Network Working Group
Request for Comments: 1317

B. Stewart, Editor
Xyplex, Inc.

April 1992

Definitions of Managed Objects for RS-232-like Hardware Devices

Status of this Memo

This document specifies an IAB standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "IAB Official Protocol Standards" for the standardization state and status of this protocol. Distribution of this memo is unlimited.

1. Abstract

This memo defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP based internets. In particular, it defines objects for the management of RS-232-like devices.

2. The Network Management Framework

The Internet-standard Network Management Framework consists of three components. They are:

RFC 1155 which defines the SMI, the mechanisms used for describing and naming objects for the purpose of management. RFC 1212 defines a more concise description mechanism, which is wholly consistent with the SMI.

RFC 1156 which defines MIB-I, the core set of managed objects for the Internet suite of protocols. RFC 1213, defines MIB-II, an evolution of MIB-I based on implementation experience and new operational requirements.

RFC 1157 which defines the SNMP, the protocol used for network access to managed objects.

The Framework permits new objects to be defined for the purpose of experimentation and evaluation.

3. Objects

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. Objects in the MIB are

defined using the subset of Abstract Syntax Notation One (ASN.1) [7] defined in the SMI. In particular, each object has a name, a syntax, and an encoding. The name is an object identifier, an administratively assigned name, which specifies an object type.

The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, we often use a textual string, termed the OBJECT DESCRIPTOR, to also refer to the object type.

The syntax of an object type defines the abstract data structure corresponding to that object type. The ASN.1 language is used for this purpose. However, the SMI [3] purposely restricts the ASN.1 constructs which may be used. These restrictions are explicitly made for simplicity.

The encoding of an object type is simply how that object type is represented using the object type's syntax. Implicitly tied to the notion of an object type's syntax and encoding is how the object type is represented when being transmitted on the network.

The SMI specifies the use of the basic encoding rules of ASN.1 [8], subject to the additional requirements imposed by the SNMP.

3.1. Format of Definitions

Section 5 contains the specification of all object types contained in this MIB module. The object types are defined using the conventions defined in the SMI, as amended by the extensions specified in [9,10].

4. Overview

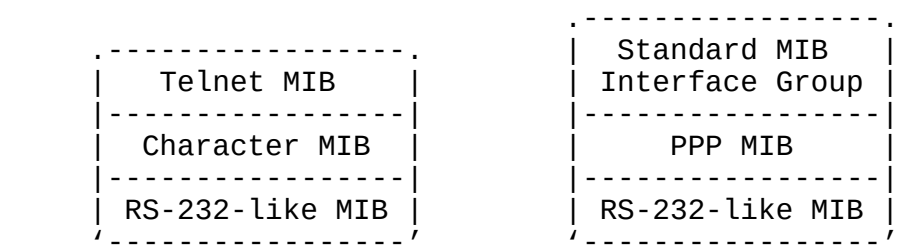
The RS-232-like Hardware Device MIB applies to interface ports that might logically support the Interface MIB, a Transmission MIB, or the Character MIB. The most common example is an RS-232 port with modem signals.

The RS-232-like MIB is one of a set of MIBs designed for complementary use. At this writing, the set comprises:

```
Character MIB
PPP MIB
RS-232-like MIB
Parallel-printer-like MIB
```

The RS-232-like MIB and the Parallel-printer-like MIB represent the physical layer, providing service to higher layers such as the Character MIB or PPP MIB. Further MIBs may appear above these.

The following diagram shows two possible "MIB stacks", each using the RS-232-like MIB.



The intent of the model is for the physical-level MIBs to represent the lowest level, regardless of the higher level that may be using it. In turn, separate higher level MIBs represent specific applications, such as a terminal (the Character MIB) or a network connection (the PPP MIB).

The RS-232-like Hardware Device MIB is mandatory for all systems that have such a hardware port supporting services managed through some other MIB, for example, the Character MIB or PPP MIB.

The MIB includes multiple similar types of hardware, and as a result contains objects not applicable to all of those types. Such objects are in a separate branch of the MIB, which is required when applicable and otherwise absent.

The RS-232-like Hardware Port MIB includes RS-232, RS-422, RS-423, V.35, and other asynchronous or synchronous, serial physical links with a similar set of control signals.

The MIB contains objects that relate to physical layer connections. Such connections may provide interesting hardware signals (other than for basic data transfer), such as RNG and DCD. Hardware ports also have such attributes as speed and bits per character.

Usefulness of error counters in this MIB depends on the presence of non-error character counts in higher level MIBs.

The MIB comprises one base object and four tables, detailed in the following sections. The tables contain objects for all ports, asynchronous ports, and input and output control signals.

