eZ80® Microprocessors

SNMP v1 Support for the ZiLOG Internet Protocol Stack, Beta 2

TA000501-0503 Technical Article

Introduction

What is SNMP?

The Simple Network Management Protocol, SNMP, is a protocol for accessing a database of objects. In SNMP, this database is called a Management Information Base (MIB). Each object in the MIB has a name, a type, and a value. The protocol can be used to read or write values in the MIB by using the **Get**, **Get Next**, or **Set** operations. Requests originate from the SNMP Management Entity (similar in concept to a client application) and are sent to the SNMP Agent (similar in concept to a server application) that manages the MIB of interest. After the SNMP Agent processes a request, it returns relevant information to the Management Entity. The Management Entity can obtain information about objects in the MIB using the **Get** or **Get Next** requests or it can modify the value of an object in the MIB using the Set request. Only objects specified as read/write can be modified using **Set**.

TCP/IP devices implementing SNMP are required to include a subset of the standard MIB-II objects that are applicable to each device in the local MIB (see RFC 1213, *Management Information Base for Network Management of TCP/IP-based Networks MIB-II*). In addition, the SNMP Agent can allow the MIB to be augmented with user-defined objects. As a result, the Management Entity is allowed to obtain information about the Agent device that is unique to that device's application. For example, suppose the device in which the SNMP Agent is embedded includes a sensor capable of measuring the temperature of the processor. The designer of the MIB extensions for this device could therefore add an object to the MIB containing a value that reflects the processor temperature. Of course, the Agent device would also be required to include logic to update this value at a reasonable interval. If a Management Entity was to query the value of this object (using Get), it would be able to remotely obtain the processor temperature of the Agent device. Similarly, the designer of the MIB extensions for this device could add a second object to the MIB to control the speed of a cooling fan. The value the Management Entity assigns to this object (using Set) could be used by logic in the Agent device to program the speed (in revolutions per minute) of the cooling fan.

In addition, the SNMP version 1 protocol includes a **Trap** primitive that allows an SNMP Agent to alert a Management Entity about specific events. The alert is in the form of an SNMP **Trap** message that can contain object information (name, type, and value). Continuing the example above, the SNMP Agent device could be programmed to generate a **Trap** message that is sent to a Management Entity if the processor temperature exceeded some predefined threshold. The **Trap** message could contain an object containing a value that reflects the temperature reading that triggered the **Trap**.

Object Names

Objects are named using object identifiers and arranged in the MIB hierarchically, according to each object's name. Every object identifier is a collection of subidentifiers separated by periods. In this way, the MIB can be thought of as a tree of nodes, where each node (subidentifier) can have several branches (child nodes) leading to lower objects. Therefore, a complete object identifier is simply the concatenated string of subidentifiers leading from the unnamed Root node to the object of interest. In printed form, the object identifier can either be represented using text subidentifiers or numbers. In SNMP implementations, object

identifiers are represented as a series of numbers according to the ASN.1 BER definition of an object identifier.

For example, the object identifier corresponding to the *TCP* group in the MIB-II specification is *iso.org.dod.internet.mgmt.mib.tcp*, or `1.3.6.1.2.1.6`, and the object identifier of the IP group is *iso.org.dod.internet.mgmt.mib.ip*, or `1.3.6.1.2.1.4`. Within each of these groups, the MIB-II specification identifies several child objects.

Object Type

Objects within the SNMP MIB are restricted to using a subset of the primitive data types defined within the ASN.1 standards, such as *integers*, *octet strings*, and *object identifiers*. In addition, objects can be defined using SNMP-specific data types such as *IP address*, *counter*, *gauge*, and *timeticks*—each of which are defined using ASN.1 primitive data types. SNMP also allows these primitive data types to be aggregated to create lists or tables using the ASN.1 constructor type *sequence*. By using a restricted set of data types, SNMP management tools from one vendor can interoperate with agents from a different vendor, because they all "speak the same language" of ASN.1.

ZIPS Beta 2 SNMP Implementation

The Beta 2 SNMP implementation includes an SNMP Agent that responds to **Get**, **Get Next**, and **Set** requests originating from a remote SNMP Management Entity. Beta 2 does not include a Management Entity that can direct **Get**, **Get Next**, or **Set** requests to a remote Agent.

The structure of the MIB is defined at compile time. After the MIB is created, the Beta 2 SNMP library code that implements the Agent automatically responds to requests for all static objects within the MIB received from the remote Management Entity. At the user's discretion, dynamic tables may be added to the MIB that require the user to provide support routines that the SNMP library can use to manipulate objects within the table.

User-Extensible MIB

In Beta 2, the MIB is specified in the *mib* array within the `snmib.c` source file in the `\conf` subdirectory. To add your own SNMP objects to the MIB, include the `snmib.c` file with your project and add the appropriate entries to the *mib* array. Full details will be included in the Beta 2 version of the ZiLOG Internet Protocol Stack (ZIPS) Programmer's Guide (RM0002). Note that you can either add single variables to the MIB (i.e., primitive objects) or tables (constructed objects).

Objects that you add to the MIB should be placed under the *private.enterprises* branch (`1.3.6.1.4.1`) or the *experimental* branch (`1.3.6.1.3`) using an identifier registered with the IANA. For example, ZIPS Beta 2 includes a ZiLOG-specific extension to the MIB to identify the version of the software stack. The extension is located beneath the *private.enterprises.zilog* branch of the MIB. The single instance of this object is named *private.enterprises.zilog.zips.version.0* (`1.3.6.1.4.1.12897.1.1.0`). It is a type of octet string and contains a value of `Beta 2`.

Standard MIB-II Groups included in Beta 2

ZIPS Beta 2 includes the following groups of objects as specified in RFC 1213:

- System Group
- Interfaces Group
- Address Translation Group
- IP Group
- ICMP Group
- TCP Group
- UDP Group
- SNMP Group

Standard MIB-II Groups Not included in Beta 2

ZIPS Beta 2 does not include the following groups of objects as specified in RFC 1213:

- EGP Group
- Transmission Group

Traps Supported in Beta 2

The following SNMP v1 traps have been implemented in Beta 2:

- Cold Start Trap
- Link Down Trap
- Link Up Trap
- Authentication Failure Trap
- Enterprise Specific Traps

To allow user-defined enterprise-specific traps, the SNMP layer in Beta 2 includes a **Trap** API that can be called from application programs. The API allows the programmer to specify the enterprise-specific **Trap** code and include 0, 1, or more SNMP objects to include in the trap message. All traps are directed towards the user-specified trap target device. Each of the standard trap messages can be suppressed at the user's discretion.

The following SNMP v1 traps are not implemented in Beta 2.

- Warm Start Trap
- EGP Neighbor Loss

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

ZiLOG Worldwide Headquarters
532 Race Street
San Jose, CA 95126
Telephone: 408.558.8500
Fax: 408.558.8300
www.zilog.com