
Application Note: Private MIBs for NET+OS x.x

TABLE OF CONTENTS

TABLE OF CONTENTS	ii
Overview	1
Requirements	1
Documentation	1
Qualify your resources	2
BUILD NAMIB	2
DOWNLOAD AND RUN NAMIB APPLICATION	2
COMPILE THE SAMPLE NAMIB MIB	2
Steps to Build and Test the Sample MIB Application	2
WALK THE MIB	4
ADDING A SECOND MIB	5
CONFIGURATION AND TOOLS MODIFICATIONS	5
gen.bat	5
project.bld	6
\mibs\listgh	6
otherleds.config	6
otherleds.inc	6
\mibs\otherleds.sm2	6
Building NAMIB with a second MIB	7
OTHERLEDS.SM2	8
otherleds.config	10
ADD SPECIFICATIONS TO OTHERLEDS, CONFIG	10
RECREATE THE OTHERLEDSACTION.C SOURCE FILE	10
ACTION ROUTINE SOURCE MODIFICATION	11
Add New MIB to Database and Verify	11
PROJECT BUILD AND DEBUG	12
Verify the OTHERLEDS MIB in MIB Browser	12
SAMPLE FILES	12

OVERVIEW

These tools support the integration of MIBs (Management Information Bases) in the NET+OS environment:

- SMICng
- MIBMAN

SMICng is the SNMP MIB Information Compiler. SMICng checks MIB modules at the level of strictness you specify through command line switches, compiler directives, and environment variable options. The output of SMICng is an intermediate file that is ready for processing by MIBMAN.

MIBMAN is a utility that translates Simple Network Management Protocol (SNMP) Management Information Bases (MIBs) into C code that contains:

- Templates for action routines that you implement
- Management API declarations for MIB objects that correspond to management variables.

You invoke these tools through a script named 'gen.bat'.

This document describes and demonstrates how to integrate two private MIBs into a NET+OS 6.0 application. This document also identifies the variables, tables, and interfaces you create with MIBMAN that are exposed through the Management API.

The information is presented through a sample application that uses two private MIBs to manage some common NetARM resources (GPIO PortC LEDs) on the 7520 development board.

REQUIREMENTS

Knowledge of SNMP in operation and the ability to use a MIB browser and compiler.

Knowledge of C programming and the ability to create an application for NET+OS as well as the capability to debug the application.

A NET+OS development system with a Green Hills compiler and debugger, MIBMAN, and SMICng.

A MIB browser and compiler.

DOCUMENTATION

SMICng – usesmic.htm (provided on the NET+Works CD)

MIBMAN – userguide_programmerguide_ghs.pdf – 'Using the MIBMAN Utility' (provided both on the NET+Works CD and in hard copy)

Readme document – located in the sample application root directory

'Understanding SNMP MIBs' ISBN – 0-13-437708-7 available at Amazon.com

QUALIFY YOUR RESOURCES

You must be able to:

Build, download, and execute the sample application 'namib'.

Display and set variables in the sample application's MIB with a Web browser. For instructions about your MIB browser and compiler, see the documentation you received with them.

The next sections outline a method of qualifying your development kit and software by using the sample 'namib' application.

Build namib

For instructions about how to build, download, and debug your application, see the *NET+OS with Green Hills Programmer's Guide..*

Download and Run namib application

For instructions about how to build, download, and debug your application, see the *NET+OS with Green Hills Programmer's Guide..*

. The message 'The example MIB has been loaded.' will appear in the console or HyperTerm window.

Compile the sample namib MIB

Your MIB browser requires a MIB database from which to define the operations and objects that are available to SNMP on a particular MIB. To prepare the MIB, you need to run the MIB compiler on the MIB source files.

The files for this example are located in the (PROJECT_ROOT)\mibs directory, where (PROJECT_ROOT) is the directory that contains the project.bld file (default is: C:\NETOS60_GH361\src\examples\namib).

After you compile (create the database files), you need to register the new MIB with your browser. For instructions about compiling and registering the MIB, see the documentation you received with your MIB browser/compiler.

After you compile and register the MIB, you are ready to use the MIB browser to verify the sample application's MIB.

