Application Note

Using the eZ80F91-Based SNMP v1 Agent in a Networked Environment

AN018102-0108

Abstract

This Application Note demonstrates the working of the Simple Network Management Protocol (SNMP v1) in Zilog's TCP/IP software suite (ZTP) implemented for the eZ80Acclaim*Plus!*™ product line of microcontroller units (MCU) and microprocessor units (MPU). SNMP provides an easy way of querying a network node from a remote location to identify and log the network node's working status. The SNMP implementation supported by ZTP is an agent protocol controlling its own Management Information Base (MIB).

This Application Note demonstrates the communication between the ZTP-SNMP agent software on the eZ80F91 MCU and an SNMP manager software on a PC. The SNMP manager is the Windows version of a non-commercial suite (UCD-SNMP). Although the UCD-SNMP was used to test this application, any other SNMP-compliant managers can be used. Zilog® does not endorse or promote specific SNMP managers.

Note: *The source code for the SNMP application is in the AN0181-SC01.zip file available for download at www.zilog.com.*

Zilog Product Overview

This section contains brief overviews of the Zilog products used in this Application Note, which includes the eZ80Acclaim*Plus!* MCU and the full-feature ZTP software suite.

eZ80Acclaim*Plus!*™ MCU Family Overview

The eZ80Acclaim*Plus!* family of MCUs includes Flash and non-Flash products. The Flash-based eZ80Acclaim*Plus!* MCUs, device numbers eZ80F91, eZ80F92, and eZ80F93, are an exceptional value for designing high performance embedded applications. With speeds up to 50 MHz and an on-chip Ethernet MAC (eZ80F91 only), you have the performance necessary to execute complex applications supporting networking functions quickly and efficiently. Combining on-chip Flash and SRAM, eZ80Acclaim*Plus!* devices provide the memory required to implement communication protocol stacks and achieve flexibility when performing in-system updates of application firmware.

Zilog also offers two eZ80Acclaim*Plus!* devices without Flash memory: the eZ80L92 and eZ80190 MPUs.

ZTP Overview

The ZTP Software Suite integrates a rich set of networking services with an efficient real-time operating system (RTOS). The operating system is a compact preemptive multitasking, multithreaded kernel with inter-process communications (IPC) support, and soft real-time attributes. Table 1 lists the standard network protocols implemented as part of the embedded TCP/IP protocol stack in ZTP.

Table 1. Standard Network Protocols in ZTP

HTTP	TFTP	SMTP	Telnet	IP	PPP
DHCP	DNS	TIMEP	SNMP	TCP	UDP
ICMP	IGMP	ARP	RARP		

Many TCP/IP application protocols are designed using the client-server model. The final stack size is link-time configurable and determined by the protocols included in the build.

Discussion

This section provides a brief overview of the SNMP protocol and lists the SNMP-related functions available in ZTP. ZTP is Zilog's TCP/IP software suite[1] that includes SNMP among other protocols. An overview of the SNMP manager entity used for demonstrating the SNMP communication is included in this section.

Simple Network Management Protocol

The Simple Network Management Protocol is an IETF standard protocol layer that runs on either the TCP or User Datagram Protocol (UDP) layer of the TCP/IP stack. It is defined by RFC 1157 as a protocol for exchanging information between a (client) manager and the agents (server) in a networked environment, either on LAN in a closed intranet or over the Internet. The SNMP manager can be a PC-based software utility and the SNMP agents are the network elements running the agent/server part of SNMP software. The SNMP agent's services over the Ethernet are either UDP- or TCP-based depending on the requirements for reliability of the transaction and guarantee of delivery.

The client program (called the network manager) makes virtual connections to a server program (called the SNMP agent) executing on a remote network device. The data base controlled by the SNMP agent is referred to as the Management Information Base (MIB), which consists of a standard set of variables called the Object Identifiers (OID). Both the manager and the agent interact through a set of commands executed on a MIB entries. The OIDs are accessed and updated by the manager or the agent.

