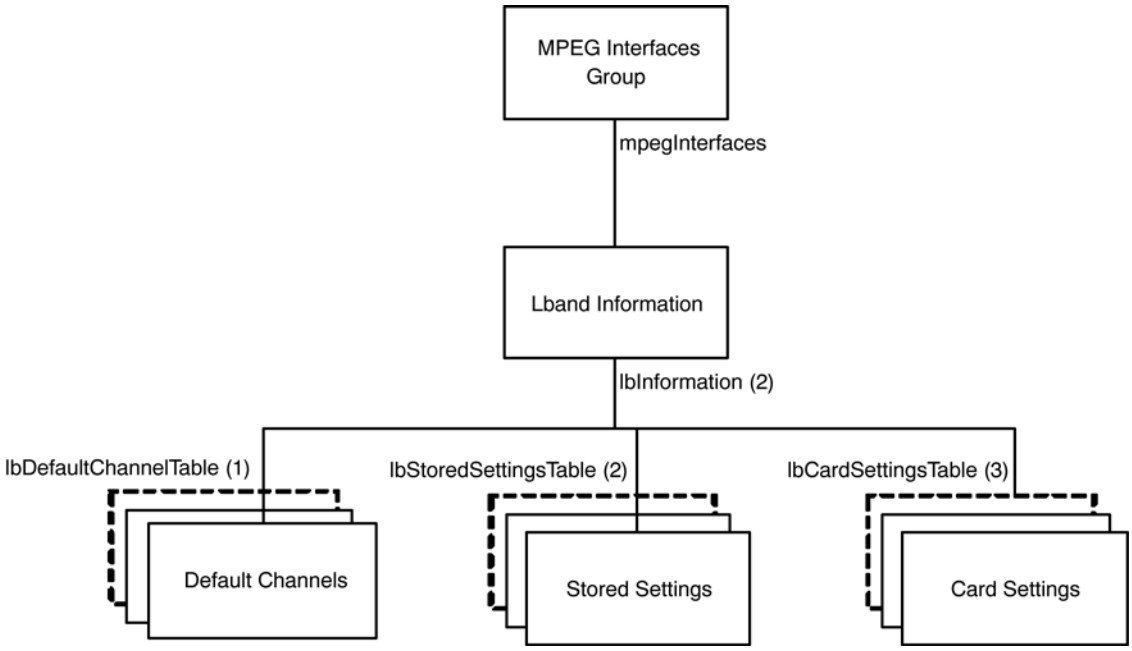


Figure 5–4: L-Band information group structure

Default Channels Table. The Default Channels table contains the name of the stored LBand Settings to use for each interface, and is defined as:

Variable	Type	Use	Access
LbDefaultChannelmifIndex(1)	Integer	The MPEG interface for which this default channel applies.	N/A
LbDefaultChannelName(2)	Octet String	The name of the selected stored settings channel. This is used to reference the required entry in the *Stored Settings* table on page 5–8.	Get/Set

The table is indexed by MPEG Interface, so in order to determine the name of the stored LBand settings for interface 1, use the following OID:

'…lbDefaultChannelName.1'.

Stored Settings Table. The Stored Settings table contains the available stored LBand settings that can be used for each interface, and is defined as:

Variable	Type	Use	Access
LbStoredmifIndex(1)	Integer	The MPEG interface for which these stored settings apply.	N/A
lbStoredChannelIndex(2)	Integer	Index to the stored channel settings used for this interface.	N/A
lbStoredName(3)	Octet string	The name given to these stored settings.	Get
lbStoredLoFreq(5)	Integer	Local Oscillator Frequency (kHz).	Get/Set
ldStoredTrFreq(6)	Integer	Transponder Frequency (kHz).	Get/Set
lbStoredPolarization(7)	Integer	Polarization (Volts) 0 = off, 1 = 13(V), 2 = 18(H).	Get/Set
lbStoredSymRate(8)	Integer	Symbol Rate (kSps).	Get/Set
lbStoredViterbiRate(9)	Integer	0 = 1/2, 1 = 2/3, 2 = 3/4 , 3 = 4/5, 4 = 5/6, 5 = 6/7	Get/Set
lbStoredViterbiRateAuto(10)	Integer	Sets ViterbiRateAuto 0 = off, 1 = on.	Get/Set
lbStoredTone22K(11)	Integer	Sets 22KHz tone 0 = off, 1 = on.	Get/Set
lbStoredInvertSpectrum(12)	Integer	Sets invert spectrum 0 = off, 1 = on.	Get/Set

The table is indexed by MPEG Interface followed by Channel Index. The stored LBand settings are persistent across all interfaces, so the Channel Index is used to reference which settings should be used from this global list. This has the consequence that if any of these values are changed on one interface, it will be changed across all interfaces. As an example, in order to reference the Transponder Frequency for interface 1, channel 2, use the following OID:

‘….lbStoredTrFreq.1.2’.

Card Settings Table. The Card Settings table contains the current settings for the LBand card, and is defined as:

Variable	Type	Use	Access
lbCardmifIndex(1)	Integer	The MPEG interface for which these card settings apply.	N/A
lbCardValidSettings(2)	Integer	Determines whether the LBand settings for this interface are valid (if this interface supports an LBand card: 1 = true, 0 = false).	Get/Set
lbCardLoFreq(4)	Integer	Local Oscillator Frequency (kHz).	Get/Set
ldCardTrFreq(5)	Integer	Transponder Frequency (kHz).	Get/Set
lbCardPolarization(6)	Integer	Polarization (Volts) 0 = off, 1 = 13(V), 2 = 18(H).	Get/Set
lbCardSymRate(7)	Integer	Symbol rate (kSps).	Get/Set
lbCardViterbiRate(8)	Integer	0 = 1/2 , 1 = 2/3, 2 = 3/4, 3 =5/6, 4 = 6/7, 5 = 7/8	Get/Set
lbCardViterbiRateAuto(9)	Integer	Sets ViterbiRateAuto 0 = off, 1 = on.	Get/Set
lbCardTone22K(10)	Integer	Sets 22kHz tone 0 = off, 1 = on.	Get/Set
lbCardFrontEndLock(11)	Integer	Determines whether Front End Lock is on.	Get
lbCardBER(12)	Integer	The BER. See below for specific values.	Get
lbCardInvertSpectrum (13)	Integer	Sets Invert Spectrum 0 = off, 1 = on.	Get/Set
lbCardMER(14)	Integer	MER db * $10^{6.}$	Get
lbCardActualBER(15)	Integer	BER Ratio * $10^{9.}$	Get
lbCardEVM(16)	Integer	EVM % * $10^{6.}$	Get
lbCardTEFCount(17)	Integer	TEF count.	Get
lbCardSignal(18)	Integer	Signal Strength % * $10^{6.}$	Get

The table is indexed on MPEG Interface. As an example, in order to reference the Viterbi Rate for interface 1, use the following OID:

‘….lbCardViterbiRate.1’.

The BER values returned have the following meanings:

{1.0e-1, 1},	{1.3e-2, 11},	{6.0e-5, 21},
{9.0e-2, 2},	{1.0e-2, 12},	{3.0e-5, 22},
{8.0e-2, 3},	{7.0e-3, 13},	{1.0e-5, 23},
{7.0e-2, 4},	{5.5e-3, 14},	{4.0e-6, 24},
{6.0e-2, 5},	{3.0e-3, 15},	{1.0e-6, 25},
{5.0e-2, 6},	{1.5e-3, 16},	{1.0e-7, 26},
{4.0e-2, 7},	{1.0e-3, 17},	{1.0e-8, 27},
{3.0e-2, 8},	{5.5e-4, 18},	{1.0e-9, 28}
{2.5e-2, 9},	{3.0e-4, 19},	
{1.7e-2, 10},	{1.5e-4, 20},	

g703Table. N/A

g703PlusTable. N/A

GPSITable. N/A

SI Repetition Rate Table. N/A

TMCC Basic Information Table. The TMCC Basic Information table contains the information stored in the first eight bytes of TMCC blocks for each interface. In order for the MTM400 to process the TMCC information, tmccAcquisition must be set to 1 for the appropriate stream.

Variable	Type	Use	Access
tmccmifIndex(1)	Integer	Index	N/A
tmccAcquisition(2)	Integer	Specifies whether to extract TMCC information.	Get/Set
tmccBufferReset(3)	Integer	Determines whether the buffer is reset.	Get
tmccEmergencySignal(4)	Integer	Determines whether the emergency signal is on.	Get
tmccChangeIndication(5)	Integer	Determines whether the change indication is set.	Get
tmccBeginningOfFrame(6)	Integer	Determines whether it is the beginning of a frame.	Get

Variable	Type	Use	Access
tmccBeginningOfSuper-Frame(7)	Integer	Determines whether it is the beginning of a superframe.	Get
tmccTransmissionMode1(8)	Octet string	The first transmission mode.	Get
tmccSlotAllocation1(9)	Integer	The first slot allocation.	Get
tmccTransmissionMode2(10)	Octet string	The second transmission mode.	Get
tmccSlotAllocation2(11)	Integer	The second slot allocation.	Get
tmccTransmissionMode3(12)	Octet string	The third transmission mode.	Get
tmccSlotAllocation3(13)	Integer	The third slot allocation.	Get
tmccTransmissionMode4(14)	Octet string	The fourth transmission mode.	Get
tmccSlotAllocation4(15)	Integer	The fourth slot allocation.	Get
tmccTransportID(16)	Integer	The transport ID.	Get
tmccRawBytes(17)	Octet string	Raw eight bytes of TMCC data.	Get

ASI Plus Information Table. N/A

Version Table. N/A

Physical Information Table. N/A

Stored Settings Table. N/A

QAM Information Group. Figure 5–5 shows the structure of the QAM Information Group, which contains information on the QAM Settings where appropriate.

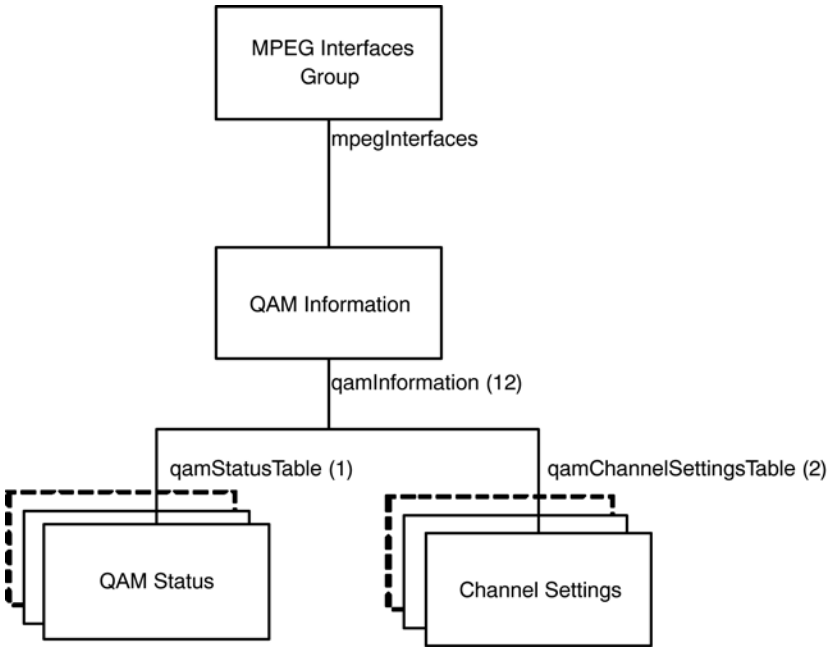


Figure 5–5: QAM information group structure

Status Table. The Status table contains the name of the selected channel settings and the status of the QAM card. The table is defined as:

Variable	Type	Use	Access
qamStatusmifIndex (1)	Integer	The MPEG interface for which this channel applies.	N/A
qamCurrentChannelName (2)	Octet string	The name of the selected stored channel. This is used to reference the required entry in the *QAM Channel Settings* table, see page 5–14.	Get/Set
qamFrontEndLock(3)	Integer	Boolean indicating the state of the front end lock 0 - no lock, 1 - in lock.	Get
qamSignalStrength (4)	Integer	The signal strength 1 to 5.	Get

Variable	Type	Use	Access
qamBER (5)	Integer	The BER (0 to 255). (See BER values below for specific values.)	Get
qamCardMER(6)	Integer	MER db * 10^6.	Get
qamCardActualBER(7)	Integer	BER Ratio * 10^9.	Get
qamCardEVM(8)	Integer	EVM % * 10^6.	Get
qamCardTEFCount(9)	Integer	TEF count.	Get
qamCardSignal(10)	Integer	Signal Strength % * 10^6.	Get

The table is indexed by MPEG Interface, so in order to determine the name of the stored QAM channel settings for interface 1, use the following OID:

‘…qamCurrentChannelName.1’.