Steps to Build and Test the Sample MIB Application

To build and test the sample namib application, first

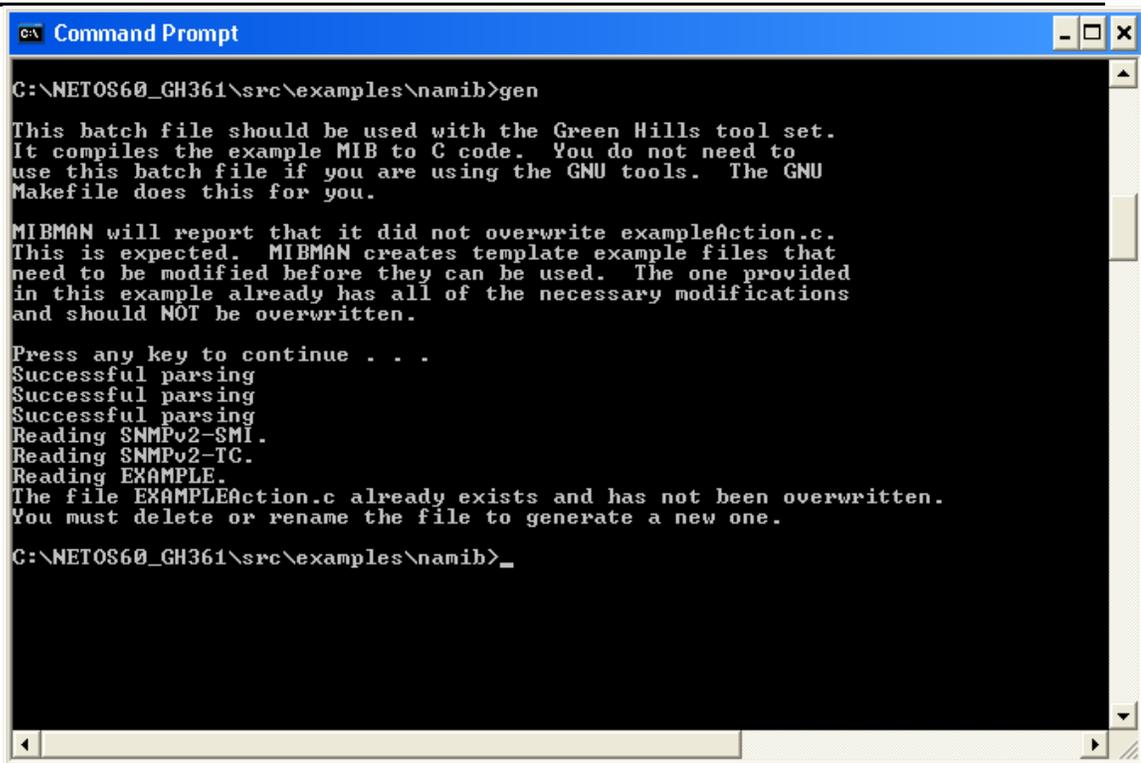
1. Open a DOS Window and set the current directory to the root directory of the namib application; for example:

```
C:\NETOS60_GH361\src\examples\namib
```
2. Verify that these files are present in the directory:
 - Gen.bat

- Mibman.jar
- Project.bld (Green Hills)
- Readme
- Root.c
- Table.c
- Table.h

(For a description of these files, see the Readme file.)

3. Verify that you have Java in your path. If you need to install the Java runtime environment, go to <http://www.sun.com> for the latest supported version.
4. Verify that the \mibs subdirectory exists and contains these files:
 - Example.inc
 - Listgh
 - Example.config
 - Example.sm2
 - Rfc1902.sm2
 - Rfc1903.sm2
 - Rfc1904.sm2
5. Run the gen command (batch file). output of the session looks similar to this



```
C:\> Command Prompt
C:\NETOS60_GH361\src\examples\namib>gen

This batch file should be used with the Green Hills tool set.
It compiles the example MIB to C code. You do not need to
use this batch file if you are using the GNU tools. The GNU
Makefile does this for you.

MIBMAN will report that it did not overwrite exampleAction.c.
This is expected. MIBMAN creates template example files that
need to be modified before they can be used. The one provided
in this example already has all of the necessary modifications
and should NOT be overwritten.

Press any key to continue . . .
Successful parsing
Successful parsing
Successful parsing
Reading SNMPv2-SMI.
Reading SNMPv2-TC.
Reading EXAMPLE.
The file EXAMPLEAction.c already exists and has not been overwritten.
You must delete or rename the file to generate a new one.

C:\NETOS60_GH361\src\examples\namib>_
```

6. Take note of several new files that appear in the namib directory. Mibman.h, mibman.c, example.h, example.c are all outputs of this process.
7. Note, in the \mibs directory, several files are produced: rfc1902.out, rfc1903.out, and example.out.
8. The next step is to compile the MIB(s) for your MIB browser. The tools you use for this step determine the procedure for compiling and registering the MIB.
9. Build the project either in the Green Hills IDE or by invoking the Green Hills Build tool on the command line ("build image.bld" from the \32b directory).
10. Open the terminal (HyperTerm) window for COM1, download the image, and execute the namib application.
11. Start the MIB browser after the application has started.

