THE IBM 8270/8272 EMBEDDED RMON FEATURE

Document Number TR 29.2256

Switch Technology Development
International Business Machines Corporation
Research Triangle Park, North Carolina

ABSTRACT

The purpose of this technical report is to provide the reader with an understanding of the Embedded Remote Monitoring (RMON) Feature of the IBM 8270 LAN Switch and IBM 8272 LAN Switch. This feature provides RMON support for Token Ring as defined by RFC1757 and RFC1513. Following a brief overview of RFC1757 and RFC1513, the capabilities and use of each supported RMON group is discussed in terms of its actual implementation in the 8270/8272. The specific SNMP steps needed to create, modify and delete a group entry are included in the discussion of each supported RMON group.

ITIRC KEYWORDS

- 8270
- 8272
- RMON
- SNMP
- LAN SWITCH

CONTENTS

THE IBM 8270/8272 EMBEDDED RMON FEATURE

INTRODUCTION

Remote Monitoring (RMON) support for Token Ring is defined by RFC1757 and RFC1513. RFC1757 (the RMON MIB) contains the base RMON groups while RFC1513 (the Token Ring RMON MIB) contains the Token Ring specific tables and extensions to RFC1757. The Embedded RMON Feature of the IBM 8270 LAN Switch and 8272 LAN Switch will provide support for the statistics, history control, history, alarm and event groups. Figure 1 lists the groups contained in RFC1757 and RFC1513; and indicates which groups will be supported by the Embedded RMON Feature of the 8270 and 8272.

Figure 1. Token Ring RMON Groups

Group	*Supported*	*Standard*
token ring statistics	**Yes**	RFC1513
history control	**Yes**	RFC1757
token ring history	**Yes**	RFC1513
alarm	**Yes**	RFC1757
host	No	RFC1757
hostTopN	No	RFC1757
matrix	No	RFC1757
filter	No	RFC1757
packet capture	No	RFC1757
event	**Yes**	RFC1757
token ring station	No	RFC1513
token ring station order	No	RFC1513
token ring station config	No	RFC1513
token ring source routing	No	RFC1513

The remainder of this section provides an overview of each of the groups as described in RFC1757 and RFC1513.

The Token Ring Statistics Group

The Token Ring statistics group contain current utilization and error statistics. The statistics are broken down into two groups, the Token Ring MAC-Layer Statistics Group and the Token Ring Promiscuous Statistics Group. The Token Ring MAC-Layer Statistics Group collects information from MAC Layer, including error reports for the ring and ring utilization of the MAC Layer. The Token Ring Promiscuous Statistics Group collects utilization statistics from data packets collected promiscuously.

The History Control Group

The history control group controls the periodic statistical sampling of data from various types of networks. This group consists of the historyControlTable.

The Token Ring History Group

The Token Ring History Group contain historical utilization and error statistics. The statistics are broken down into two groups, the Token Ring MAC-Layer History Group and the Token Ring Promiscuous History Group. The Token Ring MAC-Layer History Group collects information from MAC Layer, including error reports for the ring and ring utilization of the MAC Layer. The Token Ring Promiscuous History Group collects utilization statistics from data packets collected promiscuously.

The Alarm Group

The alarm group periodically takes statistical samples from variables in the probe and compares them to previously configured thresholds. If the monitored variable crosses a threshold, an event is generated. A hysteresis mechanism is implemented to limit the generation of alarms. This group consists of the alarmTable and requires the implementation of the event group.

The Host Group

The host group contains statistics associated with each host discovered on the network. This group discovers hosts on the network by keeping a list of source and destination MAC Addresses seen in good packets promiscuously received from the network. This group consists of the hostControlTable, the hostTable, and the hostTimeTable.

The HostTopN Group

The hostTopN group is used to prepare reports that describe the hosts that top a list ordered by one of their statistics. The available statistics are samples of one of their base statistics over an interval specified by the management station. Thus, these statistics are rate based. The management station also selects how many such hosts are reported. This group consists of the hostTopNControlTable and the hostTopNTable, and requires the implementation of the host group.

The Matrix Group

The matrix group stores statistics for conversations between sets of two addresses. As the device detects a new conversation, it creates a new entry in its tables. This group consists of the matrixControlTable, the matrixSDTable and the matrixDSTable.

The Filter Group

The filter group allows packets to be matched by a filter equation. These matched packets form a data stream that may be captured or may generate events. This group consists of the filterTable and the channelTable.

The Packet Capture Group

The Packet Capture group allows packets to be captured after they flow through a channel. This group consists of the bufferControlTable and the captureBufferTable, and requires the implementation of the filter group.

The Event Group

The event group controls the generation and notification of events from this device. This group consists of the eventTable and the logTable.

The Token Ring Ring Station Group

The Token Ring Ring Station Group contains statistics and status information associated with each Token Ring station on the local ring. In addition, this group provides status information for each ring being monitored.

The Token Ring Ring Station Order Group

The Token Ring Ring Station Order Group provides the order of the stations on monitored rings.

The Token Ring Ring Station Config Group

The Token Ring Ring Station Config Group manages token ring stations through active means. Any station on a monitored ring may be removed or have configuration information downloaded from it.