[1]For more details on ZTP, refer to *Zilog TCP/IP Software Suite Programmer's Guide Reference Manual (RM0008)*.

The network manager queries the agent constantly through a set of messages in order to identify and log the agent's working status. A standard SNMP message consists of information on the SNMP version number, the community, the error status, and the OID number, besides the usual IP and UDP datagram/TCP headers. This message format conveys sufficient information to the manager or agent to act further. A trap message differs from the other messages, it is used by the agent to inform the manager that an event has occurred without the manager querying for it; for example, an error condition or change in value of an OID.

The SNMP architecture and its working details are covered in RFC 1157 issued by IETF.

SNMP Functions Available in ZTP

Table 2 lists the services available for applications intending to use the SNMP protocol supported by ZTP. `TrapGen()` is an API function and `Get`, `Get Next` and `Set` are primitives.

Table 2. ZTP-SNMP Services

Primitive/API Name	Description
`Get`	Obtain status of the specified OID
`Get Next`	Obtain status of the next OID to the one specified
`Set`	Modify value of the specified write-able OID
`TrapGen()`	Generate a trap on specific event occurrence

The SNMP implementation in ZTP is an embedded agent entity, which is a server program that replies to the queries generated by the SNMP manager entity, which is a client program on a PC. The UDP port 161 is configured on the ZTP-SNMP agent and is used to receive queries from the manager. The manager transmits SNMP messages on the UDP port 162.

For more details on ZTP-SNMP services, refer to *Zilog TCP/IP Software Suite Programmer's Guide*

Reference Manual (RM0008), available for download at www.zilog.com.

Applying SNMP in Practice

A typical scenario to illustrate ZTP-SNMP services is a factory automation environment. A number of eZ80F91 MCU-based products are networked together, each running ZTP with the SNMP agent code. A PC-based SNMP manager, on the same network or accessible via the Internet, is used to query the eZ80F91-based SNMP agents for values of specific variables.

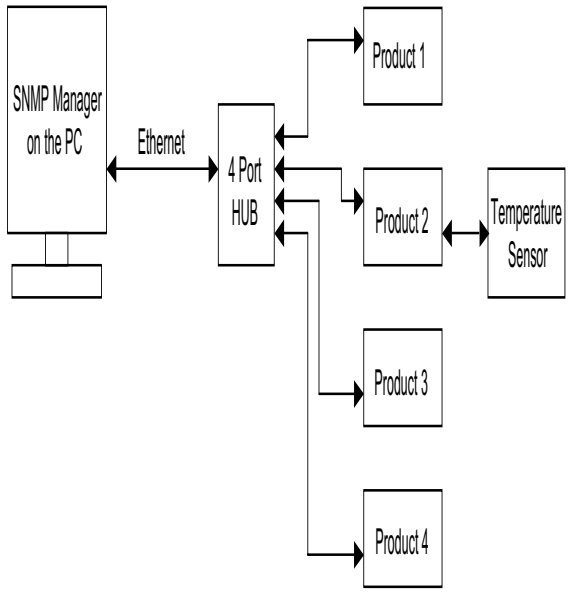

Figure 1. Using ZTP-SNMP in a Networked Factory Setup

In Figure 1, an eZ80F91-based SNMP agent (Product 2) obtains input from a temperature sensor. This agent is queried by the PC-based SNMP manager for temperature values at constant time intervals. In addition, the agent can issue a trap message to the SNMP manager when the temperature value reaches the upper threshold, alerting the SNMP manager about the situation. The SNMP manager can either log the event or take further appropriate action.

SNMP Manager Utility

UCD-SNMP[2] (available on www.net-snmp.org) is an open-source tool suite consisting of tools and libraries relating to the Simple Network Management Protocol. The suite consists of a Microsoft Windows-based SNMP manager utility among others, which is utilized to query the OIDs present in the MIB table of the eZ80F91-based ZTP-SNMP agents.