For the exact steps for contacting your target and retrieving information from the SNMP agent, see the documentation you received with your MIB browser.

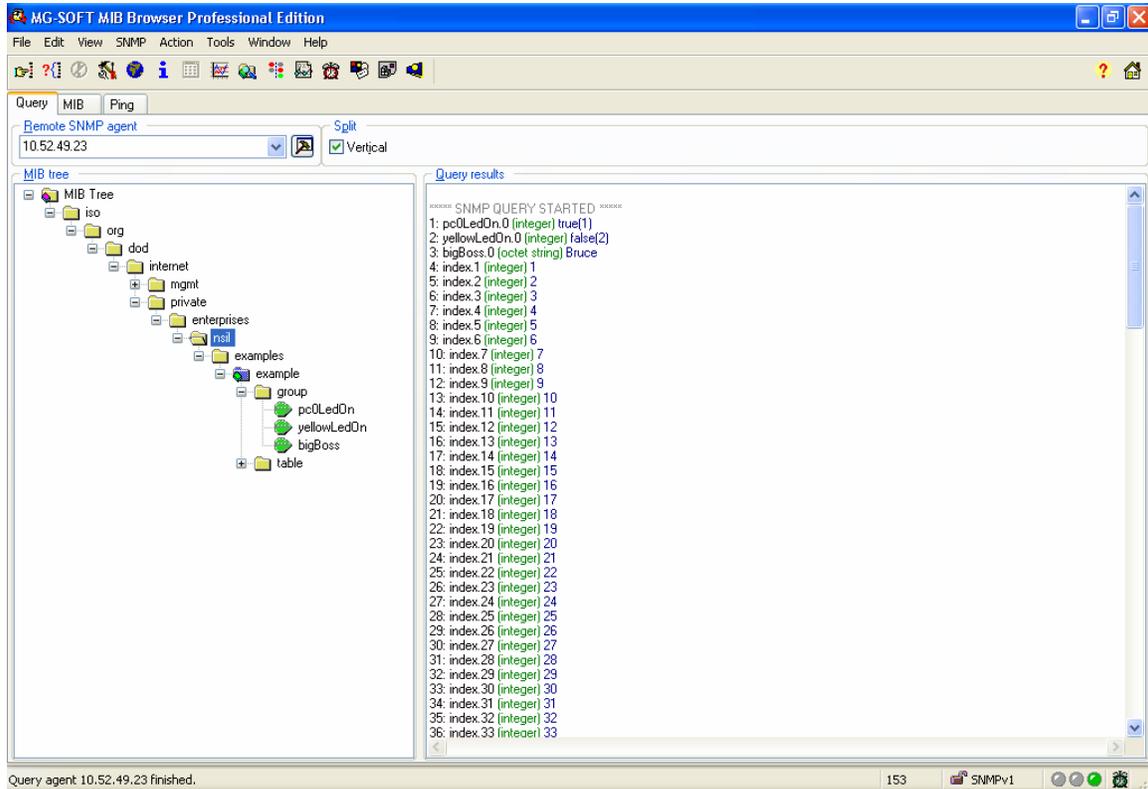
Walk the MIB

First, you must contact the SNMP agent running in your application. Your MIB browser provides the means for testing this connection.

After you establish contact, you can examine the contents of the application's MIB.

Next is a screen capture of the basic namib MIB. This output was generated by performing a 'walk' with the selection on 1.3.6.1.4.1.901 (iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).nsil(901)).

When you can do this (retrieve the objects' identifiers and values), you have a working application with an SNMP agent running and a properly compiled MIB.



Adding a second MIB

We will add a MIB to the existing application.