The Token Ring Source Routing Group

The Token Ring Source Routing Group contains utilization statistics derived from source routing information optionally present in token ring packets.

THE RMON MIBS (RFC1757 AND RFC1513)

The following table provides an overview of the support which will be provided by the 8270 and 8272 Embedded RMON Feature.

Figure 2. Supported RMON Groups, Tables and Traps

Group	*Table*	*Anchor Point*	*Note*
statistics		rmon 1	
	tokenRingMLStatsTable	statistics 2	Current TR MAC statistics.
	tokenRingPStatsTable	statistics 3	Current TR promiscuous statistics.
history		rmon 2	
	historyControlTable	history 1	Controls collection of historical TR statistics.
	tokenRingMLHistoryTable	history 3	Historical TR MAC statistics.
	tokenRingPHistoryTable	history 4	Historical TR promiscuous statistics.
alarm		rmon 3	
	alarmTable	alarm 1	Controls monitoring of TR counters. Actions for rising and failing conditions are specified in the eventTable.
event		rmon 9	
	eventTable	event 1	Controls sending traps and creating logTable entries for rising and falling alarm conditions.
	logTable	event 2	Log Entries for rising and falling alarm conditions.
traps			
	risingAlarm	rmon ENTERPRISE 1	Rising Alarm Trap for monitored TR counters.
	fallingAlarm	rmon ENTERPRISE 2	Falling Alarm Trap for monitored TR counters.

The tokenRingMLStatsTable

The tokenRingMLStatsTable contains the current values of MAC statistics for Token Ring ports. There will be one entry in the table for each 8270/8272 Token Ring port. The table is supported with read-only access. SET is not supported as the data is always present and cannot be created or deleted. Collection of data for this table has no adverse effect on the performance of the switch.

The following two tables illustrate the name, data types, access and short description of the objects in tokenRingMLStatsTable.

Figure 3. tokenRingMLStatsTable (1 of 2)

Object	*Type*	*Access*	*Description*
tokenRingMLStatsIndex	INTEGER	RO	Table Index. Range 1 to 65535.
tokenRingMLStatsDataSource	OID	*RW	ifIndex of the port. Set of this object is not supported.
tokenRingMLStatsDropEvents	Counter	RO	Packets dropped by the probe.
tokenRingMLStatsMacOctets	Counter	RO	Not Implemented. Always zero.
tokenRingStatsMacPkts	Counter	RO	Not Implemented. Always zero.
tokenRingMLStatsRingPurgeEvents	Counter	RO	Number of ring purge events detected by the port.
tokenRingMLStatsRingPurgePkts	Counter	RO	Ring purge MAC packets detected by the port.
tokenRingMLStatsBeaconEvents	Counter	RO	Number of beacon events detected by the port.
tokenRingMLStatsBeaconTime	TimeTicks	RO	Length of time in beacon state by the port.
tokenRingMLStatsBeaconPkts	Counter	RO	Beacon MAC packets detected by the port.
tokenRingMLStatsClaimTokenEvents	Counter	RO	Number of claim tokens events detected by the port.
tokenRingMLStatsClaimTokenPkts	Counter	RO	Claim token MAC packets detected by the port.

Figure 4. tokenRingMLStatsTable (2 of 2)

Object	*Type*	*Access*	*Description*
tokenRingMLStatsNAUNChanges	Counter	RO	Number of NAUN changes detected by the port.
tokenRingMLStatsLineErrors	Counter	RO	Number of line errors detected by the port.
tokenRingMLStatsInternalErrors	Counter	RO	Number of internal errors detected by the port.
tokenRingMLStatsBurstErrors	Counter	RO	Number of burst errors detected by the port.
tokenRingMLStatsACErrors	Counter	RO	Number of address copy errors detected by the port.
tokenRingMLStatsAbortErrors	Counter	RO	Number of abort errors detected by the port.
tokenRingMLStatsLostFrameErrors	Counter	RO	Number of lost frame errors detected by the port.
tokenRingMLStatsCongestionErrors	Counter	RO	Number of congestion errors detected by the port.
tokenRingMLStatsFrameCopiedErrors	Counter	RO	Number of frame copy errors detected by the port.
tokenRingMLStatsFrequencyErrors	Counter	RO	Number of frequency errors detected by the port.
tokenRingMLStatsTokenErrors	Counter	RO	Number of token errors detected by the port.
tokenRingMLStatsSoftErrorReports	Counter	RO	Number of soft errors detected by the port.
tokenRingMLStatsRingPollEvents	Counter	RO	Number of ring poll errors detected by the port.
tokenRingMLStatsOwner	String	*RW	The port is always owned by the switch. Set of this object is not supported.
tokenRingMLStatsStatus	INTEGER	*RW	The value of this object is always 1 (valid entry). Set of this object is not supported.

The tokenRingPStatsTable

The tokenRingPStatsTable contains the current values of promiscuous statistics for Token Ring ports. There will be one entry in the table for each 8270/8272 Token Ring port. The table is supported with read-only access. SET is not supported as the data is always present and cannot be created or deleted. Collection of data for this table has no adverse effect on the performance of the switch.

The following two tables illustrate the name, data types, access and short description of the objects in tokenRingPStatsTable.