The UCD-SNMP manager supports SNMP v1, v2c, and v3, but the ZTP-SNMP agent is restricted to use SNMP v1 only. The UCD-SNMP manager must be configured appropriately for the reception of trap messages generated by the ZTP-SNMP agent. The available user interface for the UCD-SNMP manager utility is Command-Line-based and the entire text of the commands should be typed at the MS DOS prompt.

Developing the Application with eZ80F91 Development Kit

This section explains how to interface the eZ80F91 Development Platform (running the ZTP-SNMP agent) with the PC (running the UCD-SNMP manager) to exchange SNMP messages.

Hardware Interface

Figure 2 on page 7 displays a block diagram of the eZ80F91 Development Platform running the ZTP-SNMP agent and a Windows-based PC running the Windows-based UCD-SNMP manager software.

Software Implementation

The software as implemented for this application note consists of modifying the existing MIB table of the ZTP-SNMP agent by adding new OIDs, and generating a trap message by the ZTP-SNMP agent protocol when a switch on the eZ80 Development

[2]Although the UCD-SNMP was used to test this application, any other SNMP-compliant managers can be used. Zilog® does not endorse or promote specific SNMP managers.

Platform is pressed. This trap message is sent to the PC-based SNMP manager, and is displayed at the DOS prompt.

The SNMPapp.c runs on the eZ80 Development Platform, as a part of the ZTP-SNMP agent. The main purpose of this code is to update the variable newval in the MIB table, which is accessed by the SNMP manager. The value of the variable newval is toggled between 00 and 01 on every press of the switch SW1 (PB0). The trap message is generated when the switch SW2 (PB1) on the eZ80 Development Platform is pressed. This trap message is sent to the SNMP manager with the current value of the variable newval. The flowchart in Appendix A—Flowcharts on page 11 displays the working of this code.

The commands (queries) from the SNMP manager are configured appropriately to access the variables (OIDs) from the eZ80F91-based ZTP-SNMP agent and to display the received trap messages. The SNMP manager and the ZTP-SNMP agent exchange SNMP messages in the form of UDP datagrams.

The descriptions of the various variables, functions, and APIs used in the files are explained below:

SNMPapp.h

The variable newval, its type and the OID value are declared inside the structure snmpObj. The variable is added in the structure mib[] in the file snmib.c, and its value is presented to the SNMP manager when queried.

SNMPapp.c

The tasks performed by the code in this file are listed below:

- The eZ80F91 MCU's GPIO port pins PB0 and PB1, connected to switches SW1 and SW2 respectively, are initialized inside the function PortBInit().
- The function, UpdateSwitchInput(), polls switch SW1 and toggles the value of the variable newval accordingly.
- Switch SW2 is polled next. If it is pressed, a trap message is generated using the TrapGen() API. The arguments to the TrapGen() API consist of the type of trap (SN_TRAP_ENTERPRISE_SPECIFIC), the code type (signifies serial number of this trap), the number of variables (1 for newval), and the pointer to the object (snmpObj). A number of traps can be configured in a similar way according to the requirement.

The file snmp_conf.c (ZTP file) contains the target (SNMP manager) IP address and the port number in the string variable snmp_trap_target. This variable must be modified to reflect the user's customized IP address. The SNMP agent sends the trap messages to this IP address.

Adding and Integrating SNMP Demo Files to ZTP

The SNMP Demo described in this Application Note requires the eZ80 Development Platform (with the eZ80F91 MCU) and the ZTP. For the Demo execution, some of the files specific to the Demo must be added and integrated to the ZTP stack before it is downloaded onto the eZ80 Development Platform.

The Demo files that must be added to the ZTP project files are in the AN0181-SC01.zip file available on the Zilog website. The Demo files are of C (*.c) file type.

The ZTP stack is available on the Zilog website and can be downloaded to a PC with a user registration key. ZTP can be installed in any location as specified by you; its default location is C:\Program Files\ZiLOG.