The BER values returned have the following meanings:

{1.0e-1, 1},
{9.0e-2, 2},
{8.0e-2, 3},
{7.0e-2, 4},
{6.0e-2, 5},
{5.0e-2, 6},
{4.0e-2, 7},
{3.0e-2, 8},
{2.5e-2, 9},
{1.7e-2, 10},
{1.3e-2, 11},
{1.0e-2, 12},
{7.0e-3, 13},
{5.5e-3, 14},
{3.0e-3, 15},
{1.5e-3, 16},
{1.0e-3, 17},
{5.5e-4, 18},
{3.0e-4, 19},
{1.5e-4, 20},
{6.0e-5, 21},
{3.0e-5, 22},
{1.0e-5, 23},
{4.0e-6, 24},
{1.0e-6, 25},
{1.0e-7, 26},
{1.0e-8, 27},
{1.0e-9, 28}

QAM Channel Settings Table. The QAM Channel Settings table contains the stored QAM settings that can be used for each interface, and is defined as:

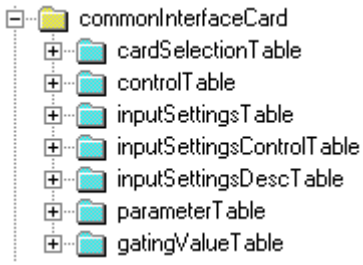
Variable	Type	Use	Access
qamChannelSettingsmifIndex (1)	Integer	The MPEG interface for which these channel settings apply.	N/A
qamChannelIndex (2)	Integer	Index to the stored channel settings used for this interface.	N/A
qamChannelName (3)	Octet string	The name given to these channel settings.	Get
qamChannelRxFreq (4)	Integer	The rx frequency of the channel in Hz.	Get/Set
qamChannelSymRate (5)	Integer	The symbol rate of the channel.	Get/Set
qamChannel2LoFreq (6)	Integer	The 2nd Local Oscillator frequency of the channel in Hz.	Get/Set
qamChannelConstellation (7)	Integer	The constellation (trellis patters) of the channel. The MIB value is mapped to the constellation as follows: 0 = 4, 1 = 16, 2 = 64, 3 = 256.	Get/Set
qamChannelInversion (8)	Integer	Specifies inversion for the channel. 0= not inverted, 1= inverted.	Get/Set
qamChannelVControl (9)	Integer	Specifies V Control for the channel. 0 = off, 1 = on.	Get/Set

Variable	Type	Use	Access
qamChannelUncorrectable PacketMode(10)	Integer	Specifies the uncorrectable packet mode. The values have the following meanings : 0 = Do not filter bad packets, 1 = Filter bad packets, 2 = Do not send uncorrupted event status messages, 3 = Send uncorrupted event status messages.	Get/Set
QamChannelLockConfidence (11)	Integer	Reserved.	Get/Set
QamChannelCorrection Confidence (12)	Integer	Reserved.	Get/Set
qamChannelCarrierReceiverLoopBandwidth(13)	Integer	0 = Normal, 1 = Wide.	Get/Set
qamChannelCarrier AcquisitionRange(14)	Integer	0 = Normal, 1 = Wide.	Get/Set
qamChannelTroubleShoot(15)	Integer	0 = off, 1 = on.	Get/Set

The table is indexed by the MPEG Interface followed by the Channel Index. The QAM channel settings are persistent across all interfaces, so the Channel Index is used to reference which settings should be used from this global list. This has the consequence that if any of these values are changed on one interface, it will be changed across all interfaces.

Common Interface Cards

This group is used to control the common interface (CIP) cards, the CIP platform carries a number of demodulators so the settings change according to card type.



Card Selection Table. This table allows the required CommonInterface Card to be selected. It contains the following elements.

Variable	Type	Use	Access
MifIndex (0)	Table index		N/A
currentCard (1)	Integer	The index of the required card, corresponding to the CardNumber in the configuration file and the CardDetails HTTP query. Get this value to determine the current card selected. Set this value to change the card.	N/A

Control Table

This table allows a card to be reset, and the input to be selected, and contains the following elements:

Variable	Type	Use	Access
MifIndex (0)	Table index		N/A
cardIndex	Table index		N/A
resetCard	Integer	Setting this to any value resets the card.	Set
currentInput	Integer	The index of the required input. Get this value to determine the current input selected. Set this value to change the input.	Get/Set

Input Settings Table

This table allows the settings to be applied to an input of the card, and will contain the following elements:

Variable	Type	Use	Access
MifIndex (0)	Table index		N/A
cardIndex	Table index		N/A
inputIndex	Table index		N/A
inputSettingsName	Octet String	The name of the settings applied to the input and the configuration file. Get this value to determine the name of the setting currently being used. Set this value to change the settings used.	Get/Set

Input Settings Control Table

This table allows settings to be added and deleted, and contains the following elements.

Variable	Type	Use	Access
MifIndex (0)	Table index		N/A
cardIndex	Table index		N/A
inputIndex	Table index		N/A
addSettings	Octet String	Set this value to add settings with the specified name to the list of settings for the input.	Set
deleteSetting	Octet String	Set this value to delete the settings with the specified name from the list of settings for the input. This operation will fail if the settings specified are currently set as the inputSettingsName in the Input Settings Table.	Set

Input Settings Description Table

This table contains the descriptions of the settings that can be applied to the input of a card, and contains the following elements.

Variable	Type	Use	Access
MifIndex (0)	Table index		N/A
cardIndex	Table index		N/A
inputIndex	Table index		N/A
settingsIndex	Table index		N/A
settingsName	Octet String	The name of the settings for the settings index. Get this value to determine the current name of the settings. Set this value to change the name of the settings - this will fail if the settings are currently being used.	Get/Set

Parameters Table

This contains the parameter values for the input of a card, and consists of the following elements

Variable	Type	Use	Access
MifIndex (0)	Table index		N/A
cardIndex	Table index		N/A
inputIndex	Table index		N/A
settingsIndex	Table index		N/A
parameterIndex	Table index		N/A
parameterValue	Octet String	The value of the parameter. Get this value to determine the current value. Set this value to change the value.	Get/Set

Gating Value Table

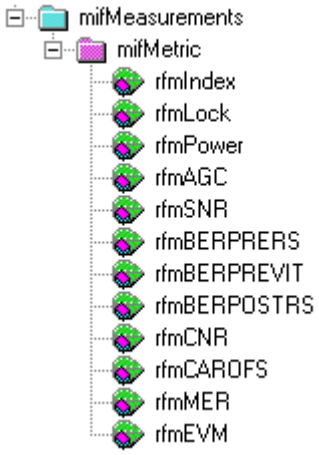
This contains the gating value for the input of a card, and consists of the following elements:

Variable	Type	Use	Access
MifIndex (0)	Table index		N/A
cardIndex	Table index		N/A
inputIndex	Table index		N/A

Variable	Type	Use	Access
settingsIndex	Table index		N/A
gatingValue	Integer	The gating value for the input. 0 = off, 1 = on, 2 = auto. Get this value to determine the current value. Set this value to change the value.	Get/Set

Measurements

This section returns the metrics from CIP cards. The cards vary so not all measurements are appropriate to all cards.



Index.

Name:	rfmIndex
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.1
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmIndex(1)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	A unique value identifying a particular MPEG interface metric. The index for this table.

Lock.

Name:	rfmLock
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.2
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmLock(2)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	Lock status of the interface, 0=unlocked 1=locked

Power.

Name:	rfmPower
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.3
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmPower(3)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	Input power in dBm * 100

AGC.

Name:	rfmAGC
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.4
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmAGC(4)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	AGC in % * 100

SNR.

Name:	rfmSNR
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.5
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmSNR(5)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	SNR in dBm * 100

BER Pre RS error correction.

Name:	rfmBERPRERS
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.6
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmBERPRERS(6)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	BER pre reed-solomon correction * 1e9

BER Pre Viterbi correction.

Name:	rfmBERPREVIT
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.7
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1).mifMeasure ments(14).mifMetric(1).rfmBERPREVIT(7)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	BER pre viterbi error correction * 1e9

BER Post RS error correction.

Name:	rfmBERPOSTRS
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.8
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmBERPOSTRS(8)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	BER post reed-solomon correction * 1e9

CNR.

Name:	rfmCNR
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.9
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmCNR(9)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	CNR in dBm * 100

Carrier Offset.

Name:	rfmCAROFS
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.10
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmCAROFS(10)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	Carrier offset in Hz

MER.

Name:	rfmMER
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.11
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmMER(1 1)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	MER in dBm * 100

EVM.

Name:	rfmEVM
OID:	1.3.6.1.4.1.128.5.1.17.1.14.1.12
Full path:	iso(1).org(3).dod(6).internet(1).private(4).enterprises(1). tek(128).tvt(5).tvtproducts(1).adMpeg(17).mpegInterfaces(1). mifMeasurements(14).mifMetric(1).rfmEVM(12)
Module:	AD-MPEG-MIB
Parent:	mifMetric
Numerical syntax:	Integer (32 bit)
Max access:	read-only
Description:	EVM in % * 100

MPEG Events Group

The MTM400 may generate several events for each MPEG interface. Normally, an event may be in one of five states:

- 'Red' (0x3xxx) indicates that there is currently an error condition.
- 'Yellow' (0x2000) indicates that there is currently no error condition, but that one has occurred since this event was last reset.
- 'Green' (0x1000) indicates that there is no error condition.
- 'Gray' (0x0000) indicates the state is unknown (or link lost).
- 'White' (0x4000) indicates that the event is disabled.

Each event also has an alarm value associated with it, which indicates the type of alarm that will be triggered (for example, audible or relay), if an error occurs. The full list of box events is specified in the MTM400 Technical Reference (Tektronix part number: 071-1560-xx).

Figure 5–6 shows the structure of the MPEG Events Group, which contains information on the states and alarm values of events on each MPEG Interface.

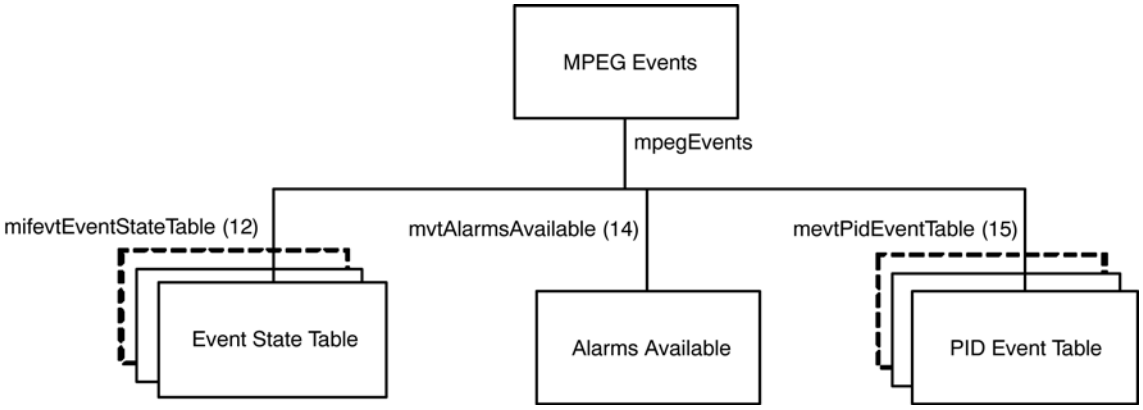


Figure 5–6: MPEG events group structure

Event State Table

The Event State table contains the state of each MPEG Interface event on every interface, and is defined as:

Variable	Type	Use	Access
mifevtMifIndex(1)	Integer	The MPEG interface for which these events apply.	N/A
mifevtEventIndex(2)	EvId	An index uniquely identifying the event.	N/A
mifevtEventName(3)	Octet String	A short textual title for this event.	Get
mifevtEventDescription(4)	Octet String	A brief description of this event.	Get
mifevtEventState(5)	EvState	The state of this event. Writing any value will reset the event. The effect of resetting is to change a 'yellow' event state to either 'green' or 'unknown'.	Get/Set
mifevtAlarmValue(6)	AlmValue	The alarms that will be triggered for this event.	Get/Set
mifevtEventEnable(7)	Simple Boolean	Specifies whether this event is enabled (0 = disabled, 1 = enabled).	Get/Set

Indexing. The table is indexed by MPEG Interface followed by EvId. As an example, in order to reference the alarm value of event 0x2000 (8192) on interface 1, use the following OID '…mifevtAlarmValue.1.8192'.