5. Definitions

```
               RFC1317-MIB DEFINITIONS ::= BEGIN

                    IMPORTS
                       Counter
                               FROM RFC1155-SMI
                       transmission
                               FROM RFC1213-MIB
                       OBJECT-TYPE
                               FROM RFC-1212;

       -- this is the MIB module for RS-232-like hardware devices

       rs232    OBJECT IDENTIFIER ::= { transmission 33 }


       -- the generic RS-232-like group

       -- Implementation of this group is mandatory for all
       -- systems that have RS-232-like hardware ports
       -- supporting higher level services such as character
       -- streams or network interfaces

       rs232Number OBJECT-TYPE
           SYNTAX INTEGER
           ACCESS read-only
           STATUS mandatory
           DESCRIPTION
               "The number of ports (regardless of their current
               state) in the RS-232-like general port table."
           ::= { rs232 1 }


       -- the RS-232-like general Port table

       rs232PortTable OBJECT-TYPE
           SYNTAX SEQUENCE OF Rs232PortEntry
           ACCESS not-accessible
           STATUS mandatory
           DESCRIPTION
               "A list of port entries.  The number of entries is
               given by the value of rs232Number."
           ::= { rs232 2 }

       rs232PortEntry OBJECT-TYPE
           SYNTAX Rs232PortEntry
           ACCESS not-accessible
```

```
            STATUS mandatory
            DESCRIPTION
                "Status and parameter values for a port."
            INDEX { rs232PortIndex }
            ::= { rs232PortTable 1 }

        Rs232PortEntry ::=
            SEQUENCE {
                rs232PortIndex
                    INTEGER,
                rs232PortType
                    INTEGER,
                rs232PortInSigNumber
                    INTEGER,
                rs232PortOutSigNumber
                    INTEGER,
                rs232PortInSpeed
                    INTEGER,
                rs232PortOutSpeed
                    INTEGER    }

        rs232PortIndex OBJECT-TYPE
            SYNTAX INTEGER
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "A unique value for each port.  Its value ranges
                between 1 and the value of rs232Number.  By
                convention and if possible, hardware port numbers
                map directly to external connectors.  The value for
                each port must remain constant at least from one
                re-initialization of the network management agent to
                the next."
            ::= { rs232PortEntry 1 }

        rs232PortType OBJECT-TYPE
            SYNTAX INTEGER { other(1), rs232(2), rs422(3),
                             rs423(4), v35(5) }
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "The port's hardware type."
            ::= { rs232PortEntry 2 }

        rs232PortInSigNumber OBJECT-TYPE
            SYNTAX INTEGER
            ACCESS read-only
            STATUS mandatory
```

```
            DESCRIPTION
                "The number of input signals for the port in the
                input signal table (rs232PortInSigTable).  The table
                contains entries only for those signals the software
                can detect."
            ::= { rs232PortEntry 3 }

        rs232PortOutSigNumber OBJECT-TYPE
            SYNTAX INTEGER
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "The number of output signals for the port in the
                output signal table (rs232PortOutSigTable).  The
                table contains entries only for those signals the
                software can assert."
            ::= { rs232PortEntry 4 }

        rs232PortInSpeed OBJECT-TYPE
            SYNTAX INTEGER
            ACCESS read-write
            STATUS mandatory
            DESCRIPTION
                "The port's input speed in bits per second."
            ::= { rs232PortEntry 5 }

        rs232PortOutSpeed OBJECT-TYPE
            SYNTAX INTEGER
            ACCESS read-write
            STATUS mandatory
            DESCRIPTION
                "The port's output speed in bits per second."
            ::= { rs232PortEntry 6 }


        -- the RS-232-like Asynchronous Port group

        -- Implementation of this group is mandatory if the system
        -- has any asynchronous ports.  Otherwise it is not
        -- present.

        rs232AsyncPortTable OBJECT-TYPE
            SYNTAX SEQUENCE OF Rs232AsyncPortEntry
            ACCESS not-accessible
            STATUS mandatory
            DESCRIPTION
                "A list of asynchronous port entries.  The maximum
                entry number is given by the value of rs232Number.
```

```
            Entries need not exist for synchronous ports."
        ::= { rs232 3 }

    rs232AsyncPortEntry OBJECT-TYPE
        SYNTAX Rs232AsyncPortEntry
        ACCESS not-accessible
        STATUS mandatory
        DESCRIPTION
            "Status and parameter values for an asynchronous
            port."
        INDEX { rs232AsyncPortIndex }
        ::= { rs232AsyncPortTable 1 }

    Rs232AsyncPortEntry ::=
        SEQUENCE {
            rs232AsyncPortIndex
                INTEGER,
            rs232AsyncPortBits
                INTEGER,
            rs232AsyncPortStopBits
                INTEGER,
            rs232AsyncPortParity
                INTEGER,
            rs232AsyncPortAutobaud
                INTEGER,
            rs232AsyncPortParityErrs
                Counter,
            rs232AsyncPortFramingErrs
                Counter,
            rs232AsyncPortOverrunErrs
                Counter

        }

    rs232AsyncPortIndex OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "A unique value for each port.  Its value is the
            same as rs232PortIndex for the port."