Figure 5. tokenRingPStatsTable (1 of 2)

Object	*Type*	*Access*	*Description*
tokenRingPStatsIndex	INTEGER	RO	Table Index. Range 1 to 65535.
tokenRingPStatsDataSource	OID	*RW	ifIndex of the port. Set of this object is not supported.
tokenRingPStatsDropEvents	Counter	RO	Packets dropped by the probe.
tokenRingPStatsDataOctets	Counter	RO	Octets received by the port.
tokenRingPStatsDataPkts	Counter	RO	Packets received by the port.
tokenRingPStatsDataBroadcastPkts	Counter	RO	Broadcast packets received by the port.
tokenRingPStatsDataMulticastPkts	Counter	RO	Multicast packets received by the port.

Figure 6. tokenRingPStatsTable (1 of 2)

tokenRingPStatsDataPkts18to63Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts64to127Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts128to255Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts256to511Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts512to1023Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts1024to2047Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts2048to4095Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts4096to8191Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPkts8192to18000Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsDataPktsGreaterThan18000Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPStatsOwner	String	*RW	The port is always owned by the switch. Set of this object is not supported.
tokenRingPStatsStatus	INTEGER	*RW	The value of this object is always 1 (valid entry). Set of this object is not supported.

The historyControlTable

The historyControlTable contains the current values which control the collection of Token Ring port history statistics, i.e. entries in the tokenRingMLHistoryTable and tokenRingPHistoryTable. When the 8270/8272 boots, this table will contain zero entries. Previously created entries are NOT saved between boots. Entries may be created by the customer until the switch's maximum entries for the table have been reached or the switch's pool of tokenRingHistory records has been exhausted in response to historyControlBucketsRequested requests. Creation of a valid entry will cause the switch to create tokenRingMLHistoryTable and tokenRingPHistoryTable entries for the specified port at the specified time interval. Deletion of an entry will cause the switch to delete all associated tokenRingMLHistoryTable and tokenRingPHistoryTable entries and return the historyControlBucketsGranted to the switch's pool of tokenRingHistory records.

The following table illustrates the name, data types, access and short description of the objects in historyControlTable.

Figure 7. historyControlTable

Object	*Type*	*Access*	*Description*
historyControlIndex	INTEGER	RO	Table Index. Range 1 to 65535.
historyControlDataSource	OID	RW	ifIndex of the port.
historyControlBucketsRequested	INTEGER	RW	Requested maximum number of history records to collect for the port. Range 1 to 65535.
historyControlBucketsGranted	INTEGER	RW	Actual maximum number of history records which will be collected for the port. Range 1 to 65535.
historyControlInterval	INTEGER	RW	The interval time size in seconds. Range 1 to 3600.
historyControlOwner	INTEGER	RW	Customer specified history control owner.
historyControlStatus	INTEGER	RW	The current status of the entry. 1=valid. 2=create request (create an entry). 3=under creation. 4=invalid (delete the entry).

Creating a historyControlTable Entry: An entry in the historyControlTable may be created by issuing an SNMP SET of the historyControlStatus object with a non-existent index and a value of 2 ("create request").

```
Example Parameters

IP Address...9.67.219.31 <------ Switch IP Address
Community....private <---------- read-write Community Name
Object.......historyControlStatus
Index........1 <---------------- non-existent historyControlIndex
Value........2 <---------------- create request
```

Optionally the SET PDU may also contain customer specified values for the following read-write objects:

- historyControlDataSource
- historyControlBucketsRequested
- historyControlBucketsInterval
- historyControlBucketsOwner

Upon successful completion of the SET, a new entry will be created in the historyControlTable with a historyControlStatus of "under creation" (3) and switch defaults for any unspecified optional parameters.

Modifying a historyControlTable Entry: While an entry has a historyControlStatus of "under creation" (3), the following read-write objects may be modified:

- historyControlDataSource
- historyControlBucketsRequested
- historyControlBucketsInterval
- historyControlBucketsOwner
- historyControlStatus

After the historyControlStatus of an entry has been SET to "valid" (1) by the manager, the switch will begin creating tokenRingMLHistoryTable and tokenRingPHistoryTable entries for the port. While an entry has a historyControlStatus of "valid" (1), the read-write objects of this table have the following modification properties:

- May NOT Be Modified
    1. historyControlDataSource
    2. historyControlBucketsInterval
- May Be Modified
    1. historyControlBucketsRequested
    2. historyControlBucketsOwner
    3. historyControlStatus

Deleting a historyControlTable Entry: An entry in the historyControlTable may be deleted by issuing an SNMP SET of the historyControlStatus object with a valid index and a value of 4 ("invalid").

```
Example Parameters

IP Address...9.67.219.31 <------ Switch IP Address
Community....private <---------- read-write Community Name
Object.......historyControlStatus
Index........1 <---------------- existent historyControlIndex
Value........4 <---------------- invalid
```

Deletion of an entry will cause the switch to delete all associated tokenRingMLHistoryTable and tokenRingPHistoryTable entries; and return the historyControlBucketsGranted to the switch's pool of tokenRingHistory records.

Deriving the Switch's historyControlTable Limits: The switch has a limit on the number of historyControlTable entries which may be created and the number of tokenRingMLHistoryTable/tokenRingPHistoryTable entries which may be created. An object containing the value of these limits is not provided by RFC1757 but these limits can be derived.