Unsupported Events. Events that are not supported on an interface will have an event state of 0x0000.

Setting Event States. Setting an event that is in the 'Yellow' (0x2000), to any value, resets the event. Setting an event with a 'Red' state has no effect, because this indicates that there is a persistent error.

Setting Alarm Values. An alarm value specifies which alarms will be triggered when an error occurs in the corresponding event.

The value is a combination of those specified in AlmValue, (for example, 0x00020401 will set TTL2, Relay3, and Audible alarms to be triggered).

Alarms Available

Variable	Type	Use	Access
mevtAlarmsAvailable(14)	AlmValue	Indicates the types of alarms that can be triggered for stream events.	Get

The value returned is a bitfield that shows which alarm action can be enabled/disabled.

Buzzer = 0x00000001
Recorder = 0x00000020
Relay1 = 0x00000100
Relay2 = 0x00000200
Relay3 = 0x00000400
Relay4 = 0x00000800
Relay5 = 0x00001000
TTL1 = 0x00010000
TTL2 = 0x00020000
TTL3 = 0x00040000
TrapRaise = 0x00100000
TrapClear = 0x00200000
Logging = 0x01000000

PID Event Table

The PID Event table contains a table of MPEG PID specific events on every interface, and is defined as:

Variable	Type	Use	Access
mevtPidMifIndex (1)	Integer	The MPEG interface for which these events apply.	N/A
mevtPidEventIndex (2)	Evid	A unique index identifying a particular type of PID event. The values for this index are prescribed, and can be found in Appendix A of the MIB Specification.	N/A
mevtPidPidIndex (3)	Integer	The PID number + 1.	N/A

Variable	Type	Use	Access
mevtPidEventState (4)	Evstate	Reading this returns the current event status for the PID as described for the EvState type. Writing any value will reset the event. The effect of resetting is to change a 'yellow' event state to either 'green' or 'unknown'.	Get/Set
mevtPidEventEnable (5)	Simple Boolean	If a per PID event is disabled, the EvState will always be reported as 'disabled', no alarms will be generated for the event and the system does not need to perform any processing associated with the event.	Get/Set

MPEG PIDs Group

Figure 5–7 shows the structure of the PIDs Group, which contains PID, PID Group and Program limit and rate information:

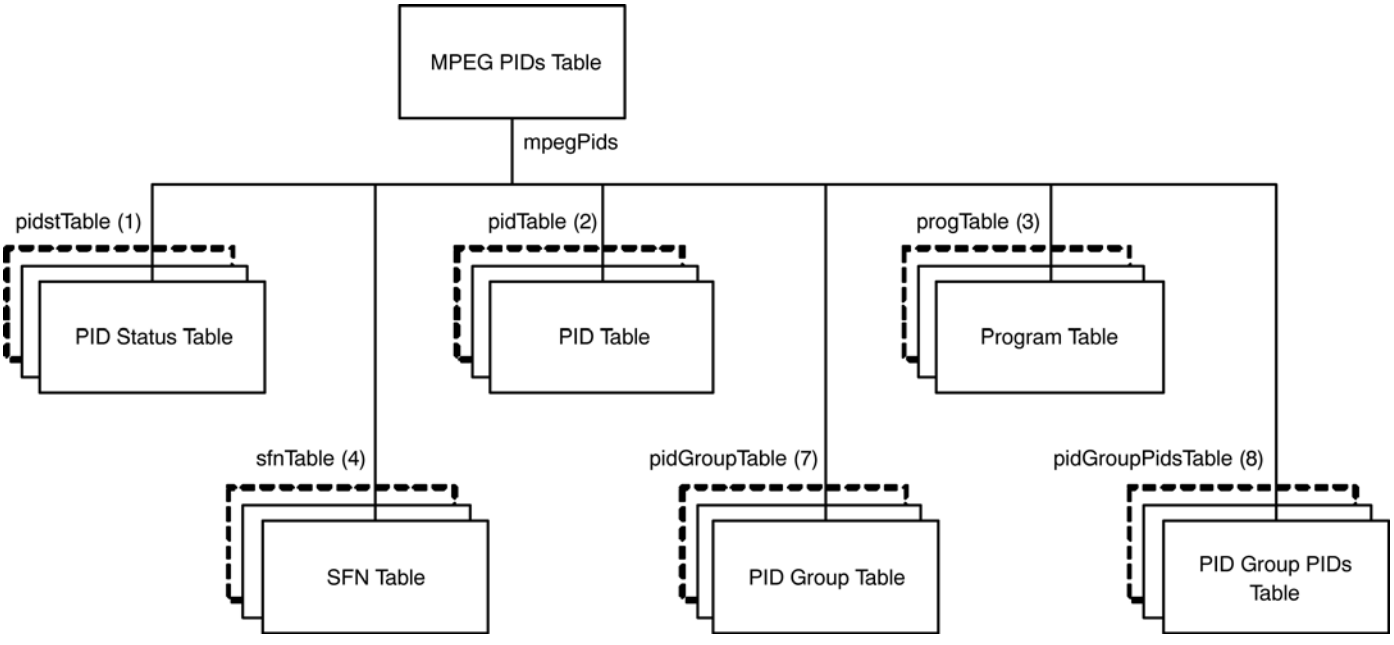


Figure 5–7: MPEG PIDs group structure

PID Status Table

The PID Status table contains PID and Program status information for each interface, and is defined as:

Variable	Type	Use	Access
pidstMifIndex(1)	Integer	MPEG interface for which these elements apply.	N/A
pidstClearLimits (2)	Integer	Setting this clears all the PID rate limits for this interface. Reading this value has no meaning.	Get/Set
progstClearLimits (3)	Integer	As above, but for programs.	Get/Set
pidstResetRates(4)	Integer	Setting this resets all PID minimum and maximum rate measurements for this interface. Reading this value has no meaning.	Get/Set
progstResetRates(5)	Integer	As above, but for programs.	Get/Set
pidgroupstClearLimits (10)	Integer	Setting this clears all the PID group rate limits for this interface. Reading this value has no meaning.	Get/Set
pidgroupstResetRates (11)	Integer	Writing any value to this object will reset the currently latched minimum and maximum bit rates for all PID groups. Reading this value has no meaning.	Get/Set
pidgroupstNewPidGroupIndex(12)	Integer	Reading this will create a new pid group on the MTM400 device. The value returned is the group index. This is used to index this group in the pidGroupTable and pidGroupPidsTable.	Set
pidgroupstDeletePidGroupIndex (13)	Integer	Writing a value will delete the group with the index specified by the value set.	Get/Set

The table is indexed by MPEG Interface. As an example, in order to reference pidsResetRates for interface 1, use the following OID:

‘…pidstResetRates.1’.

PID Table

The PID table contains information for each PID on each interface, and is defined as:

Variable	Type	Use	Access
pidsMifIndex(1)	Integer	The MPEG interface for which these readings apply.	N/A
pidsPidIndex(2)	Integer	The PID index - this is the PID number + 1 to avoid a 0 index.	N/A
pidsRate(3)	Integer	The most recently measured rate for this PID.	Get
pidsMinRate(4)	Integer	The minimum rate latched for this PID since last reset.	Get
pidsMaxRate(5)	Integer	The maximum rate latched for this PID since last reset.	Get
pidsMinLimit(6)	Integer	The minimum limit for this PID.	Get/Set
pidsMaxLimit(7)	Integer	The maximum limit for this PID.	Get/Set
pidsState(8)	Evstate	The state of this PID.	Get/Set
pidsScrambled (11)	Simple boolean	0 = PID not scrambled, 1 = PID scrambled.	Get
pidsUnreferenced(13)	Simple boolean	Indicates whether the PID is un-referenced. 1 = un-referenced, 0 = referenced.	Get
pidsForceListPresence (15)	Simple boolean	Specifies whether the PID must exist in this list, event if it does not appear in the transport stream. 0 = PID not present, 1 = PID present.	Get/Set
pidsVariability (16)	Octet string	Textual representation of variability (floating point number).	Get
pidsISDBTLayer(18)	Integer	Indicates which ISDB-T layer the PID is transmitted on. (1=A, 2=B, 3=C)	Get

- **Indexing**. The table is indexed by the MPEG Interface, followed by the PID Index. As an index of 0 is not allowed in SNMP tables, the PID Index is actually PID+1. Therefore, in order to reference the required PID item, for example pidsMinLimit, use the following OID:

  '…pidsMinLimit.<interface>.<pid+1>'.

- **Reading PID Information**. The list of PIDs for which readings are available can change fairly rapidly, so the management application must be notified

that subsequent requests for PID elements may result in values for a different set of PIDs. Consequently, if a client application requests all of the pidsMinRates followed by pidsMaxRates, it is not guaranteed that the values obtained will be for exactly the same set of PIDs. Therefore, in order to force the agent to include a PID in its list, the management application should set the corresponding Min and Max limits.

- **PID Limits**. By default, the limits for each PID are not defined; this is represented by the pidsMinLimit and pidsMaxLimit values being set to 0 and -1 respectively. When setting a limit, the management application must ensure that the value of pidsMaxLimit is always greater than pidsMinLimit, otherwise the new setting will not be accepted by the MTM400. The new and current values of pidsMinLimit and pidsMaxLimit will therefore affect the order in which the management application sets these limits.

  The limits for a PID can be cancelled at a later date by setting the pidsMinLimit to 0 and then setting pidsMaxLimit to -1. Although -1 is less than 0, this is a special case, which is accepted by the MTM400.

- **PID Occupancy Events**. The MPEG Interface event 0x2001 will be generated whenever any PID occupancy exceeds its limits. The management application can choose to poll this at the required interval.

Program Table

The following are the objects in the program table in the PID group:

Variable	Type	Use	Access
progsMifIndex(1)	Integer	The MPEG interface for which these readings apply.	N/A
progsProgIndex(2)	Integer	The program index - this is the program number + 1 to avoid a 0 index.	N/A
progsRate(3)	Integer	The most recently measured rate for this program.	Get
progsMinRate(4)	Integer	The minimum rate latched for this program since last reset.	Get
progsMaxRate(5)	Integer	The maximum rate latched for this program since last reset.	Get
progsMinLimit(6)	Integer	The minimum limit for this program.	Get/Set
progsMaxLimit(7)	Integer	The maximum limit for this program.	Get/Set
progsState(8)	Evstate	The state of this program.	Get

Variable	Type	Use	Access
progsPMTTestEnabled(9)	Simple boolean	Determines whether PMT Test is enabled for this program. 0 = disabled, 1 = enabled.	Get/Set
progsPMTTestState (11)	Evstate	Reading this returns the current state of the PMT Test for the program. Writing any value will reset the PMT Test for the program. The effect of resetting is to change a 'yellow' event state to either 'green' or 'unknown'.	Get/Set

- **Indexing**. The table is indexed by MPEG Interface, followed by Program Index. An index of 0 is not allowed in SNMP tables, so the Program Index is actually Program + 1. Therefore, in order to reference the required Program item, for example progsMinLimit, use the following OID:

  '…pidsMinLimit.<interface>.<prog+1>'.

- **Program Limits**. By default, the limits for each Program are not defined; this is represented by the progsMinLimit and progsMaxLimit values being set to 0 and -1 respectively. When setting a limit, the management application must ensure that the value of progsMaxLimit is always greater than progsMinLimit, otherwise the new setting will not be accepted by the MTM400. The new and current values of progsMinLimit and progsMaxLimit will therefore affect the order in which the management application sets these limits.