        ::= { rs232AsyncPortEntry 1 }

    rs232AsyncPortBits OBJECT-TYPE
        SYNTAX INTEGER (5..8)
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
```

```
            "The port's number of bits in a character."
        ::= { rs232AsyncPortEntry 2 }

    rs232AsyncPortStopBits OBJECT-TYPE
        SYNTAX INTEGER { one(1), two(2),
                         one-and-half(3), dynamic(4) }
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
            "The port's number of stop bits."
        ::= { rs232AsyncPortEntry 3 }

    rs232AsyncPortParity OBJECT-TYPE
        SYNTAX INTEGER { none(1), odd(2), even(3),
                         mark(4), space(5) }
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
            "The port's sense of a character parity bit."
        ::= { rs232AsyncPortEntry 4 }

    rs232AsyncPortAutobaud OBJECT-TYPE
        SYNTAX INTEGER { enabled(1), disabled(2) }
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
            "A control for the port's ability to automatically
            sense input speed.

            When rs232PortAutoBaud is 'enabled', a port may
            autobaud to values different from the set values for
            speed, parity, and character size.  As a result a
            network management system may temporarily observe
            values different from what was previously set."
        ::= { rs232AsyncPortEntry 5 }

    rs232AsyncPortParityErrs OBJECT-TYPE
        SYNTAX Counter
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Total number of characters with a parity error,
            input from the port since system re-initialization
            and while the port state was 'up' or 'test'."
        ::= { rs232AsyncPortEntry 6 }

    rs232AsyncPortFramingErrs OBJECT-TYPE
        SYNTAX Counter
```

```
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "Total number of characters with a framing error,
                input from the port since system re-initialization
                and while the port state was 'up' or 'test'."
            ::= { rs232AsyncPortEntry 7 }

        rs232AsyncPortOverrunErrs OBJECT-TYPE
            SYNTAX Counter
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "Total number of characters with an overrun error,
                input from the port since system re-initialization
                and while the port state was 'up' or 'test'."
            ::= { rs232AsyncPortEntry 8 }


        -- the RS-232-like Synchronous Port group

        -- Implementation of this group is mandatory if the system
        -- has any synchronous ports.  Otherwise it is not
        -- present.

        rs232SyncPortTable OBJECT-TYPE
            SYNTAX SEQUENCE OF Rs232SyncPortEntry
            ACCESS not-accessible
            STATUS mandatory
            DESCRIPTION
                "A list of synchronous port entries.  The maximum
                entry number is given by the value of rs232Number.
                Entries need not exist for asynchronous ports."
            ::= { rs232 4 }

        rs232SyncPortEntry OBJECT-TYPE
            SYNTAX Rs232SyncPortEntry
            ACCESS not-accessible
            STATUS mandatory
            DESCRIPTION
                "Status and parameter values for a synchronous
                port."
            INDEX { rs232SyncPortIndex }
            ::= { rs232SyncPortTable 1 }

        Rs232SyncPortEntry ::=
            SEQUENCE {
                rs232SyncPortIndex
```

```
                INTEGER,
            rs232SyncPortClockSource
                INTEGER,
            rs232SyncPortFrameCheckErrs
                Counter,
            rs232SyncPortTransmitUnderrunErrs
                Counter,
            rs232SyncPortReceiveOverrunErrs
                Counter,
            rs232SyncPortInterruptedFrames
                Counter,
            rs232SyncPortAbortedFrames
                Counter
        }

    rs232SyncPortIndex OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "A unique value for each port.  Its value is the
            same as rs232PortIndex for the port."
        ::= { rs232SyncPortEntry 1 }

    rs232SyncPortClockSource OBJECT-TYPE
        SYNTAX INTEGER  { internal(1), external(2), split(3) }
        ACCESS read-write
        STATUS mandatory
        DESCRIPTION
            "Source of the port's bit rate clock. 'split' means
            the tranmit clock is internal and the receive clock
            is external."
        ::= { rs232SyncPortEntry 2 }

    rs232SyncPortFrameCheckErrs OBJECT-TYPE
        SYNTAX Counter
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Total number of frames with an invalid frame check
            sequence, input from the port since system
            re-initialization and while the port state was 'up'
            or 'test'."
        ::= { rs232SyncPortEntry 3 }

    rs232SyncPortTransmitUnderrunErrs OBJECT-TYPE
        SYNTAX Counter
        ACCESS read-only
```

```
            STATUS mandatory
            DESCRIPTION
                "Total number of frames that failed to be
                transmitted on the port since system
                re-initialization and while the port state was 'up'
                or 'test' because data was not available to the
                transmitter in time."
            ::= { rs232SyncPortEntry 4 }

        rs232SyncPortReceiveOverrunErrs OBJECT-TYPE
            SYNTAX Counter
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "Total number of frames that failed to be received
                on the port since system re-initialization and while
                the port state was 'up' or 'test' because the
                receiver did not accept the data in time."
            ::= { rs232SyncPortEntry 5 }

        rs232SyncPortInterruptedFrames OBJECT-TYPE
            SYNTAX Counter