Follow the steps to add and integrate the Demo files to the ZTP stack:

1. Download ZTP, browse to the location where ZTP is downloaded, and open the `..\SNMPDemo` folder.
2. Download the `AN0181-SC01.zip` file and extract its contents to a folder on your PC. The code files are for the SNMP agent entity.
3. Select and copy the *.c file, `SNMPapp.c`, located in the `\<extracted folder>` folder and paste it into the `..\ZTP\SNMPDemo` directory.
4. Select and copy the *.h file, `SNMPapp.h`, located in the `\<extracted folder>` folder and paste it into the `..\ZTP\includes` directory.
5. Launch ZDS II—eZ80Acclaim!, and open the `AcclaimSNMPDemo.pro` file available in the path: `..\ZTP\SNMPDemo`.
6. Add the file `SNMPapp.c` to the `AcclaimSNMPDemo.pro` from within the ZDS II IDE.
7. Open the `main.c` file of the AcclaimSNMPDemo project and enter the IP address for the eZ80F91 module in the following `BootInfo` structure definition:

```
struct BootInfo Bootrecord = {
"192.168.1.1",/* Default IP address */
"192.168.1.4",/* Default Gateway */
"192.168.1.5",/* Default Timer Server */
"192.168.1.6",/* Default File Server */
"",
"192.168.1.7",/* Default Name Server */
"",
0xffffff00UL/* Default Subnet Mask */
};
```

   The Bootrecord variable contains the network parameters and settings (in the four-octet dotted decimal format) that are specific to the local area network at Zilog as default.

   Note: *Modify the above structure definition with appropriate IP addresses within your local area network.*

8. In the `main.c` file, add the following line to include the `network.h` file:

```
#include <network.h>
```

   and add the following external function declarations:

```
extern void PortBInit(void);//
Initialize pins PB0 &
// PB1

extern int UpdateSwitchIn-
put(void);// Check & update switch
// status
```

9. In the `main.c` file, add the following lines of code before the `return(OK);` call:

```
PortBInit();// Initialize portB

while(1)// Main loop

{

  UpdateSwitchInput();// Check SW1
and SW2 and
// update variables

  sleep(1);// Sleep for 1 second

}
```

10. Open the `eZ80_HW_Config.c` file and change the default MAC address (provided by ZTP) such that each eZ80 Development Platform on the LAN contains a unique MAC address. For example:

```
const BYTE f91_mac_addr [EP_ALEN]
= {0x00, 0x90, 0x23, 0x00, 0x0F,
0x91};
```

    In the 6-byte MAC address shown above, the first three bytes must not be modified; the last three bytes can be used to assign a unique MAC address to the eZ80 Development Platform.

11. Open the `ipw_ez80.c` file. For this application, DHCP (Dynamic Host Configuration Protocol) is disabled; therefore, ensure the following:

    ```
    b_use_dhcp = FALSE
    ```

12. Open the file `snmp_conf.c` located in the path `..ZTP\SNMPDemo\`. Modify the enterprise ID and product ID numbers with values appropriate for the local network:

    ```
    #define ENTERPRISE_ID 12897

    #define PRODUCT_CODE  0x01
    ```

    Note: *The enterprise ID number 12897 refers to Zilog Inc. and is assigned and registered by IANA (Internet Assigned Numbers Authority).*

13. In the `snmp_conf.c` file, modify the tag, `char snmp_trap_target[]` to reflect the IP address and the port number of the PC-based SNMP manager where the trap messages are to be sent. By default, the port number 162 is used by the SNMP agent in ZTP.

14. In the same `snmp_conf.c` file, ensure that the variable

    ```
    Bool Generate_Enterprise_Traps = TRUE
    ```

    to enable generation of enterprise-specific custom traps.