  The limits for a Program can be cancelled at a later date by setting progsMinLimit to 0, and then setting progsMaxLimit to -1. Although -1 is less than 0, this is a special case, which is accepted by the MTM400.

- **Program Occupancy Events**. The MPEG Interface event 0x2002 will be generated whenever any Program occupancy limit is exceeded. The management application can choose to poll this at the required interval.

SFN Table

The SFN table contains the Single Frequency Network Information for each interface, and is defined as:

Variable	Type	Use	Access
sfnMifIndex(1)	Integer	MPEG interface for which these elements apply.	N/A
sfnSynchronisation (2)	Integer	The SFN Synchronization Scheme (usually 0).	Get
sfnSectionLength (3)	Integer	Number of bytes following the section_length field.	Get
sfnPointer(4)	Integer	Number of transport packets between the MIP and the first packet of the succeeding Mega Frame.	Get
sfnPeriodicFlag(5)	Integer	0 = aperiodic, 1 = periodic insertion of the MIP.	Get
sfnSynchronisationTimeS-tamp(6)	Integer	Time difference between the latest pulse of the 'one pulse per second' reference and the actual start of this Mega Frame in units of 100 ns.	Get
sfnMaximumDelay(7)	Integer	Delay between start of Mega Frame at the antenna, and the start of it at the SFN adapter in units of 100 ns.	Get
sfnTPSMip(8)	Octet string	Four bytes containing bit-stream P0-P31 of the Transport Parameter Signaling (TPS) information defined in TS 101 191 V1.2.1.	Get
SfnIndividualAddressing-Length(9)	Integer	Total length of the individual addressing field in bytes.	Get
sfnMegaFrameSize(10)	Integer	Calculated Mega Frame Size.	Get
sfnDelay(11)	Integer	Calculated Delay.	Get
sfnInaccuracy(12)	Integer	Calculated Inaccuracy.	Get
sfnFunctionBytes(13)	Octet string	The bytes immediately follow-ing the individual_address-ing_length field of the MIP up to the crc_32, which contains the function descriptors.	Get
sfnExists(14)	Integer	Indicates whether the SFN PID (0x15) exists in the transport stream. 0 = false, 1= true.	Get

The table is indexed by MPEG Interface. As an example, in order to reference sfnTPSMip for interface 1, use the following OID:

'…sfnTPSMip.1'.

PID Group Table

The PID Group table provides access to PID group related information for each interface, and is defined as:

Variable	Type	Use	Access
pidGroupMifIndex (1)	Integer	MPEG interface for which these PID groups apply.	N/A
pidGroupIndex (2)	Integer	The index of this group.	N/A
pidGroupName (3)	Octet string	The PID group name.	Get/Set
pidGroupRate (4)	Integer	The most recently measured bit rate of this PID group. Units are bit/s.	Get
pidGroupMinRate (5)	Integer	The lowest measured bit rate of this PID group since the minimum measured rate was last reset. Units are bit/s.	Get
pidGroupMaxRate (6)	Integer	The highest measured bit rate of this PID group since the maximum measured rate was last reset. Units are bit/s.	Get
pidGroupMinLimit (7)	Integer	The lower bit rate limit on this PID group. Units are bit/s.	Get/Set
pidGroupMaxLimit (8)	Integer	The upper bit rate limit on this PID group. Units are bit/s.	Get/Set
pidGroupState (9)	Evstate	Reading this returns the current event status with respect to whether the PID group's bit rate has gone outside the bit rate limits. See the EvState type. Writing any value will reset the 'PID Group Occupancy' event.	Get
pidGroupNewPid (11)	Integer	Setting this value adds the PID specified to the group. Reading this field has no meaning.	Get/Set
pidGroupDeletePid (12)	Integer	Setting this value deletes the PID specified from the group. Reading this field has no meaning.	Get/Set

PID Group PIDs Table

The PID Group PIDs table provides access to the lists of PIDS defined for each group. It is defined as follows:

Variable	Type	Use	Access
pidGroupPidsMifIndex (1)	Integer	MPEG interface for which these PID group PIDs apply.	N/A
pidGroupPidsGroupIndex (2)	Integer	The index of the group of interest.	N/A
pidGroupPidsPidIndex (3)	Integer	The PID plus 1. This index is one greater than the number of the PID because PID 0 is valid, but an index of 0 into an SNMP table is not.	N/A
pidGroupPidsInGroup (4)	Simple Boolean	Specifies whether the PID (as specified by pidGroupPidsPi-dIndex - 1) belongs to the group. Setting this to 0 will remove the PID from the group.	Get

MPEG Structure Group 2

This provides access to the unformatted raw byte stream information stored in the MPEG Tables that describe the structure of MPEG transport streams.

There are two main problems with attempting to provide MPEG structure information through an SNMP interface. Firstly, the amount of information stored in MPEG Tables can grow to an arbitrarily large size, certainly more than the 484 bytes SNMP systems are required to support, and potentially larger than the maximum UDP packet size. Secondly, this information can change fairly rapidly.

In order to solve the first problem, the information for each MPEG Table is split up into manageable ‘chunks’ with a maximum size of 128 bytes. However, the second problem of potential rapid updates means that the MPEG table information can change between reading the separate chunks. Consequently, serial numbers are used to represent versions of MPEG Tables at particular times.

Figure 5–8 shows the way in which the MPEG transport stream information is represented within the MTM400 MIB. SNMP tables have been used to represent the data stored in MPEG Tables, and it is possible that some confusion may arise over terminology, consequently specific reference has been made as to whether MPEG or SNMP tables are being discussed in the descriptions below.

In the MPEG standard, each MPEG Table has an identifier, which is represented as a single byte value. For example, the Program Association Table has a table id

of 0x00. The use of these MPEG Table identifiers within the Structure Group is consistent with this standard.

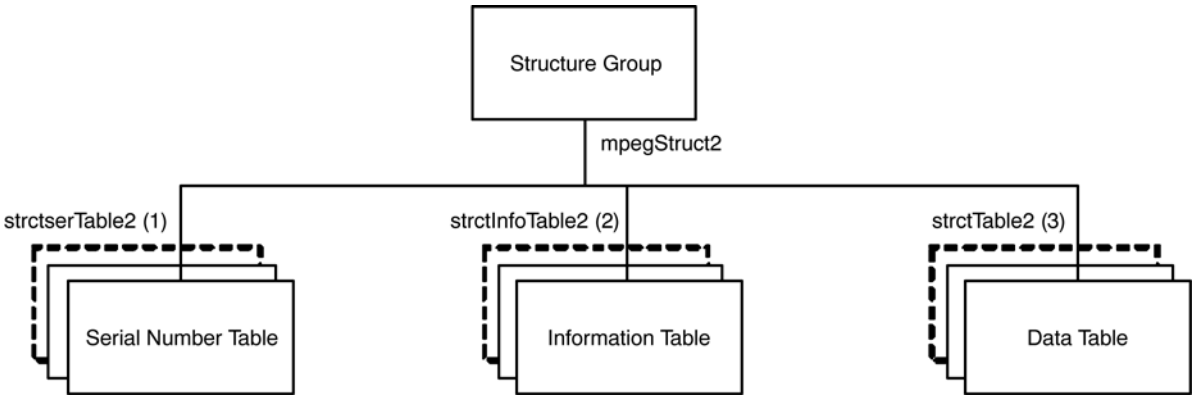


Figure 5–8: Structure group 2 structure

Serial Number Table

The SNMP Serial Number table contains the serial numbers that should be used to index the SNMP Size and Data tables in order to obtain the most up-to-date information for each MPEG Table. Each serial number is incremented every time its MPEG Table changes. Management applications attempting to use out-of-date serial numbers to read the Size and Data SNMP tables will receive SNMP 'No Such Name' errors. If this happens, they should attempt to obtain the new serial number for this table and start again. (MPEG Tables can also disappear completely without being replaced by a more up to date version, in which case, the management application will need to abort the operation.)

The Serial Number table is defined as follows:

Variable	Type	Use	Access
strctserMifIndex2(1)	Integer	The MPEG interface for which these readings apply.	N/A
strctserTableIndex2(2)	Integer	The MPEG Table Id (+1).	N/A
strctserMajorExtensionIn-dex2(3)	Integer	Top 32 bits of the sub-table unique identifier (+ 1).	N/A
strctsetMinorExtensionIn-dex2(4)	Integer	Bottom 32 bits of the sub-table unique identifier (+ 1).	N/A
strctserNumber2(5)	Integer	The serial number of the most up to date version of this MPEG Table.	Get

As an example, the OID '…strctserNumber2.1.67.1081.54' would return the most recent serial number for the DVB table id 66 (Service Description Table) where:

.1. = Stream 1 (default)

.67. = table id + 1

.1081.54 = unique identifier of the subtable

Info Table

This contains the total number of bytes stored for a specified version (referenced by serial number) of each MPEG Table on each interface. The size should be used to check that the correct numbers of bytes are read from the SNMP Data table. The table also contains the PID number on which the table was transmitted.

The table is defined as follows:

Variable	Type	Use	Access
strctInfoMifIndex2(1)	Integer	The MPEG interface for which these readings apply.	N/A
strctInfoTableIndex2(2)	Integer	The MPEG Table Id (+1).	N/A
strctInfoMajorExtensionIndex2(3)	Integer	Top 32 bits of the table unique identifier (+ 1).	N/A
strctInfoMinorExtensionIndex2(4)	Integer	Bottom 32 bits of the table unique identifier (+ 1).	N/A
stcrtInfoSerialIndex2(5)	Integer	The serial number of this table.	N/A
strctInfoSize2(6)	Integer	The number of bytes in this table.	Get
strctInfoPid2(7)	Integer	The PID this table was transmitted on.	Get

As an example, the OID '…strctInfoSize2.1.67.1081.54.2' would return the size of the table id 66 (SDT) with the serial number 2.

Data Table

This table contains the data from each version (referenced by serial number) of each MPEG Table on each interface split which has been split into ‘chunks’.

Variable	Type	Use	Access
strctMifIndex(1)	Integer	The MPEG interface for which these readings apply.	N/A
strctTableIndex2(2)	Integer	The MPEG Table Id (+1).	N/A
strctExtensionIndex2(3)	Integer	Top 32 bits of the table unique identifier (+ 1)	N/A
strctMinorExtensionIndex2(4)	Integer	Bottom 32 bits of the table unique identifier (+ 1)	N/A
stcrtSerialIndex2(5)	Integer	The serial number of this table.	N/A
strctChunkIndex2(6)	Integer	The chunk index of this table.	Get
strctTableData2(7)	Octet string	The raw bytes in this chunk.	Get

The data from the MPEG Table is split into sequential ‘chunks’ of up to 128 bytes, and the Chunk Index is the ‘chunk’ number that this TableData item represents. Management applications must concatenate the appropriate ‘chunks’ together in order to reconstruct the data contained in the corresponding MPEG Table.

As an example, the following OIDs would return all of the data for serial number 2 of MPEG Table 66 (SDT) , assuming it was split into 3 ‘chunks’:

‘…strctTableData2.1.67.1081.54.2.1’,
‘…strctTableData2.1.67.1081.54.2.2’
and ‘…strctTableData2.1.67.1081.54.2.3’

strctChunkIndex2(6) To download the entire table, each successive chunk must be read and the resulting chunks appended. Using get next operations on this index is the way to work through all of the chunks.

It is important to check the returned OID to make sure you are still downloading chunks for the table you thought you were - tables can go away, and a get next operation will simply start with the next table, or even some other bit of MIB tree entirely if there are no more tables.

strctTableData2(7) The MPEG structure table data itself. The size of this object is determined by the strctChunkSize object, unless this is the last chunk in a table, in which case it may be smaller than the current chunk size.

The complete MPEG structure table for interface i, table number t, serial number s is formed by concatenating all of the instances of this object of the form:

strctTableData.i.t+1.s.*

where '*' indicates all values of strctChunkIndex. The chunks should be reassembled with these final index values in ascending order. The contents of these tables is determined by the way in which they are being used on a given MPEG stream. This part of the MIB module makes no attempt to interpret these structure tables in any way, it just makes them available for download as raw byte streams.

MPEG Log Group

Figure 5–9 shows the structure of the MPEG Log Group, which provides access to the stream specific log items.