To derive the switch's limit on the number of historyControlTable entries which may be created, simply create historyControlTable entries until the switch rejects an additional "create request". The number of entries which were created prior to the rejection is the switch's limit on number of historyControlTable entries.

To derive the switch limit on the number of tokenRingMLHistoryTable/tokenRingPHistoryTable entries which may created, simply create one historyControlTable entry and modified the value of the entry's historyControlBucketsRequested object to 65535. After completion of the SET, the entry's historyControlBucketsGranted will contain the switch's limit on the number of tokenRingMLHistoryTable/tokenRingPHistoryTable entries which may be created.

historyControlTable OIDs: The following object identifiers (OIDs) may be helpful in natively manipulating the historyControlTable where "n" is the historyControlIndex.

Figure 8. historyControlTable OIDs

Object	*OID*
historyControlIndex	1.3.6.1.2.1.16.2.1.1.1.n
historyControlDataSource	1.3.6.1.2.1.16.2.1.1.2.n
historyControlBucketsRequested	1.3.6.1.2.1.16.2.1.1.3.n
historyControlBucketsGranted	1.3.6.1.2.1.16.2.1.1.4.n
historyControlBucketsInterval	1.3.6.1.2.1.16.2.1.1.5.n
historyControlBucketsOwner	1.3.6.1.2.1.16.2.1.1.6.n
historyControlBucketsStatus	1.3.6.1.2.1.16.2.1.1.6.n

The OID for the ifIndex object is 1.3.6.1.2.2.1.1.n where "n" is the value of the port's ifIndex. This is used when specifying the historyControlDataSource.

The tokenRingMLHistoryTable

The tokenRingMLHistoryTable contains the historical values of MAC statistics for Token Ring ports. The creation of tokenRingMLHistoryTable entries is controlled by the entries in historyControlTable. For each "valid" entry in the historyControlTable, the switch will create corresponding entries in the tokenRingMLHistoryTable for the specified port at the specified interval. When the number of entries for a given port reaches the value (limit) contained in the historyControlBucketsGranted, the switch will delete the oldest entry for the port before adding a new entry for the port. The table is thus in effect a table of sliding windows, one for each historyControlTable entry.

After an tokenRingMLHistoryTable entry is created it remains accessible until one of the following conditions occurs:

1. The switch is reset (boots).
2. The entry is aged out by a new entry.
3. The corresponding historyControlTable entry is deleted.

The following two tables illustrate the name, data types, access and short description of the objects in tokenRingMLHistoryTable.

Figure 9. tokenRingMLHistoryTable (1 of 2)

Object	*Type*	*Access*	*Note*
tokenRingMLHistoryIndex	INTEGER	RO	Table Index One.
tokenRingMLHistorySampleIndex	INTEGER	RO	Table Index Two.
tokenRingMLHistoryIntervalStart	TimeTicks	RO	Inverval start time in hundredths of a second.
tokenRingMLHistoryDropEvents	Counter	RO	Packets dropped by the probe during this interval.
tokenRingMLHistoryMacOctets	Counter	RO	Not Implemented. Always zero.
tokenRingMacPkts	Counter	RO	Not Implemented. Always zero.
tokenRingMLHistoryRingPurgeEvents	Counter	RO	Number of ring purge events detected by the port during this interval.
tokenRingMLHistoryRingPurgePkts	Counter	RO	Ring purge MAC packets detected by the port during this interval.
tokenRingMLHistoryBeaconEvents	Counter	RO	Number of beacon events detected by the port during this interval.
tokenRingMLHistoryBeaconTime	TimeTicks	RO	Length of time in beacon state by the port during this interval.
tokenRingMLHistoryBeaconPkts	Counter	RO	Beacon MAC packets detected by the port during this interval.
tokenRingMLHistoryClaimTokenEvents	Counter	RO	Number of claim tokens events detected by the port during this interval.
tokenRingMLHistoryClaimTokenPkts	Counter	RO	Claim token MAC packets detected by the port during this interval.
tokenRingMLHistoryNAUNChanges	Counter	RO	Number of NAUN changes detected by the port during this interval.

Figure 10. tokenRingMLHistoryTable (2 of 2)

Object	*Type*	*Access*	*Note*
tokenRingMLHistoryLineErrors	Counter	RO	Number of line errors detected by the port during this interval.
tokenRingMLHistoryInternalErrors	Counter	RO	Number of internal errors detected by the port during this interval.
tokenRingMLHistoryBurstErrors	Counter	RO	Number of burst errors detected by the port during this interval.
tokenRingMLACHistoryErrors	Counter	RO	Number of address copy errors detected by the port during this interval.
tokenRingMLHistoryAbortErrors	Counter	RO	Number of abort errors detected by the port during this interval.
tokenRingMLHistoryLostFrameErrors	Counter	RO	Number of lost frame errors detected by the port during this interval.
tokenRingMLHistoryCongestionErrors	Counter	RO	Number of congestion errors detected by the port during this interval.
tokenRingMLHistoryFrameCopiedErrors	Counter	RO	Number of frame copy errors detected by the port during this interval.
tokenRingMLHistoryFrequencyErrors	Counter	RO	Number of frequency errors detected by the port during this interval.
tokenRingMLHistoryTokenErrors	Counter	RO	Number of token errors detected by the port during this interval.
tokenRingMLHistorySoftErrorReports	Counter	RO	Number of soft errors detected by the port during this interval.
tokenRingMLHistoryRingPollEvents	Counter	RO	Number of ring poll errors detected by the port during this interval.
tokenRingMLHistoryActiveStations	Counter	RO	Number of active stations detected by the port during this interval.