15. Open the file `snmib.c` located in the path `..ZTP\SNMPDemo\` and add the following variable declaration:

    ```
    /* Variables used by the SNMP Demo project */

     INT8newval = 0xBB;
    ```

16. In the `snmib.c` file, locate the following structure:

    ```
    struct mib_info mib[]
    ```

    and add the following line to describe all the SNMP parameters relevant to the variable `newval` :

    ```
    {"AN0181",              "demo.",
    {{{4,1,12897,2,20,0},6}, SN_INT8,
    &newval}, TRUE,   LEAF,  snleaf,
    NULLPTR}
    ```

17. In the `snmib.c` file, modify the following lines:

    ```
    oid RFC1213_MIB[] = { 1, 3, 6, 1,
    2, 1 };

    static char Standard_Prefix[] =
    ".1.3.6.1.2.1";
    ```

    to the following:

    ```
    oid RFC1213_MIB[] = { 1, 3, 6, 1 };

    static char Standard_Prefix[] =
    ".1.3.6.1";
    ```

    to enable the SNMP manager to issue commands for all the categories of the MIB OIDs.

18. Save all the files and close the `AcclaimSNMP-Demo.pro` file.

Building the SNMP Manager

The SNMP manager used to demonstrate the SNMP application described in this Application Note is the UCD-SNMP tool suite. This suite is downloaded from the SourceForge website. The tool suite, `ucd-snmp-4.2.6.tar.gz`, has a size ~ 1.6 MB, and was last updated on 2002-10-11, at 06:14. Although the UCD-SNMP was used to test this application, any other SNMP-compliant managers can be used. Zilog does not endorse or promote specific SNMP managers.

For compiling this tool suite, Microsoft's Platform SDK is installed along with the Visual C++ IDE. The SDK is available for download from the Microsoft website. This SDK installs driver libraries for the networking APIs used by the UCD-SNMP tool suite. This SDK must be registered with Microsoft Visual Studio for the libraries to be accessible.

Note: *The SDK installation is not necessary to successfully build and execute this code; however it is used to execute the default commands available in the UCD-SNMP suite.*

Follow the steps below to compile and build the SNMP Manager:

1. Download the UCD-SNMP tool suite zipped file from the SourceForge website.

Note: *This Application Note was tested using only version 4.2.6 of the file. When using any other version, necessary changes (according to the current release) should be incorporated.*

2. Unzip the UCD-SNMP package. This creates a directory structure starting with `\ucd-snmp-4.2.6`.
3. Launch Visual C++ IDE and open the workspace file `win32.dsw` in the path `\win32`.
4. From the menu choose **Build → Batch Build** option. Uncheck the box **Selection Only** and click **Rebuild All.**

Note: *Only the SNMP manager files must build correctly and errors/warnings in the agent code can be ignored to execute this code.*

5. The compiler builds all of the executable for the UCD-SNMP project and stores them in the `\win32\bin` directory. Files with `_d` appended with the base name denote executable files with the debug option enabled.

Testing

This section contains the requirements and instructions required to setup the SNMP Demo and run it. The Demo demonstrates the exchange of messages between the eZ80F91 MCU running ZTP (with SNMP services) and a PC client running the UCD-SNMP manager on the Microsoft Windows-based TCP/IP stack.

Setup

The basic setup to assemble the Demo is displayed in Figure 2. This setup displays the connections between the PC, LAN/WAN/Internet, and the eZ80 Development Platform with the eZ80F91 Module.

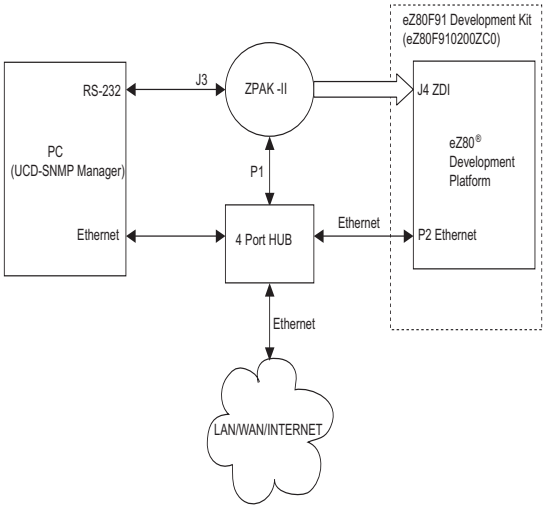

Figure 2. Setup to Test SNMP Demo

The requirements are classified under hardware and software.