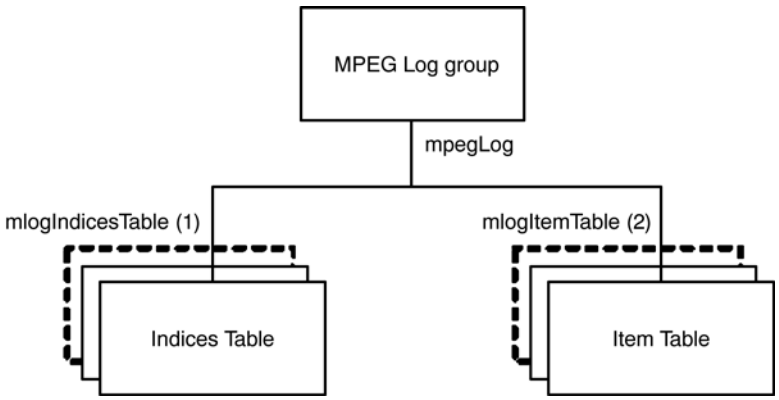


Figure 5–9: MPEG log group structure

Indices Table

The Indices table contains the most recent and oldest indices of the log entries for each stream, and is defined as:

Variable	Type	Use	Access
mlogIndecesMifIndex (1)	Integer	The MPEG Interface.	N/A
mlogRecentIndex (2)	LogIndex	The index of the most recent log entry on this interface.	Get

Variable	Type	Use	Access
mlogOldestIndex (3)	LogIndex	The index of the most oldest log entry on this interface.	Get
mlogClear (4)	Integer	Writing any value to this variable will clear the stream log. Reading this field has no meaning.	Get/Set

- **Indexing**. As the table index is an integer, this may wrap around if the number of entries in the log becomes significantly large. This means that the element with the largest index is not necessarily the latest log entry.

Item Table

The Item table contains the log entries for each interface, and is defined as:

Variable	Type	Use	Access
mlogItemMifIndex(1)	Integer	The MPEG Interface.	N/A
mlogItemIndex(2)	LogIndex	The log item index.	N/A
mlogItem(3)	Octet string	Byte stream containing the log entry (see below for details).	Get

- **Log Entry**. The mlogItem entry will be empty if the index requested is not valid. This occurs if the management application requests an entry that no longer exists, for example, if the log was full and the entry was deleted from the end of the list to make room for new entries. If mlogItem is not empty, the format of the octet string is as follows:

  All numeric values are coded L.S.B. first:

  Bytes 0..7: Public Timestamp Structure as defined in 2.1.5

  Bytes 8..9: Stream number (1 for MTM400)

  Bytes 10..11: EvId event id for the event

  Bytes 12..13: Event id extension (zero if not applicable)

  Bytes 14 onwards: Log text coded as UTF-8.

MPEG Trap Control

The trap control group provides the variables to support the traps sent and the configuration items to control trap generation.

Clients subscribe to traps by writing their IP address into trapSink; they are automatically deleted from the notification list after trapSinkTimeout minutes. So a client should subscribe every few minutes. TrapSinkTimeout may be 0, which means infinite.

TrapThrottle limits how many Traps per second may be generated, this is to stop the network being overloaded with traps. The limit is across the network, not per client, so if this is set to 10 and there are 2 clients, each will see up to 5 traps per second.

There is a single trap type, this has a payload that defines the event and associated data.

There is no mechanism to remove trap sinks, this is automatically achieved by the timeout. When the sink timeout has been set to 0, subscribers can be deleted by setting the timeout to a value > 0, subscribing and rebooting.

The timeout value is applied as the subscription takes place, so if one client set the timeout to 30 and subscribed, it would not be affected if a second client reduced the value to 5 and subscribed.

Name	Type	Access	Comment
TrapSink	IpAddress	WO	Clients write their IP address into this variable to register that they want to receive traps. Multiple subscribers may be active at any one time.
TrapThrottle	Integer	RW	Specify the maximum number of traps issued per second (traps * clients).
TrapEventID	EvId	RO	Data for last trap fired.
TrapStatus	Trap status	RO	
TrapTransportID	Integer	RO	
TrapNetworkID	Integer	RO	
TrapServiceID	Integer	RO	
TrapServiceType	Integer	RO	
TrapPID	Integer	RO	
TrapTimeStamp	Octet string	RO	

Name	Type	Access	Comment
TrapThresholdValue	String	RO	
TrapActualValue	String	RO	
TrapDuration	Integer	RO	
TrapStream	Integer	RO	Stream number, fixed at one in MTM400.
TrapSinkTimeout	Integer	RW	Minutes before unsubscribing trap client, 0 is infinite.

MPEG Configuration Group

Table 5–10 shows the structure of the MPEG Configuration Group that manages the stream configuration slots.

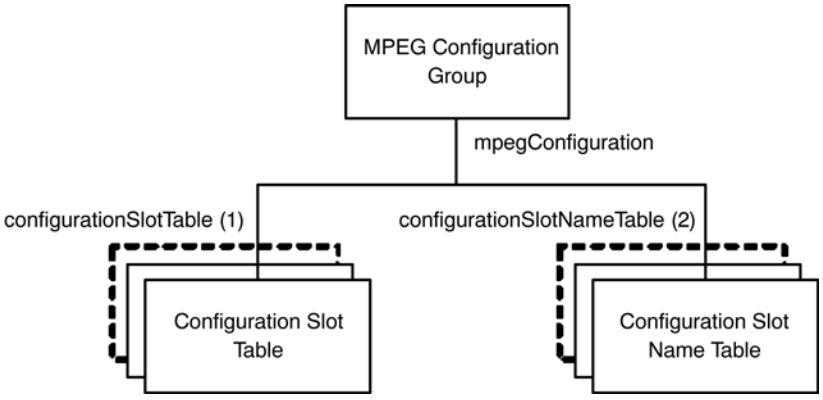


Figure 5–10: MPEG configuration group

Configuration Slot Table

The Configuration Slot table contains items for copying and storing stream configuration slots, and is defined as:

Variable	Type	Use	Access
configurationSlotMifIndex (1)	Integer	The MPEG interface for which these items apply.	N/A
copyStoredSlot(2)	Integer	Setting this copies the specified stored configuration slot to the active slot. Reading this returns the stored configuration slot last copied to the active slot. Valid values to read and set are 1-8. Reading a value of 0 implies that no stored slot has been copied to the active slot.	Get/Set
storeActiveSlot(3)	Integer	Setting this copies the current settings (held in the active slot) to the specified stored configuration slot.	Get/Set
slotCopyTime (4)	Octet string	The time at which a stored configuration was last copied to the active slot, or the active slot was copied to a stored slot. Time in time stamp format.	Get
currentConfigurationSlotName(5)	Octet string	This returns the name of the configuration last copied to the active slot.	Get
clearStoredSlot (6)	Integer	Setting this clears the contents of the specified stored configuration slot. Reading this value has no meaning.	Get/Set

Configuration Slot Name Table

The Configuration Slot Name table contains the name of the configuration stored in each slot, and is defined as:

Variable	Type	Use	Access
configurationSlotNameMifIndex (1)	Integer	The MPEG interface for which these slot names apply.	N/A
configurationSlotNameIndex (2)	Integer	The slot number of interest 1..8.	N/A
configurationSlotName (3)	Octet string	The name of the slot.	Get

Selecting a Configuration Slot

We get a number of support requests asking how to load a slot. The process to select a slot is easy to do but can be hidden in the detail of the configuration tables.

If the available configuration slot names are ‘walked’, the result will be similar to this:

1: configurationSlotName.1.1 (octet string) FreeviewCamb
2: configurationSlotName.1.2 (octet string) SandyHeath 650MHz
3: configurationSlotName.1.8 (octet string) AutoTemplate

To select “SandyHeath 650 MHz”, the value 2 must be written into CopyStoredSlot

1: copyStoredSlot.1 (integer) 2

‘Walking’ the configurationSlotTable shows that the change has been made:

1: copyStoredSlot.1 (integer) 0
2: storeActiveSlot.1 (integer) 0
3: slotCopyTime.1 (octet string) 8D.EB.4D.E7.44.FD.83.07 (hex)
4: currentConfigurationSlotName.1 (octet string) SandyHeath 650MHz
5: clearStoredSlot.1 (integer) 0

If you are using windows and have the cygwin utilities [http://www.net-snmp.org/] this can be done from the command line:

snmpset -v 1 -c public MTM400IPAddress 1.3.6.1.4.1.128.5.1.17.8.1.1.2.1 i 2

Uploading the configuration slots to begin with requires an HTTP post command, which is not so easy from the command line and would require a utility program.

MPEG Parameters Group

Figure 5–11 shows the structure of the MPEG Parameters Group, which manages the Stream, PID, Program and PID Group parameters.

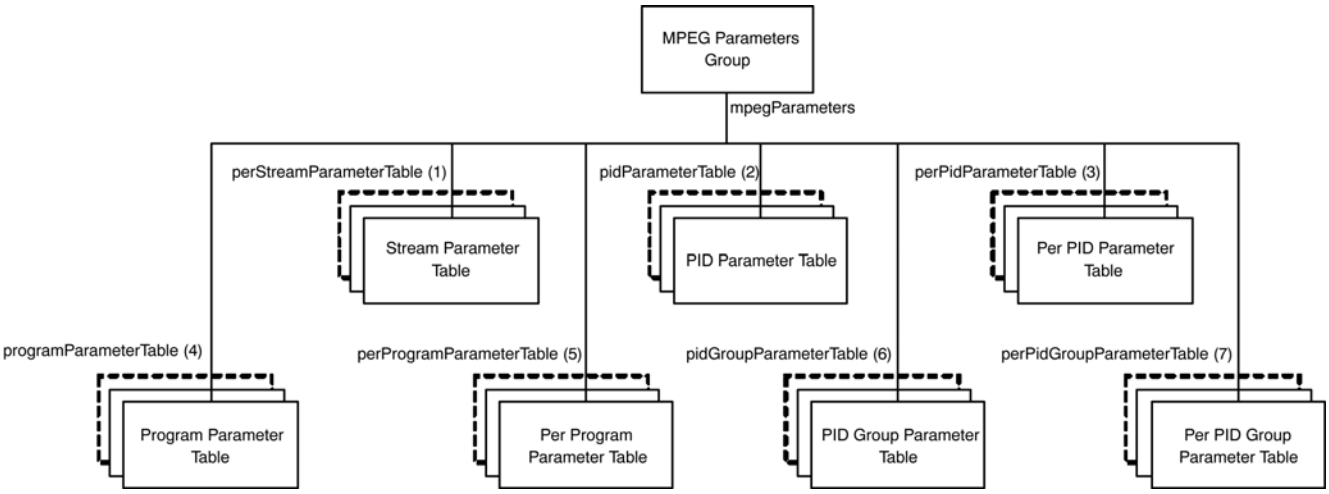


Figure 5–11: MPEG parameters group structure

Stream Parameter Table

The Stream Parameter table provides access to the stream parameters, and is defined as:

Variable	Type	Use	Access
perStreamParameterMifIndex (1)	Integer	The MPEG interface associated with these parameters.	N/A
perStreamParameterIndex (2)	Integer	The unique Id of the required parameter.	N/A
perStreamParameterValue (3)	Octet string	The value of this parameter as a string.	Get/Set

PID Parameter Table

The PID Parameter table provides access to the default PID parameter values, and is defined as:

Variable	Type	Use	Access
pidParameterMifIndex (1)	Integer	The MPEG interface associated with these parameters.	N/A
pidParameterIndex (2)	Integer	The unique Id of the required PID parameter.	N/A
pidParameterDefaultValue (3)	Octet string	The default value of this PID parameter as a string.	Get/Set

PID tests and events will inherit these values by default. Individual PID parameters can be customized using the Per PID Parameter table.