The tokenRingPHistoryTable

The tokenRingPHistoryTable contains the historical values of promiscusous statistics for Token Ring ports. The creation of tokenRingPHistoryTable entries is controlled by the entries in historyControlTable. For each "valid" entry in the historyControlTable, the switch will create corresponding entries in the tokenRingPHistoryTable for the specified port at the specified interval. When the number of entries for a given port reaches the value (limit) contained in the historyControlBucketsGranted, the switch will delete the oldest entry for the port before adding a new entry for the port. The table is thus in effect a table of sliding windows, one for each historyControlTable entry.

After an tokenRingPHistoryTable entry is created it remains accessible until one of the following conditions occurs:

1. The switch is reset (boots).
2. The entry is aged out by a new entry.
3. The corresponding historyControlTable entry is deleted.

The following two tables illustrate the name, data types, access and short description of the objects in tokenRingPHistoryTable.

Figure 11. tokenRingPHistoryTable (1 of 2)

Object	*Type*	*Access*	*Note*
tokenRingPHistoryIndex	INTEGER	RO	Table Index One.
tokenRingPHistorySampleIndex	INTEGER	RO	Table Index Two.
tokenRingMLHistoryIntervalStart	TimeTicks	RO	Inverval start time in hundredths of a second.
tokenRingPHistoryDropEvents	Counter	RO	Packets dropped by the probe during this interval.
tokenRingPHistoryDataOctets	Counter	RO	Octets received by the port during this interval.
tokenRingPHistoryDataPkts	Counter	RO	Packets received by the port during this interval.
tokenRingPHistoryDataBroadcastPkts	Counter	RO	Broadcast packets received by the port during this interval.
tokenRingPHistoryDataMulticastPkts	Counter	RO	Multicast packets received by the port during this interval.

Figure 12. tokenRingPHistoryTable (2 of 2)

Object	*Type*	*Access*	*Note*
tokenRingPHistory-DataPkts18to63Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPkts64to127Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPkts128to255Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPkts256to511Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPkts512to1023Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPDataPkts1024to2047Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPkts2048to4095Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPkts4096to8191Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPkts8192to18000Octets	Counter	RO	Not Implemented. Always zero.
tokenRingPHistory-DataPktsGreaterThan18000Octets	Counter	RO	Not Implemented. Always zero.

The alarmTable

The alarmTable contains the current values which control the monitoring of a specific Token Ring port counter. When the 8270/8272 boots, this table will contain zero entries. Previously created entries are NOT saved between boots. Entries may be created by the customer until the switch's maximum entries for the table have been reached. Creation of a valid entry will cause the switch to perform actions (send traps and/or create logTable entries) as specified by the eventTable entry for a rising and falling condition. Deletion of an entry will cause the switch to stop performing actions as specified by the eventTable entry for a rising and falling condition.

The following table illustrates the name, data types, access and short description of the objects in alarmTable.

Figure 13. alarmTable

Object	*Type*	*Access*	*Description*
alarmIndex	INTEGER	RO	Table Index One. Range 1 to 65535.
alarmInterval	INTEGER	RW	Alarm interval in seconds. Range 1 to 2,147,483,647.
alarmVariable	OID	RW	Object ID identifying the counter and port to monitor.
alarmSampleType	INTEGER	RW	Sample type. 1=absolute value. 2=delta value.
alarmValue	INTEGER	RO	Value of the port counter during the last interval.
alarmStartUpAlarm	INTEGER	RW	Action to take when the alarm becomes valid. 1=check for rising alarm condition. 2=check for falling alarm condition. 3=check for rising and falling alarm conditions.
alarmRisingThreshold	INTEGER	RW	Rising alarm threshold for the port counter. Range 1 to 2,147,483,647.
alarmFallingThreshold	INTEGER	RW	Falling alarm threshold for the port counter. Range 1 to 2,147,483,647.
alarmRisingEventIndex	INTEGER	RW	Rising alarm eventTable index which contains log and trap options. Range 0 to 65535.
alarmFallingEventIndex	INTEGER	RW	Falling alarm eventTable index which contains log and trap options. Range 0 to 65535.
alarmOwner	String	RW	Customer specified alarm owner name.
alarmStatus	INTEGER	RW	The current status of the entry. 1=valid. 2=create request (create an entry). 3=under creation. 4=invalid (delete the entry).

The following three tables illustrate the variables (port counters) which may be monitored where "n" is the ifIndex of the port.