Hardware Requirements

The hardware requirements are as follows:

- eZ80F91 Development Kit, which includes the following:
  - eZ80 Development Platform
  - eZ80F91 Module
  - 9 V DC Power Supply
  - ZPAKII Debug Interface Module, with power supply

  - 4-port 10 BaseT Ethernet Hub with power supply
- PC with HyperTerminal and Microsoft Visual C++ IDE

Software Requirements

The software requirements are as follows:

- Zilog ZDS II—eZ80Acclaim! (Zilog Developer Studio II for eZ80Acclaim*Plus!* devices)
- Zilog's TCP/IP Software Suite (ZTP)
- UCD-SNMP manager tool suite
- Microsoft Visual C++ IDE to build the client-side UCD-SNMP manager

Settings

HyperTerminal Settings

Set HyperTerminal to 57.6 kbps Baud and 8-N-1, with no flow control.

Jumper Settings

For the eZ80 Development Platform:

- J2 is ON
- J7, J11, J3, J20, J21, J22 are OFF
- For J14, connect 2 & 3
- For J19, CS_EX_IN is ON, MEM_CEN1, MEM_CEN2, MEM_CEN3 are OFF

For the eZ80F91 MCU:

- JP3 is OFF

Procedure for Setting Up the SNMP Demo

Follow the below procedure to setup the SNMP Demo application prior to execution:

1. Ensure that the required Demo files are added and integrated to ZTP before proceeding. For more details, see Adding and Integrating SNMP Demo Files to ZTP on page 4.
2. Make the connections as per Figure 2. Follow the jumper settings provided in the section on Jumper Settings.
3. Connect the 9 V power supply to the eZ80F91 Development Kit.
4. Connect the 5 V power supply to ZPAK II and the 7.5 V power supply to the Ethernet HUB.
5. Launch the HyperTerminal and follow the settings provided in the HyperTerminal Settings section.
6. From within the HyperTerminal, press *z* repeatedly, and then press the reset button on ZPAK II to view the menu to set the ZPAK II IP address.
7. Enter *H* to display the help menu, and follow the menu instructions to obtain the IP address for ZPAK II in order to download the Demo file. This ZPAK II IP address must be entered in the ZDS II.
8. Launch ZDS II for eZ80Acclaim! and open the Demo project file (`AcclaimSNMPDemo.pro`) located in the path: `..\ZTP\SNMPDemo`.
9. Open the `main.c` file located in the path: `..\ZTP\SNMPDemo`. Ensure that the `BootInfo` structure contains information that is relevant to your network configuration.
10. Build the AcclaimSNMPDemo project and download the resulting `snmpdemo.lod` file to the eZ80F91 MCU on the eZ80 Development Platform, using ZDS II.

Executing the SNMP Demo

Follow the steps below to execute the SNMP Demo:

1. Execute the AcclaimSNMPDemo by clicking **GO** icon in the ZDS II IDE toolbar.
2. Open the HyperTerminal console and note the IP address assigned to the eZ80F91 server (eZ80F91 MCU with the ZTP-SNMP agent). This IP address is used as the ZTP-SNMP agent's IP address.

3. Open a DOS window and use the `cd` command to change the directory location to `\ucd-snmp-4.2.6\win32 \bin`.
4. Next enter the following command at the DOS prompt:

   ```
   snmpget -Cf -c public -v 1 <eZ80F91 IP> 2.1.4.1.0
   ```

   This command is used to obtain the value of OID 1.3.6.1.2.1.4.1.0 from the eZ80F91-based ZTP-SNMP agent. Of the numbers in the OID, the SNMP manager prefixes the numbers 1.3.6.1 to the numbers supplied by the command. The command with the parameters is sent in the SNMP message format as a UDP datagram to the eZ80F91 MCU. The reply from the eZ80F91-based ZTP-SNMP agent, which is essentially a UDP packet containing an SNMP message, is parsed by the PC-based SNMP manager and is printed in the DOS window as follows:

   ```
   .iso.3.6.1.2.1.4.1.0 = 2
   ```

5. To check for the status of the variable `newval` using the custom OID 1.3.6.1.4.1.12897.2.20.0, use the following command:

   ```
   Snmpget -Cf -c public -v 1 <eZ80F91 IP> 4.1.12897.2.20.0
   ```

   The response obtained is printed in the DOS window as follows:

   ```
   .iso.3.6.1.4.1.12897.2.20.0 = 0
   ```

6. Press the switch SW1 (PB0) on the eZ80 Development Platform for 2 seconds and release it.
7. Issue the command in step 5 again to check the status of the OID; the value now changes to 1, as in:

   ```
   .iso.3.6.1.4.1.12897.2.20.0 = 1
   ```

8. To check for the trap functionality, enter the following command in the DOS window:

   ```
   snmptrapd -p 162 -T UDP -d -o ..\output.log
   ```

   This command specifies the SNMP receive port (SNMP manager) as 162, the message type as UDP, with debug option set to ON and the outputs to be redirected to a file `output.log`, besides printing the response in the DOS window. With this command, the trap daemon begins to execute on the PC-based SNMP manager; the daemon actively scans the UDP port 162 for SNMP messages.
9. Now, press the switch SW2 (PB1) on the eZ80 Development Platform. This event (switch press) creates a trap event on the eZ80F91-based ZTP-SNMP agent and transfers a SNMP trap message to the SNMP manager. The trap message is in the following format:

   ```
   Received 48 bytes from 192.168.1.1:49156

   0000: 30 2E 02 01 00 04 06 70 75
   62 6C 69 63 A4 21 06

   0016: 08 2B 06 01 04 01 E4 61 01
   40 04 AC 10 06 C6 02

   0032: 01 06 02 01 00 43 02 5A 61
   30 05 30 03 02 01 00
   ```

   Here, only the HEX values of the message are shown. The last byte (`00`) contains the current value of the variable `newval`, which can be toggled by pressing the switch SW2 (PB1).

   Note: *SW1 is pressed to toggle the value between `00` and `01`, and SW2 is pressed to view the changes in the command window.*

10. To exit from the program, enter `reboot` in the HyperTerminal window and close all running programs.

Summary

The implementation of SNMP v1 protocol in Zilog's TCP/IP stack provides an easy way for querying and logging the status of Internet-enabled products in a managed network. This Application Note helps in initiating a product designer to use SNMP for application-specific uses. The included demonstration presents a practical approach towards understanding the SNMP protocol.

References

The documents associated with eZ80®, eZ80Acclaim!®, and eZ80Acclaim*Plus!*™ family of products are listed below:

- Zilog TCP/IP Software Suite Programmer's Guide Reference Manual (RM0008)
- eZ80 CPU User Manual (UM0077)
- eZ80F91 MCU Product Specification (PS0192)
- Zilog Developer Studio II—eZ80Acclaim! User Manual (UM0144)
- SNMP Protocol RFC 1157, the full text of the RFC 1157 is available at the URL http://www.ietf.org/rfc/rfc1157.txt?number=1157
- List Of Private Enterprise MIBs, available at the URL http://www.iana.org/assignments/enterprise-numbers
- Non-commercial version of the SNMP manager for Windows systems, available at the URL http://www.net-snmp.com/COPYING.txt
- UCD-SNMP, downloaded from the website: http://www.net-snmp.org

Appendix A—Flowcharts

This appendix contains the flowchart for the SNMP application described in this Application Note. The flowchart for the SNMP agent application is displayed in Figure 3.