Per PID Parameter Table

The Per PID Parameter table provides access to individual PID parameters, and is defined as:

Variable	Type	Use	Access
perPidParameterMifIndex (1)	Integer	The MPEG interface associated with these PID specific parameters.	N/A
perPidParameterIndex (2)	Integer	The unique Id of the required PID parameter.	N/A
perPidParameterPidIndex (3)	Integer	The PID of interest (Pid Number + 1).	N/A
perPidParameterValue (4)	Octet string	The value of this specific PID parameter as a string.	Get/Set

Program Parameter Table

The Program Parameter table provides access to the default Program parameter values, and is defined as:

Variable	Type	Use	Access
programParameterMifIndex (1)	Integer	The MPEG interface associated with these parameters.	N/A
programParameterIndex (2)	Integer	The unique Id of the required Program parameter.	N/A
programParameterDefaultValue (3)	Octet string	The default value of this Program parameter as a string.	Get/Set

Program tests and events will inherit these values by default. Individual program parameters can be customized using the Per Program Parameter table.

Per Program Parameter Table

The Per Program Parameter table provides access to individual Program parameters, and is defined as:

Variable	Type	Use	Access
perProgramParameterMifIndex (1)	Integer	The MPEG interface associated with these Program specific parameters.	N/A
perProgramParameterIndex (2)	Integer	The unique Id of the required Program parameter.	N/A
perProgramParameterProgramIndex (3)	Integer	The Program (program number +1) of interest.	N/A
perProgramParameterValue (4)	Octet string	The value of this specific Programs parameter as a string.	Get/Set

PID Group Parameter Table

The PID Group Parameter table provides access to the default PID group parameter values, and is defined as:

Variable	Type	Use	Access
pidGroupParameterMifIndex (1)	Integer	The MPEG interface associated with these parameters.	N/A
pidGroupParameterIndex (2)	Integer	The unique Id of the required PID Group parameter.	N/A
pidGroupParameterDefaultValue (3)	Octet string	The default value of this PID Group parameter as a string.	Get/Set

PID Group tests will inherit these values by default. Individual PID group parameters can be customized using the Per PID Group Parameter table.

Per PID Group Parameter Table

The Per PID Group Parameter table provides access to individual PID group parameters, and is defined as:

Variable	Type	Use	Access
perPidGroupParameterMifIndex (1)	Integer	The MPEG interface associated with these PID specific parameters.	N/A
perPidGroupParameterIndex (2)	Integer	The unique Id of the required PID Group parameter.	N/A
perPidGroupParameterPidGroup Index (3)	Integer	The PID Group of interest (Group Number).	N/A
perPidGroupParameterValue (4)	Octet string	The value of this specific PID Groups parameter as a string.	Get/Set

MPEG Record Group

The MPEG Record table provides the control and monitoring interface for the MTM400 triggered recording function, and is defined as follows:

Variable	Type	Use	Access
mpegRecordMifIndex (1)	Integer	The MPEG interface.	N/A
mpegRecordState (2)	Integer	State of recording: 0 = Idle 1 = Waiting for Trigger 2 = Recording in Progress 3 = Recording Complete	Get
mpegRecordTriggerType (3)	Integer	Type of recording trigger: 0 = Immediate (default) 1 = External Rising Edge 2 = External Falling Edge 3 = Event Alarm	Get/Set
mpegRecordLargestAllowed (4)	Integer	Largest number of packets allowed to record.	Get
mpegRecordPreTrigger (5)	Integer	Percentage of stream prepended to the recording before the trigger set off.	Get/Set
mpegRecordActualSize (6)	Integer	Actual size of recording in packets.	Get

Variable	Type	Use	Access
mpegRecordTotalMemorySize (7)	Integer	Total system memory size (in Megabytes) available for recording.	Get
mpegRecordActivate (8)	Integer	Setting this to 1 arms the trigger, setting to 0 aborts the arming, or stops the recording at the current position.	Get/Set
mpegRecordClear (9)	Integer	Setting this clears the recording.	Get/Set
mpegRecordTimestampAvailable (10)	Integer	Specifies whether device is capable of time-stamping recorded packets.	Get
mpegRecordUseTimestamp (11)	Simple Boolean	Specifies whether to time-stamp packets.	Get/Set
mpegRecordProgress (12)	Integer	The percentage of the recording completed.	Get
mpegRecordDesiredSize (13)	Integer	Desired size of recording in packets.	Get/Set
mpegRecordTriggerTime (14)	Octet string	This returns the time at which the trigger for the current recording occurred, or zero if not currently meaningful.	Get

Web Server URLs

Web Server URLs

The following sections define the URLs supported by the MTM400 Web Server. Note that using the Web interface will not enable access to option dependent data, for example schedules or service logging.

Configuration

Upload Configuration

http:://<MTM400 IP Address>/cgi-bin/uploadconfiguration?stream=x&slot=y

The 'stream' parameter is always 1 for MTM400. The slot corresponds to the configuration slot into which the new configuration parameters will be loaded. This is in the range 1…8.

Download Configuration

http:://<MTM400 IP Address>/cgi-bin/
downloadconfiguration?stream=x&slot=y

The 'stream' parameter is always 1 for MTM400. The slot corresponds to the configuration slot from which the parameters will be downloaded. This is in the range 1…8.

Configuration Schema

http:://<MTM400 IP Address>/config.xsd

Editing configuration files can be difficult, however with an advanced XML editor the process can be made easier with an XSD file. This file will allow smart content completion and error checking.

Parameter Definitions

http:://<MTM400 IP Address>/wmsm/configuration/parameters.xml

All the parameters in the configuration files are defined in this XML file.

http:://<MTM400 IP Address>/wmsm/configuration/parametermap.xml

This XML file defines the linkage between each test and the settable parameters.

Schedules

Upload Schedule

http:://<MTM400 IP Address>/cgi-bin/uploadschedule?stream=x

This URL is used to upload a schedule file for the specified stream interface. The 'stream' parameter is always 1 for MTM400.

Download Schedule

http::://<MTM400 IP Address>/cgi-bin/downloadschedule?stream=x

This URL is used to download a schedule file from the specified stream interface. The ‘stream’ parameter is always 1 for MTM400.

Recording

Download Recording

http::://<MTM400 IP Address>/data/recording?start=x&end=y

This URL is used to download a stream recording. The ‘start’ and ‘end’ parameters define the range of packets of interest.

Logging

Download Stream Log

http::://<MTM400 IP Address>/cgi-bin/streamlog?start=x&end=y

This URL is used to download the stream log. The ‘start’ and ‘end’ parameters define the range of log entries of interest. The available range of log entries can be determined from the SNMP MIB table ‘mpegLog’.

The stream log may be very large, so the URL allows for sections of the log to be downloaded. An XML format log file is downloaded in response to the invocation of this URL. There is no acknowledgement required because the log entries are only destroyed by the wrapping of its circular buffer.

Download Device Log

http::://<MTM400 IP Address>/cgi-bin/ devicelog?start=x&end=y

This URL is used to download the device log. The ‘start’ and ‘end’ parameters define the range of log entries of interest. The range of log entries available can be determined from the SNMP MIB table ‘adsysLog’.

An XML format log file is downloaded in response to the invocation of this URL. There is no acknowledgement required because the log entries are only destroyed by the wrapping of its circular buffer.

The Stream and Device Log downloads can also take a language parameter. For example,

http://<MTM400 IP Address>/cgi-bin/devicelog?start=x&end=y&lang=x

where x=24 (English), 52 (Japanese) or 134 (Chinese).

Service Logs

Download Service Log

http::://<Despina IP Address>/service/log?reqid=ID

This URL results in a CSV file of service Log results being downloaded. Each request has a unique user-defined ID that is used in the acknowledge phase.

It must be acknowledged that data has been processed fully before it can be destroyed, so the client must make a request with a matching ‘reqid’ to clear the data as follows:

http::://<Despina IP Address>/service/logack?reqid=ID

The PIDs involved in the service log can be specified in the configuration file or configured by HTTP requests. The following two URLs will introduce or remove a PID from the service log.

http::://<Despina IP Address>/service/addpid?pid=1234
http::://<Despina IP Address>/service/delpid?pid=1234

Use the following URL to return a list of current PIDs in XML format.

http://<Despina IP Address>//service/current.xml

TMCC Information

Download TMCC IIP Information

http::://<MTM400 IP Address>/cgi-bin/tmcciipinfo

This URL is used to download TMCC and IIP information for ISDB-T streams. The page is in the following format:

```
<TMCCIIPInfo>
  <TMCCIdentifier>2</TMCCIdentifier>
  <BufferResetControlFlag>1</BufferResetControlFlag>
  <SwitchOnControlFlag>0</SwitchOnControlFlag>
  <InitialisationTimingHeadPacketFlag>0</InitialisationTimingHeadPacketFlag>
  <FrameHeadPacketFlag>0</FrameHeadPacketFlag>
  <FrameIndicator>1</FrameIndicator>
  <LayerIndicator>8</LayerIndicator>
  <TMCCCountDown>15</TMCCCountDown>
  <ACDataInvalidFlag>1</ACDataInvalidFlag>
  <TSPCounter>4607</TSPCounter>
  <ACData>4294967295</ACData>
  <TMCCSynchronizationWord>0</TMCCSynchronizationWord>
  <IIPCountDown>15</IIPCountDown>
  <Configuration  Name="Current">
      <GuardMode>3</GuardMode>
      <GuardInterval>2</GuardInterval>
      <Layer Number="1">
```

```
            <Segments>1</Segments>
            <TimeInterleaving>2</TimeInterleaving>
            <Modulation>1</Modulation>
            <CodeRatingInnerCode>0</CodeRatingInnerCode>
            <BitRate>0</BitRate>
        </Layer>
        <Layer Number="2">
            … format as layer 1
        </Layer>
        <Layer Number="3">
            … format as layer 1
        </Layer>
    </Configuration>
    <Configuration Name="Next">
        … format as current
    </Configuration>
</TMCCIIPInfo>
```

Templates

Template Results

http::://<MTM400 IP Address>/cgi-bin/templateresults

This page is used to drive the MTM400 template test UI; it consists of a labels section used for localization, followed by a structure showing the expected and actual values of items specified in the template.

```
<Template State="Red">
  <Labels>
    <Template>Template</Template>
    <TransportStreamId>TransportStream ID</TransportStreamId>
    <NetworkId>Network ID</NetworkId>
    <OriginalNetworkId>Original Network ID</OriginalNetworkId>
    <OtherServicesAllowed>Other Services Allowed</OtherServicesAllowed>
    <ServiceList>Services</ServiceList>
    <Service>Service</Service>
    <Constraint>Constraint</Constraint>
    <ServiceType>Service Type</ServiceType>
    <ServiceName>Service Name</ServiceName>
    <PCRPID>PCR PID</PCRPID>
    <OtherPIDsAllowed>Other PIDs Allowed</OtherPIDsAllowed>
    <PIDList>PIDs</PIDList>
    <PID>PID</PID>
    <StreamType>Stream Type</StreamType>
    <CADescriptorPresent>Conditional Access Descriptor Present</CADescriptorPresent>
    <IsScrambled>Is PID Scrambled</IsScrambled>
    <RatingList>Ratings</RatingList>
    <DVBRatingList>DVB Ratings</DVBRatingList>
    <DVBRatingTemplate>DVBRating Country = </DVBRatingTemplate>
    <DVBAcceptableValues>Acceptable Values</DVBAcceptableValues>
    <DVBRatingValue>DVB Rating Value</DVBRatingValue>
```

```
    <ATSCRatingList>ATSC Ratings</ATSCRatingList>
    <ATSCRatingTemplate>ATSC Rating Region = </ATSCRatingTemplate>
    <ATSCAcceptableValues>Acceptable Values</ATSCAcceptableValues>
    <ATSCRatingValue>ATSC Rating Value</ATSCRatingValue>
    <DCIIRatingList>DCII  Ratings</DCIIRatingList>
    <DCIIRatingTemplate>DCII Rating Region = </DCIIRatingTemplate>
    <DCIIAcceptableValues>Acceptable Values</DCIIAcceptableValues>
    <DCIIRatingValue>DCII Rating Value</DCIIRatingValue>
  </Labels>
  <TransportStreamId State="Green" ActualValue="419">419</TransportStreamId>
  <NetworkId State="Green" ActualValue="1220">1220</NetworkId>
  <OriginalNetworkId State="Green" ActualValue="901">901</OriginalNetworkId>
  <OtherServicesAllowed State="Green" ActualValue="0">0</OtherServicesAllowed>
  <ServiceList State="Red">
    <Service Number="4173" State="Red">
      <Constraint State="Red" ActualValue="Present">Present</Constraint>
      <ServiceType State="Green" ActualValue="1">1</ServiceType>
      <ServiceName State="Green" ActualValue="EEE ONE">EEE ONE</ServiceName>
      <PCRPID State="Green" ActualValue="600">600</PCRPID>
      <OtherPIDsAllowed State="Green" ActualValue="0">0</OtherPIDsAllowed>
      <PIDList State="Green">
        <PID Number="600" State="Green">
          <Constraint State="Green" ActualValue="Present">Present</Constraint>
          <StreamType State="Green" ActualValue="2">2</StreamType>
        </PID>
        <PID Number="652" State="Green">
          <Constraint State="Green" ActualValue="Present">Present</Constraint>
          <StreamType State="Green" ActualValue="11">11</StreamType>
        </PID>
      </PIDList>
      <RatingList State="Green">
        <DVBRatingList State="Green">
          <DVBRatingTemplate Country="eng" State="Unknown" ActualValue="">
            <DVBAcceptableValues>
              <DVBRatingValue>0</DVBRatingValue>
            </DVBAcceptableValues>
          </DVBRatingTemplate>
        </DVBRatingList>
      </RatingList>
    </Service>
  </ServiceList>
</Template>
```

Licensing

Upload License Information

http::://<MTM400 IP Address>/cgi-bin/license

For large systems it may be necessary to batch upload many licenses. The license info should be posted to this URL. The license string must be encoded into XML in the following format.

```
<LICENSE>
   <KEY>abcdefgh</KEY>
</LICENSE>
```

Debug Information

Data Logging

http::://<MTM400 IP Address>/info

This page is used internally during testing the output can be logged to give an indication of various items over time.