Figure 14. alarm Variables (1 of 3)

Variable	*MIB*	*OID*
ifInOctets	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.10.n
ifInUcastPkts	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.11.n
ifInNUcastPkts	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.12.n
ifInDiscards	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.13.n
ifInErrors	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.14.n
ifInUnknownProtos	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.15.n
ifOutOctets	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.16.n
ifOutUcastPkts	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.17.n
ifOutNUcastPkts	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.18.n
ifOutDiscards	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.19.n
ifOutErrors	RFC1573 - ifTable	1.3.6.1.2.1.2.2.1.20.n
ifInMulticastPkts	RFC1573 - ifXTable	1.3.6.1.2.1.31.1.1.1.2.n
ifInBroadcastPkts	RFC1573 - ifXTable	1.3.6.1.2.1.31.1.1.1.3.n
ifOutMulticastPkts	RFC1573 - ifXTable	1.3.6.1.2.1.31.1.1.1.4.n
ifOutBroadcastPkts	RFC1573 - ifXTable	1.3.6.1.2.1.31.1.1.1.5.n

Figure 15. alarm Variables (2 of 2)

Variable	*MIB*	*OID*
tokenRingMLStats-DropEvents	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.3.n
tokenRingMLStats-RingPurgeEvents	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.6.n
tokenRingMLStats-RingPurgePkts	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.7.n
tokenRingMLStats-BeaconEvents	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.8.n
tokenRingMLStats-BeaconPkts	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.10.n
tokenRingMLStats-ClaimTokenEvents	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.11.n
tokenRingMLStats-ClaimTokenPkts	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.12.n
tokenRingMLStats-NAUNChanges	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.13.n
tokenRingMLStats-LineErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.14.n
tokenRingMLStats-InternalErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.15.n
tokenRingMLStats-BurstErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.16.n
tokenRingMLStats-ACErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.17.n
tokenRingMLStats-AbortErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.18.n
tokenRingMLStats-LostFrameErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.19.n
tokenRingMLStats-CongestionErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.20.n
tokenRingMLStats-FrameCopiedErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.21.n

Figure 16. alarm Variables (3 of 3)

Variable	*MIB*	*OID*
tokenRingMLStats-FrequencyErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.22.n
tokenRingMLStats-TokenErrors	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.23.n
tokenRingMLStats-SoftErrorReports	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.24.n
tokenRingMLStats-RingPollEvents	RFC1513 - tokenRingMLStatsTable	1.3.6.1.2.1.16.1.2.1.25.n
tokenRingPStats-DropEvents	RFC1513 - tokenRingPStatsTable	1.3.6.1.2.1.16.1.3.1.3.n
tokenRingPStats-DataOctets	RFC1513 - tokenRingPStatsTable	1.3.6.1.2.1.16.1.3.1.4.n
tokenRingPStats-DataPkts	RFC1513 - tokenRingPStatsTable	1.3.6.1.2.1.16.1.3.1.5.n
tokenRingPStats-DataBroadcastPkts	RFC1513 - tokenRingPStatsTable	1.3.6.1.2.1.16.1.3.1.6.n
tokenRingPStats-DataMulticastPkts	RFC1513 - tokenRingPStatsTable	1.3.6.1.2.1.16.1.3.1.7.n

Creating an alarmTable Entry: Before creating an alarmTable entry, the customer should first create eventTable entries for rising and/or falling conditions which are to be monitored by the alarmEntry. The eventTable entries must exist to SET the alarmRisingEventIndex and alarmFallingEventIndex. See "Creating an eventTable Entry".

An entry in the alarmTable may be created by issuing an SNMP SET of the alarmStatus object with a non-existent index and a value of 2 ("create request").

```
Example Parameters

IP Address...9.67.219.31 <------ Switch IP Address
Community....private <---------- read-write Community Name
Object.......alarmStatus
Index........1 <---------------- non-existent alarmIndex
Value........2 <---------------- create request
```

Optionally the SET PDU may also contain customer specified values for the following read-write objects:

- alarmInterval
- alarmVariable
- alarmSampleType
- alarmStartUpAlarm
- alarmRisingThreshold
- alarmFallingThreshold
- alarmRisingEventIndex
- alarmFallingEventIndex
- alarmOwner

Upon successful completion of the SET, a new entry will be created in the alarmTable with an alarmStatus of "under creation" (3) and switch defaults for any unspecified optional parameters.

Modifying an alarmTable Entry: While an entry has an alarmStatus of "under creation" (3), the following read-write objects may be modified:

- alarmInterval
- alarmVariable
- alarmSampleType
- alarmStartUpAlarm
- alarmRisingThreshold
- alarmFallingThreshold
- alarmRisingEventIndex
- alarmFallingEventIndex
- alarmOwner
- alarmStatus

After the alarmStatus of an entry has been SET to "valid" (1) by the manager, the switch will begin monitoring the port counter and performing actions as specified by rising eventTable entry and falling eventTable entry. While an entry has an alarmStatus of "valid" (1), the read-write objects of this table have the following modification properties:

- May NOT Be Modified
  1. alarmInterval
  2. alarmVariable
  3. alarmSampleType
  4. alarmStartUpAlarm
  5. alarmRisingThreshold
  6. alarmFallingThreshold
  7. alarmRisingEventIndex
  8. alarmFallingEventIndex
- May Be Modified
  1. alarmOwner
  2. alarmStatus

Deleting an alarmTable Entry: An entry in the alarmTable may be deleted by issuing an SNMP SET of the alarmStatus object with a valid index and a value of 4 ("invalid").

```
Example Parameters

IP Address...9.67.219.31 <------ Switch IP Address
Community....private <---------- read-write Community Name
Object.......alarmStatus
Index........1 <---------------- existent alarmIndex
Value........4 <---------------- invalid
```

Deletion of an entry will cause the switch to stop monitoring the port counter.