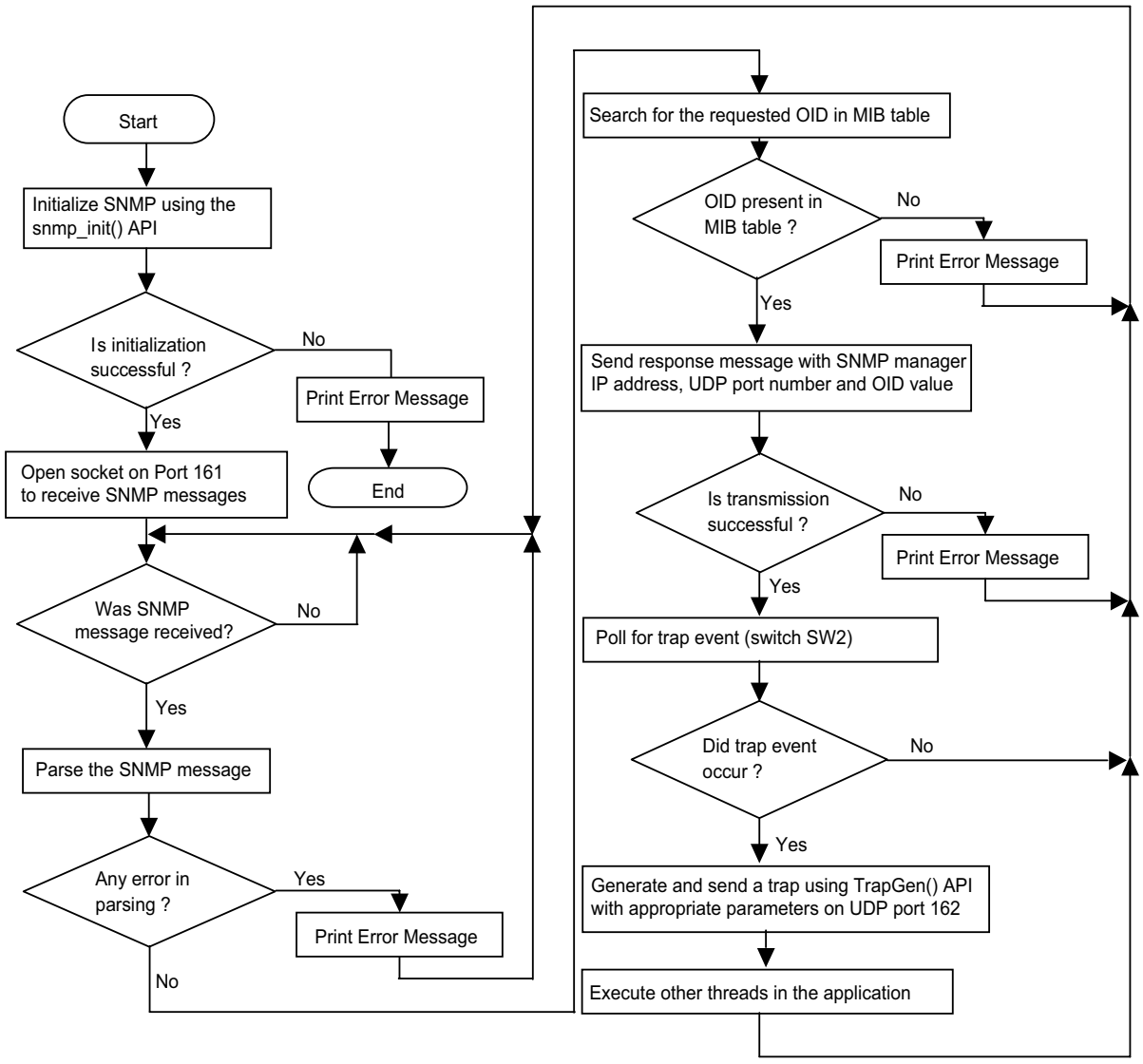

Figure 3. Flowchart for SNMP Agent Application

Warning: DO NOT USE IN LIFE SUPPORT

LIFE SUPPORT POLICY

ZILOG'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS PRIOR WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF ZILOG CORPORATION.

As used herein

Life support devices or systems are devices which (a) are intended for surgical implant into the body, or (b) support or sustain life and whose failure to perform when properly used in accordance with instructions for use provided in the labeling can be reasonably expected to result in a significant injury to the user. A critical component is any component in a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system or to affect its safety or effectiveness.

Document Disclaimer