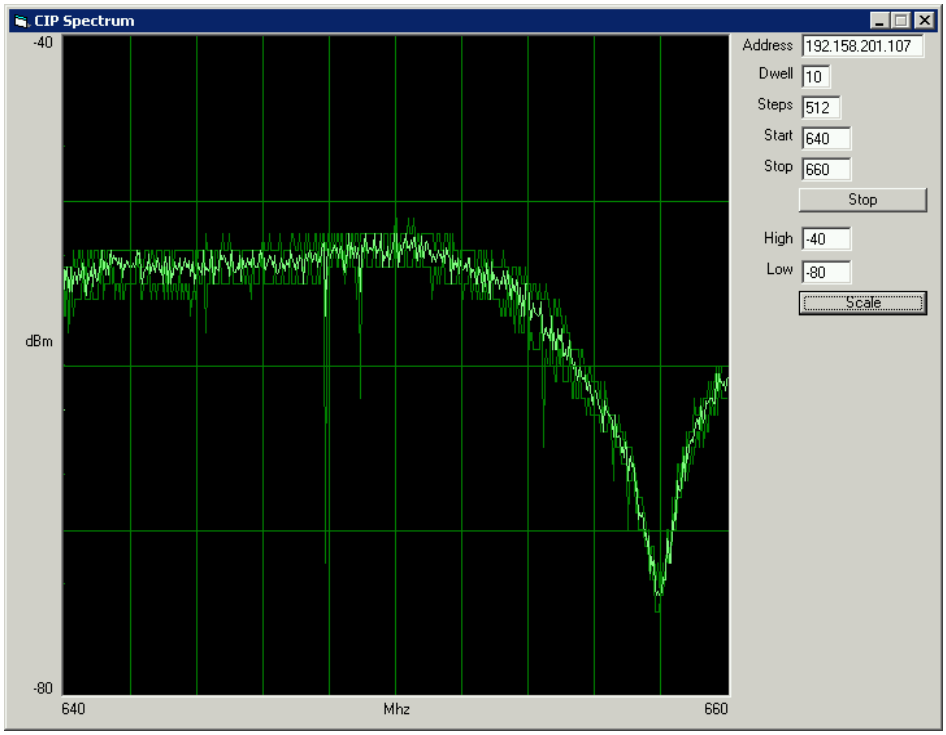
```
<DespinaStatus>
   <swver>2.2.0 Beta 04 BB26</swver>
   <biosver>2.07</biosver>
   <hwver>5</hwver>
   <license>3PNLP-VEWYN-QWLQ4-ZFJBN</license>
   <options>1 2 3 4 5 6 7 8 13 15 16 17 18 </options>
   <MAC>08-00-11-19-72-C6</MAC>
   <IP>192.158.201.107</IP>
   <SZA>SZA free</SZA>
   <SZB>SZB free</SZB>
   <CIPID>CTZZCOFDM Interface::Brd Id 8, Brd Ver 8</CIPID>
   <active>4</active>
   <bitrate>18096298</bitrate>
   <lock>1</lock>
   <TEFs>223</TEFs>
   <MER>26.100000</MER>
   <EVM>3.700000</EVM>
   <signal>-61.000000</signal>
   <BER>0.000000000000</BER>
   <CNR>0.000000000000</CNR>
   <SNR>30.600000000000</SNR>
   <NET_ID>9018</NET_ID>
   <TS_ID>4109</TS_ID>
   <NumPIDs>50</NumPIDs>
   <utctime>1122847438844102</utctime>
   <ticks>19820920</ticks>
   <Stream>1</Stream>
   <tamb>30</tamb>
   <tcpu>37</tcpu>
   <FreeRAM>45559692</FreeRAM>
```

```
  <PeakRAM>11071332</PeakRAM>
  <FPGAoffs>0</FPGAoffs>
  <SNMPRX>26282668</SNMPRX>
  <SNMPTX>26282668</SNMPTX>
  <HTTPRX>10261792</HTTPRX>
  <HTTPTX>172235557</HTTPTX>
  <CURRCODE>15</CURRCODE>
  <LASTCODE>11</LASTCODE>
  <SSTATEMSG>63</SSTATEMSG>
  <PCRMSG>21678263</PCRMSG>
  <SBYTEMSG>40</SBYTEMSG>
  <SECTMSG>37607724</SECTMSG>
  <FREETAB>9316401</FREETAB>
  <PEAKTAB>10485760</PEAKTAB>
  <FREENET>452224</FREENET>
  <PEAKNET>1048576</PEAKNET>
  <RAWQP>29</RAWQP>
  <APPQP>29</APPQP>
  <SYSQP>29</SYSQP>
  <RAWQPMAX>29</RAWQPMAX>
  <APPQPMAX>29</APPQPMAX>
  <SYSQPMAX>29</SYSQPMAX>
  <lastcmd>none</lastcmd>
  <lastval>0.000000</lastval>
</DespinaStatus>
```

A secondary use is to control the RF interface cards during the validation. Most controls are possible, a selection of which is shown below:

http::://<MTM400 IP Address>/info?UUT_FREQ=1234
http::://<MTM400 IP Address>/info?UUT_QAM =64
http::://<MTM400 IP Address>/info?UUT_SYM =20000
http::://<MTM400 IP Address>/info?UUT_J83QAM

These commands can be used for items like scanning channels or even as a rudimentary spectrum analyzer.



Box Identification

http::://<MTM400 IP Address>/idon
http::://<MTM400 IP Address>/idoff

In a large system it can be hard to identify units for service in a rack. This command will cause the network LED on the front panel to flash.

Controls

Remote Reset

http::://<MTM400 IP Address>/cgi-bin/reset?magic=DE5B12A

Table Information

Table Version Information

http::://<MTM400 IP Address>/cgi-bin/tableserialnumbers

```
<TableSerialNumbers>
  <Tables Update="Incremental">
  <Table ID="0" SerialNumber="21" />
  <Table ID="1" SerialNumber="8" />
  <Table ID="2" SerialNumber="455" />
  <Table ID="3" SerialNumber="7" />
  </Tables>
</TableSerialNumbers>
```

This information shows all of the versions found by the MTM400, and does not relate directly to the table version numbers. This information can be monitored to detect a change in version or new data.

Available Table Information

http::://<MTM400 IP Address>/cgi-bin/availabletables

```
<Tables>
  <Table TableId="0">1</Table>
  <Table TableId="2">6</Table>
</Tables>
```

This information shows the tables that the MTM400 is holding that can be downloaded in binary format.

Available Sub-Table Information

http::://<MTM400 IP Address>/cgi-bin/availablesubtables?tableid=2

```
<TableInfo TableId="2" TableName="PMT">
  <SubtableIdKeys Update="Incremental">
      <SubtableId Key="1">table_id</SubtableId>
      <SubtableId Key="2">program_number</SubtableId>
  </SubtableIdKeys>
  <Subtables>
      <Subtable PID="4173" ID="2.4173" DespinaSubtableID="2.4173"
                          LastCompleteSerialNumber="3" VersionNumber="21"
                          CurrentSectionCount="1"/>
      <Subtable PID="4237" ID="2.4237" DespinaSubtableID="2.4237"
                          LastCompleteSerialNumber="3" VersionNumber="31"
                          CurrentSectionCount="1"/>
  </Subtables>
</TableInfo>
```

For each table type, this information is used to break down each available instance of the required table. In the example above, there are two PMT tables

that can be downloaded. The ID is passed to the subtabledata URL to specify the required data. The subtable data is returned in binary format.

http:://<MTM400 IP Address>/cgi-bin/ subtabledata?tableext=2.4173

PCR/PTS Information

Download xxxx IIP Information

http:://<MTM400 IP Address>/cgi-bin/pcrvalues?stream=1&pid=y
http:://<MTM400 IP Address>/cgi-bin/pcrvalues?stream=1&pid=y&slow=1

There are two variants of this command: full speed and slow. If you specify slow=1, it allows one second between samples (parameter PS178). For a slow−changing value such as drift, this may give better data.

Some parameters change according to the interface in use; attributes like PP11="PP51" indicate where a substitution should be made.

The URL returns a number of samples covering all PCR measurements. The attributes AC_V, OJ_V, DR_V and FO_V denote whether the figure provided is reliable (the sample was not taken during a settling period).

```
<PCRValues Stream="1" PID="600" UTCOffset="60"
        PP2="PP2" PP4="PP50" PP11="PP51" PP12="PP52" PP13="PP53" >
  <PP2>40</PP2>
  <PP4>500</PP4>
  <PP11>25000</PP11>
  <PP12>800</PP12>
  <PP13>350</PP13>
  <PCRResults>
      <PCRResult UTCTime="1122851331359159"
                  AC_V="1" OJ_V="1" DR_V="1" FO_V="1" >
          <Accuracy>-8</Accuracy>
          <Jitter>-1456</Jitter>
          <Drift>45</Drift>
          <FrequencyOffset>-15</FrequencyOffset>
          <Interval>8811</Interval>
      </PCRResult>
      <PCRResult UTCTime="1122851331388414"
                  AC_V="1" OJ_V="1" DR_V="1" FO_V="1" >
          <Accuracy>-16</Accuracy>
          <Jitter>-887</Jitter>
          <Drift>45</Drift>
          <FrequencyOffset>-15</FrequencyOffset>
          <Interval>29254</Interval>
      </PCRResult>
  </PCRResults>
</PCRValues>
```

http:://<MTM400 IP Address>/cgi-bin/ptsvalues?stream=1&pid=600

Using PTS values are much simpler than using PCR values, but only returns a timestamp each time a PTS is received.

```
<PTSValues Stream="1" PID="600" UTCOffset="60">
  <PP5>700</PP5>
  <PTSResults>
      <PTSResult UTCTime="1122851546098"></PTSResult>
      <PTSResult UTCTime="1122851546038"></PTSResult>
  </PTSResults>
</PTSValues>
```

Repetition Information

Download Section Repetition Information

http:://IP/cgi-bin/sectionreptninterval?stream=x&pid=p&tableext=a.b

See "Available Sub-Table Information" (see page 6–9) for details about which table extension to use. The output is a simple XML file showing the intervals with accurate timestamps.

Using the details for the PMT above:

http://IP/cgi-bin/sectionreptninterval?stream=1&pid=600&tableext=2.4173

results in this data:

```
<SectionRepetitionIntervals UTCOffset="60" Parameter="PS9" Limit="500">
    <Interval UTCTime="1122883815632289">188</Interval>
    <Interval UTCTime="1122883815823445">191</Interval>
    <Interval UTCTime="1122883816011024">188</Interval>
    ... more ...
</SectionRepetitionIntervals>
```

Tables that may be graphed in this way are, standards permitting:

PAT, PMT, ACTUAL_NIT, ACTUAL_SDT, TOT, TDT, MGT, STT, TERRESTRIAL_VCT, CABLE_VCT, RRT, ATSCEIT, DCCSCT.