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "Total number of frames that failed to be received
                or transmitted on the port due to loss of modem
                signals since system re-initialization and while the
                port state was 'up' or 'test'."
            ::= { rs232SyncPortEntry 6 }

        rs232SyncPortAbortedFrames OBJECT-TYPE
            SYNTAX Counter
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "Number of frames aborted on the port due to
                receiving an abort sequence since system
                re-initialization and while the port state was 'up'
                or 'test'."
            ::= { rs232SyncPortEntry 7 }


        -- the Input Signal table

        rs232InSigTable OBJECT-TYPE
            SYNTAX SEQUENCE OF Rs232InSigEntry
            ACCESS not-accessible
```

```
            STATUS mandatory
            DESCRIPTION
                "A list of port input control signal entries."
            ::= { rs232 5 }

        rs232InSigEntry OBJECT-TYPE
            SYNTAX Rs232InSigEntry
            ACCESS not-accessible
            STATUS mandatory
            DESCRIPTION
                "Input control signal status for a hardware port."
            INDEX { rs232InSigPortIndex, rs232InSigName }
            ::= { rs232InSigTable 1 }

        Rs232InSigEntry ::=
            SEQUENCE {
                rs232InSigPortIndex
                    INTEGER,
                rs232InSigName
                    INTEGER,
                rs232InSigState
                    INTEGER,
                rs232InSigChanges
                    Counter
            }

        rs232InSigPortIndex OBJECT-TYPE
            SYNTAX INTEGER
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "The value of rs232PortIndex for the port to which
                this entry belongs."
            ::= { rs232InSigEntry 1 }

        rs232InSigName OBJECT-TYPE
            SYNTAX INTEGER { rts(1), cts(2), dsr(3), dtr(4), ri(5),
                             dcd(6), sq(7), srs(8), srts(9),
                             scts(10), sdcd(11) }
            ACCESS read-only
            STATUS mandatory
            DESCRIPTION
                "Identification of a hardware signal, as follows:

                    rts    Request to Send
                    cts    Clear to Send
                    dsr    Data Set Ready
                    dtr    Data Terminal Ready
```

```
                ri      Ring Indicator
                dcd     Received Line Signal Detector
                sq      Signal Quality Detector
                srs     Data Signaling Rate Selector
                srts    Secondary Request to Send
                scts    Secondary Clear to Send
                sdcd    Secondary Received Line Signal Detector
            "
        REFERENCE
            "EIA Standard RS-232-C, August 1969."
        ::= { rs232InSigEntry 2 }

    rs232InSigState OBJECT-TYPE
        SYNTAX INTEGER { none(1), on(2), off(3) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "The current signal state."
        ::= { rs232InSigEntry 3 }

    rs232InSigChanges OBJECT-TYPE
        SYNTAX Counter
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "The number of times the signal has changed from
            'on' to 'off' or from 'off' to 'on'."
        ::= { rs232InSigEntry 4 }


    -- the Output Signal table

    rs232OutSigTable OBJECT-TYPE
        SYNTAX SEQUENCE OF Rs232OutSigEntry
        ACCESS not-accessible
        STATUS mandatory
        DESCRIPTION
            "A list of port output control signal entries."
        ::= { rs232 6 }

    rs232OutSigEntry OBJECT-TYPE
        SYNTAX Rs232OutSigEntry
        ACCESS not-accessible
        STATUS mandatory
        DESCRIPTION
            "Output control signal status for a hardware port."
        INDEX { rs232OutSigPortIndex, rs232OutSigName }
        ::= { rs232OutSigTable 1 }
```

```
    Rs232OutSigEntry ::=
        SEQUENCE {
            rs232OutSigPortIndex
                INTEGER,
            rs232OutSigName
                INTEGER,
            rs232OutSigState
                INTEGER,
            rs232OutSigChanges
                Counter
        }

    rs232OutSigPortIndex OBJECT-TYPE
        SYNTAX INTEGER
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "The value of rs232PortIndex for the port to which
            this entry belongs."
        ::= { rs232OutSigEntry 1 }

    rs232OutSigName OBJECT-TYPE
        SYNTAX INTEGER { rts(1), cts(2), dsr(3), dtr(4), ri(5),
                         dcd(6), sq(7), srs(8), srts(9),
                         scts(10), sdcd(11) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "Identification of a hardware signal, as follows:

                rts    Request to Send
                cts    Clear to Send
                dsr    Data Set Ready
                dtr    Data Terminal Ready
                ri     Ring Indicator
                dcd    Received Line Signal Detector
                sq     Signal Quality Detector
                srs    Data Signaling Rate Selector
                srts   Secondary Request to Send
                scts   Secondary Clear to Send
                sdcd   Secondary Received Line Signal Detector
            "
        REFERENCE
            "EIA Standard RS-232-C, August 1969."
        ::= { rs232OutSigEntry 2 }
```

```
    rs232OutSigState OBJECT-TYPE
        SYNTAX INTEGER { none(1), on(2), off(3) }
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "The current signal state."
        ::= { rs232OutSigEntry 3 }

    rs232OutSigChanges OBJECT-TYPE
        SYNTAX Counter
        ACCESS read-only
        STATUS mandatory
        DESCRIPTION
            "The number of times the signal has changed from
            'on' to 'off' or from 'off' to 'on'."
        ::= { rs232OutSigEntry 4 }

    END
```