Deriving the Switch's alarmTable Limits: The switch has a limit on the number of alarmTable entries which may be created. An object containing the value of this limit is not provided by RFC1757 but this limit can be derived.

To derive the switch's limit on the number of alarmTable entries which may be created, simply create alarmTable entries until the switch rejects an additional "create request". The number of entries which were created prior to the rejection is the switch's limit on the number of alarmTable entries.

alarmTable OIDs: The following object identifiers (OIDs) may be helpful in natively manipulating the alarmTable where "n" is the alarmIndex.

Figure 17. alarmTable OIDs

Object	*OID*
alarmIndex	1.3.6.1.2.1.16.3.1.1.1.n
alarmInterval	1.3.6.1.2.1.16.3.1.1.2.n
alarmVariable	1.3.6.1.2.1.16.3.1.1.3.n
ifIndex	1.3.6.1.2.1.2.2.1.1.n
alarmSampleType	1.3.6.1.2.1.16.3.1.1.4.n
alarmValue	1.3.6.1.2.1.16.3.1.1.5.n
alarmStartUpAlarm	1.3.6.1.2.1.16.3.1.1.6.n
alarmRisingThreshold	1.3.6.1.2.1.16.3.1.1.7.n
alarmFallingThreshold	1.3.6.1.2.1.16.3.1.1.8.n
alarmRisingEventIndex	1.3.6.1.2.1.16.3.1.1.9.n
alarmFallingEventIndex	1.3.6.1.2.1.16.3.1.1.10.n
alarmOwner	1.3.6.1.2.1.16.3.1.1.11.n
alarmStatus	1.3.6.1.2.1.16.3.1.1.12.n

The eventTable

The eventTable contains the current values which control the action(s) taken where a port counter is detected to be in a rising or falling condition as specified by an entry in alarmTable. When the 8270/8272 boots, this table will contain zero entries. Previously created entries are NOT save between boots. Entries may created by the customer until the switch's maximum entries for the table have been reached. Creation of a valid entry in itself will NOT cause the switch to perform any action(s). A valid alarmTable entry must be created which contains the index of a valid eventTable entry before the switch will perform any actions for rising or falling conditions. Deletion of an entry will cause the switch to stop performing actions related to the entry which are currently active and delete all associated logEntries.

The following table illustrates the name, data types, access and short description of the objects in the eventTable.

Figure 18. eventTable

Object	*Type*	*Access*	*Description*
eventIndex	INTEGER	RO	Table Index One. Range 1 to 65535.
eventDescription	String	RW	Customer specified event description.
eventType	INTEGER	RW	Action to perform. 1=None. 2=create a logTable entry. 3=send a trap. 4=create a logTable entry and send a trap.
eventCommunity	String	RW	Community name to send in a trap. If not specified, the trap will not be sent.
eventLastTimeSent	TimeTicks	RO	Time in hundredths of a second of when the event last occurred.
eventOwner	String	RW	Customer specified event owner name.
eventStatus	INTEGER	RW	The current status of the entry. 1=valid. 2=create request (create an entry). 3=under creation. 4=invalid (delete the entry).

Creating an eventTable Entry: An entry in the eventTable may be created by issuing an SNMP SET of the eventStatus object with a non-existent index and a value of 2 ("create request").

```
Example Parameters

IP Address...9.67.219.31 <------ Switch IP Address
Community....private <---------- read-write Community Name
Object.......eventStatus
Index........1 <---------------- non-existent eventIndex
Value........2 <---------------- create request
```

Optionally the SET PDU may also contain customer specified values for the following read-write objects:

- eventDescription
- eventType
- eventSampleType
- eventCommunity
- eventOwner

Upon successful completion of the SET, a new entry will be created in the eventTable with an eventStatus of "under creation" (3) and switch defaults for any unspecified optional parameters.

Modifying an eventTable Entry: While an entry has an eventStatus of "valid" (1) or "under creation" (3), the following read-write objects may be modified:

- eventDescription
- eventType
- eventSampleType
- eventCommunity
- eventOwner
- eventStatus

Deleting an eventTable Entry: An entry in the eventTable may be deleted by issuing an SNMP SET of the eventStatus object with a valid index and a value of 4 ("invalid").

```
Example Parameters

IP Address...9.67.219.31 <------ Switch IP Address
Community....private <---------- read-write Community Name
Object.......eventStatus
Index........1 <---------------- existent eventIndex
Value........4 <---------------- invalid
```

Deletion of an entry will cause the switch to stop performing actions related to the entry which are currently active and delete all associated logEntries.

Deriving the Switch's eventTable Limits: The switch has a limit on the number of eventTable entries which may be created. An object containing the value of this limit is not provided by RFC1757 but this limit can be derived.