Download Section Repetition Information

http:://<IPAddr>/cgi-bin/intersectiongap?stream=x&pid=p&tableext=a.b

Using details for the actual NIT:

http://IP/cgi-bin/intersectiongap?stream=1&pid=16&tableext=64.12290

results in the data:

```
<InterSectionGaps UTCOffset="60" Parameter="PS14" Limit="25">
    <Interval UTCTime="1122883737581832">9984</Interval>
    <Interval UTCTime="1122883747581355">9985</Interval>
    <Interval UTCTime="1122883757583120">9987</Interval>
```

... more ...
</InterSectionGaps>

Tables that may be graphed in this way are, standards permitting:

ACTUAL_NIT, OTHER_NIT, ACTUAL_SDT, OTHER_SDT, BAT, ACTUAL_EITPF, OTHER_EITPF, ACTUAL_EITS, OTHER_EITS, RST, TDT, TOT, SDTT, BIT, CDT.

Download Sub-Table Repetition Information

http::://<IP>/cgi-bin/subtablereptninterval?stream=x&tableext=a.b

Using details for the actual NIT:

http://IP/cgi-bin/subtablereptninterval?stream=1&tableext=64.12290

results in the data.

<SubtableRepetitionIntervals UTCOffset="60" Parameter="PS15" Limit="10000">
<Interval UTCTime="1122884577571409">9999</Interval>
<Interval UTCTime="1122884587570348">9999</Interval>
<Interval UTCTime="1122884597670683">10100</Interval>
... more ...
</SubtableRepetitionIntervals>

Tables that may be graphed in this way are, standards permitting:

ACTUAL_NIT, OTHER_NIT, ACTUAL_SDT, OTHER_SDT, BAT, ACTUAL_EITPF, OTHER_EITPF, LAST_ACTUAL_EITS, LAST_OTHER_EITS, TDT, TOT, DPI, TERRESTRIAL_VCT, CABLE_VCT, ATSCEIT, DCCT, SDTT, BIT, CAT, CDT.

Download Section Repetition Information

http::://<IPAddr>/cgi-bin/cycgroupreptninterval?stream=x&tableext=a.b

This is for ISDB-T mode only; it is to graph the particular variant of EITs in use in Japan.

RF Card Information

There are 2 classes of RF interface card in use for the MTM400, known as SZ or CIP cards. SZ are the older class; these are accessed solely using SNMP. The newer CIP cards have an HTTP interface.

SZ Cards are LBand, QAMA, QAMB and QAMC.

CIP Cards are 8PSK, QAMB2, 8VSB, COFDM and GbE.

This information only applies to CIP cards.

Download Card Information

http::://<MTM400 IP Address>/cgi-bin/genericcarddetails?stream=1&card=0

This returns an XML page that describes the card in use. There are indicators (read-only), parameters (read/write), enumerations and graphs.

For indicators and Parameters

- Name - the name of the parameter
- High - the maximum value of the parameter
- Low - the minimum value of the parameter
- Nominal - the nominal value of the parameter
- Resolution - the resolution of the parameter
- Scale - the numerical exponent of the parameter (usually 0)
- Units - the units associated with the parameter
- Image - the image associated with the parameter - text, switch or menu
- EnumerationID - the enumeration id of the indicator
- CmdString - this is the short string “TLA” that is used as the command passed over the link to the interface card
- PollIndex - the index that identifies the indicator value in the Poll Query
- MetricID - This number is used to inform the RF testing code which item corresponds to tested metrics

And for graphs

- Type - the type of the graph. Valid values are B, S, H, T, which represent Bitmap, Scatter, Histogram or Trend graphs
- XTitle - the title of the x axis
- YTitle - the title of the y axis

- XMin - the minimum X value to be drawn on the left of the scale
- YMin - the minimum Y value to be drawn on the bottom of the scale
- XMax - the maximum X value to be drawn on the right of the scale
- YMax - the maximum Y value to be drawn on the top of the scale
- XGrid - the number of vertical gridlines (not including out side border). For bitmap graphs this is the width of the display
- YGrid - the number of horizontal gridlines (not including out side border) For bitmap graphs this is the height of the display

```
-- Standard xml version info --
<?xml version="1.0" encoding="UTF-8"?>
-- Card selected, number or 'none' --
<CommonIFCards Selected="0" >
-- Card description --
  <CommonIFCard ID="0"  MajorVersion="Z"  MinorVersion="Z"
            Description="COFDM Interface::Brd Id 8, Brd Ver 8">
-- Selected input number or 'none' --
    <Inputs Selected="0" >
-- Input description, passthrough=false means the controlled input --
      <Input ID="0" Available="true" PassThrough="false" Description="RF">
-- Enumerations used for menus or to return textual results --
        <EnumerationDescriptions>
          <EnumerationDescription  ID="1">QPSK,16QAM,64QAM
          </EnumerationDescription>
        </EnumerationDescriptions>
-- Parameters are things the user can change. --
        <ParameterDescriptions>
          <ParameterDescription ID="0">
            <Name>Nominal tuner frequency</Name>
-- This is the TLA it is the key index --
            <CMDString>FRQ</CMDString>
            <High>861000</High>
            <Low>49000</Low>
            <Nominal>546000</Nominal>
            <Resolution>1</Resolution>
            <Scale>0</Scale>
            <Units>KHz</Units>
-- A numeric setting --
            <Image>text</Image>
            <EnumerationID>0</EnumerationID>
-- All items are returned in a single poll, this is this items position --
            <PollIndex>1</PollIndex>
            <MetricID>0</MetricID>
          </ParameterDescription>
        </ParameterDescriptions>
        <IndicatorDescriptions>
          <IndicatorDescription ID="0">
            <Name>Overall RF lock</Name>
```

```
            <CMDString>LOK</CMDString>
            <High>1</High>
            <Low>0</Low>
            <Nominal>0</Nominal>
            <Resolution>1</Resolution>
            <Scale>0</Scale>
            <Units> </Units>
-- A Boolean indication --
            <Image>switch</Image>
            <EnumerationID>0</EnumerationID>
            <PollIndex>2</PollIndex>
-- This is a cross reference to a standard metric --
            <MetricID>1</MetricID>
          </IndicatorDescription>
          <IndicatorDescription ID="1">
            <Name>Encoding Format</Name>
            <CMDString>AEF</CMDString>
            <High>2</High>
            <Low>0</Low>
            <Nominal>0</Nominal>
            <Resolution>1</Resolution>
            <Scale>0</Scale>
            <Units> </Units>
-- A drop down menu --
            <Image>menu</Image>
-- Using QPSK, QAM16 and QAM64 from the enumerations --
            <EnumerationID>1</EnumerationID>
            <PollIndex>3</PollIndex>
-- Metric ID=0 means an item specific to this card --
            <MetricID>0</MetricID>
          </IndicatorDescription>
          <IndicatorDescription ID="2">
            <Name>Input level</Name>
            <CMDString>INP</CMDString>
            <High>-2</High>
            <Low>-88</Low>
            <Nominal>-2</Nominal>
            <Resolution>1</Resolution>
            <Scale>0</Scale>
            <Units>dBm</Units>
            <Image>text</Image>
            <EnumerationID>0</EnumerationID>
            <PollIndex>4</PollIndex>
-- This is a cross reference to a standard metric --
            <MetricID>7</MetricID>
          </IndicatorDescription>
        </IndicatorDescriptions>
        <Graphs>
          <Graph ID="0">
-- A scatter graph, trend and sweep are also possible --
            <Type>S</Type>
            <Name>RF constellation (data carriers)</Name>
            <CMDString>CST</CMDString>
```

```
            <XTitle>I</XTitle>
            <YTitle>Q</YTitle>
            <XMin>-128</XMin>
            <YMin>-128</YMin>
            <XMax>+127</XMax>
            <YMax>+127</YMax>
            <XGrid>-60</XGrid>
            <YGrid>-60</YGrid>
          </Graph>
        </Graphs>
      </Input>
-- Input description, passthrough=true means the ASi passthrough. --
      <Input ID="1" Available="true" PassThrough="true" Description="ASI">
-- The input data for RF is repeated to ease programming. --
      </Input>
    </Inputs>
  </CommonIFCard>
</CommonIFCards>
```

Download Status Information

http::://<MTM400 IP Address>/cgi-bin/pollstatus?stream=1&card=0

The status of each parameter and indicator can be read back from this URL. The attributes indicate whether scaling should be applied, or whether the value is invalid (out of range or pending setting).

```
<?xml version="1.0" encoding="UTF-8"?>
<CommonIFPollValues Card="0" unscaled="1">
  <IndicatorValue Valid="1" TLA="AEF" Index="5"
        OutOfRange="0" Units=" " Scale="1e0" >1</IndicatorValue>
  <IndicatorValue Valid="1" TLA="INP" Index="12"
        OutOfRange="0" Units="dBm" Scale="1e0" >-62</IndicatorValue>
  <IndicatorValue Valid="1" TLA="LOK" Index="0"
        OutOfRange="0" Units=" " Scale="1e0" >1</IndicatorValue>
  <ParameterValue Valid="1" TLA="FRQ" Index="0" Pending="0" >641833</ParameterValue>
</CommonIFPollValues>
```

A simplified version of the above is available when RF Testing has been enabled. Those indicators that have a metric ID set are used to drive these values.

http::://<MTM400 IP Address>/cgi-bin/metrics

```
<Metrics>
  <LOCK>1.00000</LOCK>
  <MER>25.8000</MER>
  <MER_DRIFT>0.626917</MER_DRIFT>
  <EVM>3.90000</EVM>
  <EVM_DRIFT>-0.372383</EVM_DRIFT>
  <BER_PRS scale="1e-9">0.000000</BER_PRS>
  <BER_PRS_DRIFT scale="1e-9">0.000000</BER_PRS_DRIFT>
  <BER_PVI scale="1e-9">259000</BER_PVI>
  <BER_PVI_DRIFT scale="1e-9">-45395.8</BER_PVI_DRIFT>
```

```
  <POWER>-62.0000</POWER>
  <POWER_DRIFT>0.786610</POWER_DRIFT>
  <SNR>29.8000</SNR>
  <SNR_DRIFT>-0.455364</SNR_DRIFT>
  <CAR_OFS>8.81300</CAR_OFS>
  <CAR_OFS_DRIFT>-0.000000</CAR_OFS_DRIFT>
</Metrics>
```

Download Graph Information

http://<MTM400 IP Address>/cgi-bin/genericcardgraphdata?stream=1&card=0&input=0&graphindex=S&command=TLA&sequence=0x

The TLA comes from the graph description, and the sequence number is used to avoid receiving repeated data. The client should send the last sequence number received; only new information will then be returned. Sequence = 0 will cause a full set of data to be returned

So this command:

http://IP/cgi-bin/genericcardgraphdata?stream=1&card=0&input=0&graphindex=S&command=CST&sequence=0

retrieved this information:

```
<?xml version="1.0" encoding="UTF-8"?>
<CommonIFCardGraphData Card="0" Input="0" Type="S" Rescale="1" Sequence="1437" >
  <IQSamples vtype="UINT8" DataSize="256" Type="S" >
158,99,96,33,36,219,32,213,160,99,38,151,102,159,102,41,94,217,216,221,150,215,36,217,42,2
23,160,33,94,165,162,221,100,29,164,89,218,99,40,157,92,99,220,97,34,41,220,101,40,35,156,1
63,44,97,36,215,100,155,154,39,222,97,44,31,160,223,222,159,38,99,162,155,... more ...
161,218,149,32,217,218,161,158,99
  </IQSamples>
</CommonIFCardGraphData>
```

Updating Parameters

http::://<MTM400 IP Address>/cgi-bin/ updateprm?stream=1&card=0&input=0&prm=TLA&value=y

The TLA is taken from the card description commands; the value must be in range. The MTM400 will respond with a success or failure message.

For example, this command would set the frequency of the card in the examples above to 641.833 MHz.

http::://<MTM400 IP Address>/cgi-bin/ updateprm?stream=1&card=0&input=0&prm=FRQ&value=641833