6. Acknowledgements

Based on several private MIBs, this document was produced by the Character MIB Working Group:

Anne Ambler, Spider
Charles Bazaar, Emulex
Christopher Bucci, Datability
Anthony Chung, Hughes LAN Systems
George Conant, Xyplex
John Cook, Chipcom
James Davin, MIT-LCS
Shawn Gallagher, DEC
Tom Grant, Xylogics
Frank Huang, Emulex
David Jordan, Emulex
Satish Joshi, SynOptics
Frank Kastenholz, Clearpoint
Ken Key, University of Tennessee
Jim Kinder, Fibercom
Rajeev Kochhar, 3Com
John LoVerso, Xylogics
Keith McCloghrie, Hughes LAN Systems
Donalpd Merritt, BRL
David Perkins, 3Com
Jim Reinstedler, Ungerman-Bass
Marshall Rose, PSI
Ron Strich, SSDS
Dean Throop, DG

Bill Townsend, Xylogics
Jesse Walker, DEC
David Waitzman, BBN
Bill Westfield, cisco

7. References

[1] Cerf, V., "IAB Recommendations for the Development of Internet Network Management Standards", RFC 1052, NRI, April 1988.

[2] Cerf, V., "Report of the Second Ad Hoc Network Management Review Group", RFC 1109, NRI, August 1989.

[3] Rose M., and K. McCloghrie, "Structure and Identification of Management Information for TCP/IP-based internets", RFC 1155, Performance Systems International, Hughes LAN Systems, May 1990.

[4] McCloghrie K., and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1156, Hughes LAN Systems, Performance Systems International, May 1990.

[5] Case, J., Fedor, M., Schoffstall, M., and J. Davin, Simple Network Management Protocol", RFC 1157, SNMP Research, Performance Systems International, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[6] McCloghrie K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets", RFC 1213, Performance Systems International, March 1991.

[7] Information processing systems - Open Systems Interconnection - Specification of Abstract Syntax Notation One (ASN.1), International Organization for Standardization, International Standard 8824, December 1987.

[8] Information processing systems - Open Systems Interconnection - Specification of Basic Encoding Rules for Abstract Notation One (ASN.1), International Organization for Standardization, International Standard 8825, December 1987.

[9] Rose, M., and K. McCloghrie, Editors, "Concise MIB Definitions", RFC 1212, Performance Systems International, Hughes LAN Systems, March 1991.

[10] Rose, M., Editor, "A Convention for Defining Traps for use with the SNMP", RFC 1215, Performance Systems International, March 1991.

8. Security Considerations

Security issues are not discussed in this memo.

9. Author's Address

Bob Stewart
Xyplex, Inc.
330 Codman Hill Road
Boxborough, MA 01719

Phone: (508) 264-9900
EMail: rlstewart@eng.xyplex.com