To derive the switch's limit on the number of eventTable entries which may be created, simply create eventTable entries until the switch rejects an additional "create request". The number of entries which were created prior to the rejection is the switch's limit on the number of eventTable entries.

eventTable OIDs: The following object identifiers (OIDs) may be helpful in natively manipulating the eventTable where "n" is the eventIndex.

Figure 19. eventTable OIDs

Object	*OID*
eventIndex	1.3.6.1.2.1.16.9.1.1.1.n
eventDescription	1.3.6.1.2.1.16.9.1.1.2.n
eventType	1.3.6.1.2.1.16.9.1.1.3.n
eventCommunity	1.3.6.1.2.1.16.9.1.1.4.n
eventLastTimeSent	1.3.6.1.2.1.16.9.1.1.5.n
eventOwner	1.3.6.1.2.1.16.9.1.1.6.n
eventStatus	1.3.6.1.2.1.16.9.1.1.7.n

The logTable

The logTable contains the information regarding rising and falling alarm conditions. The creation of logTable entries is controlled by the entries in the eventTable which are in turn associated with alarmTable entries. For each set of "valid" alarmTable and eventTable entries with an eventType of "log" (2) or "trap and log" (4), the switch will create corresponding entries in the logTable for the specified rising and/or falling port counter conditions. When the number of entries for a given port rising or falling condition reaches the switch's limit for a given eventTable entry, the switch will delete the oldest entry for the event before adding a new entry for the event. The table is thus in effect a table of sliding windows, one for each eventTable entry.

After a logTable entry is created it remains accessible until one of the following conditions occurs:

1. The switch is reset (boots).
2. The entry is aged out by a new entry.
3. The corresponding eventTable entry is deleted.

The following table illustrates the name, data types, access and short description of the objects in logTable.

Figure 20. logTable

Object	*Type*	*Access*	*Note*
logEventIndex	INTEGER	RO	Table Index One. eventTable index for the log entry. Range 1 to 65535.
logIndex	INTEGER	RO	Table Index Two. log index for the log entry. Range 1 to 65535.
logTime	TimeTicks	RO	Time in hundredths of a second of when the event was logged.
logDescription	String	RO	Customer specified log description copied from the event description.

The risingAlarm Trap

A risingAlarm trap contains information regarding rising alarm conditions. The sending of risingAlarm traps is controlled by the entries in the eventTable which are in turn associated with alarmTable entries. For each set of "valid" alarmTable and eventTable entries with an eventType of "trap" (3) or "trap and log" (4), the switch will send a risingAlarm trap for the specified rising port counter conditions to all of the switch's trap receivers. If the community name specified in the eventTable entry is NULL, the trap is NOT sent.

The following table illustrates the name, data types and short description of the objects in the risingAlarm Trap.

Figure 21. risingAlarm Trap

Object	*Type*	*Note*
alarmIndex	INTEGER	Index of the alarmTable entry related to this trap.
alarmVariable	OID	Object ID identifying the port counter.
alarmSampleType	INTEGER	Sample type. 1=absolute value. 2=delta value.
alarmValue	INTEGER	Value of the port counter when the rising alarm condition was detected.
alarmRisingThreshold	INTEGER	Rising alarm threshold for the port counter. Range 1 to 2,147,483,647.

The fallingAlarm Trap

A fallingAlarm trap contains information regarding falling alarm conditions. The sending of fallingAlarm traps is controlled by the entries in the eventTable which are in turn associated with alarmTable entries. For each set of "valid" alarmTable and eventTable entries with an eventType of "trap" (3) or "trap and log" (4), the switch will send a fallingAlarm trap for the specified rising port counter conditions to all of the switch's trap receivers. If the community name specified in the eventTable entry is NULL, the trap is NOT sent.

The following table illustrates the name, data types and short description of the objects in the fallingAlarm Trap.

Figure 22. fallingAlarm Trap

Object	*Type*	*Note*
alarmIndex	INTEGER	Index of the alarmTable entry related to this trap.
alarmVariable	OID	Object ID identifying the port counter.
alarmSampleType	INTEGER	Sample type. 1=absolute value. 2=delta value.
alarmValue	INTEGER	Value of the port counter when the falling alarm condition was detected.
alarmFallingThreshold	INTEGER	Falling alarm threshold for the port counter. Range 1 to 2,147,483,647.

ENABLING AND DISABLING THE EMBEDDED RMON FEATURE

The ibm8272TsEmbeddedRmonStatus Object

An ibm8272TsEmbeddedRmonStatus object has been added to the 8270/8272 Private MIB. This is a simple (single instance) object which when read reflects the current status of the Embedded RMON Feature (up or down); and when SET will either enable or disable the Embedded RMON Feature. When disabled, none of the tables or objects in the RMON MIB (RFC1757) and the Token Ring RMON MIB (RFC1513) will be accessible. As a side effect of disabling the Embedded RMON Feature, all internal storage related to any and all RFC1757 and RFC1513 tables and objects is freed; and is thus a quick and easy way of deleting all entries in all RMON tables. When the object is SET to up/enabled, the result depends on the current status of the Embedded RMON Feature. If the current status is up/enabled, nothing is changed or altered. If the current status is down/disabled, the Embedded RMON Feature is re-initialized.