Configuration and Tools Modifications

gen.bat

You must modify the gen.bat file to process the additional source (otherleds.sm2). Add a line, as shown here:

```
..\..\bin\smicng -z -cm .\mibs\otherleds.inc > .\mibs\otherleds.out
```

Insert this code just before this line

```
java -jar ..\..\bin\mibman.jar mibs\listgh mibs
```

project.bld

Modify the project.bld file) to include the following lines

```
OTHERLEDS.c
```

```
    C
```

```
OTHERLEDSAction.c
```

```
    C
```

\mibs\listgh

This line is appended to \mibs\listgh:

```
mibs\otherleds.out
```

This step appends otherleds.out in the MIBMAN processing.

otherleds.config

Add to the directory \mibs, the file otherleds.config. It contains three lines that cause MIBMAN to create action routines. This is the entirety of the file's content.

```
GenerateWriteActionRoutine 1.3.6.1.4.1.901.1000.1.1.1
```

```
GenerateWriteActionRoutine 1.3.6.1.4.1.901.1000.1.1.2
```

```
GenerateWriteActionRoutine 1.3.6.1.4.1.901.1000.1.1.3
```

otherleds.inc

Add otherleds.inc to the directory \mibs. This is identical to example.inc except for this line:

```
#condInclude "otherleds.sm2"
```

which replaces:

```
#condInclude "example.sm2".
```

\mibs\otherleds.sm2

This is the source file for the second mib 'otherleds' that is added to the project.

The second MIB in this example is called OTHERLEDS. Its source is otherleds.sm2.

The .inc file necessary for this otherleds.inc. is virtually a copy of example.inc except for this line:

```
"#condInclude "otherleds.sm2"
```

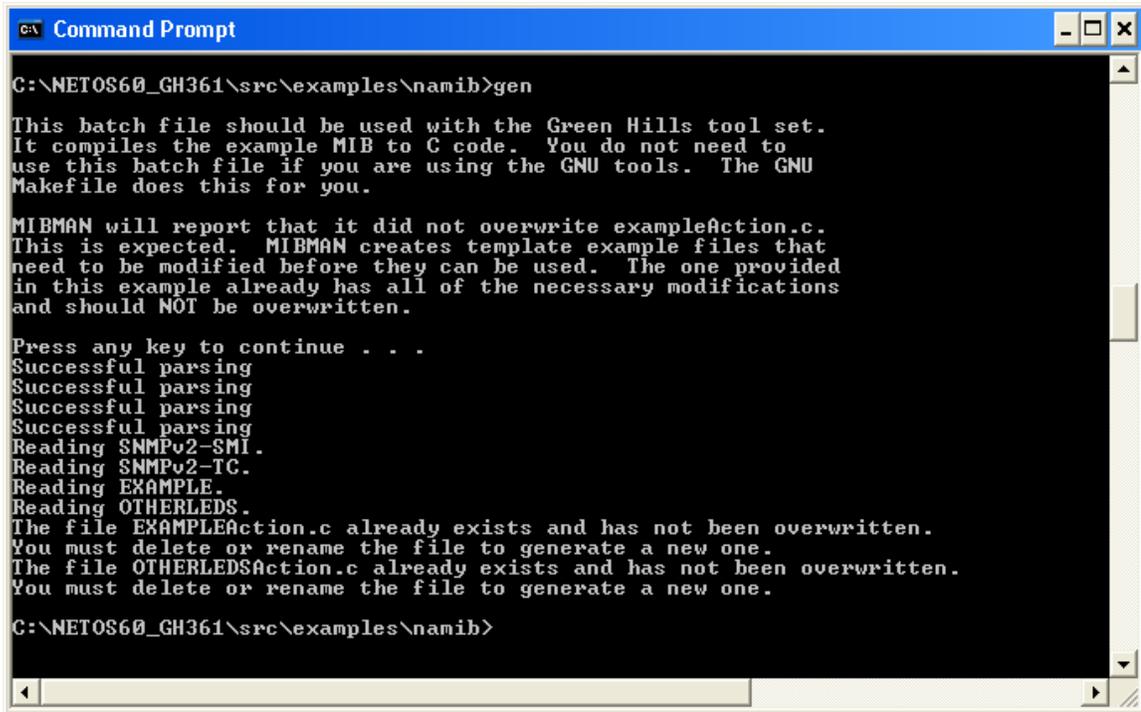
which replaces:

```
"#condInclude "example.sm2"
```

To add OTHERLEDS to this project, you need to modify the:Gen.bat file. Add a line to include the new MIB source: ..\..\..\bin\smicng -z -cm .\mibs\otherleds.inc > .\mibs\otherleds.out.

BUILDING NAMIB WITH A SECOND MIB

1. Run the gen command (batch file). The output of the session will look like this:



```
C:\NETOS60_GH361\src\examples\namib>gen
This batch file should be used with the Green Hills tool set.
It compiles the example MIB to C code. You do not need to
use this batch file if you are using the GNU tools. The GNU
Makefile does this for you.

MIBMAN will report that it did not overwrite exampleAction.c.
This is expected. MIBMAN creates template example files that
need to be modified before they can be used. The one provided
in this example already has all of the necessary modifications
and should NOT be overwritten.

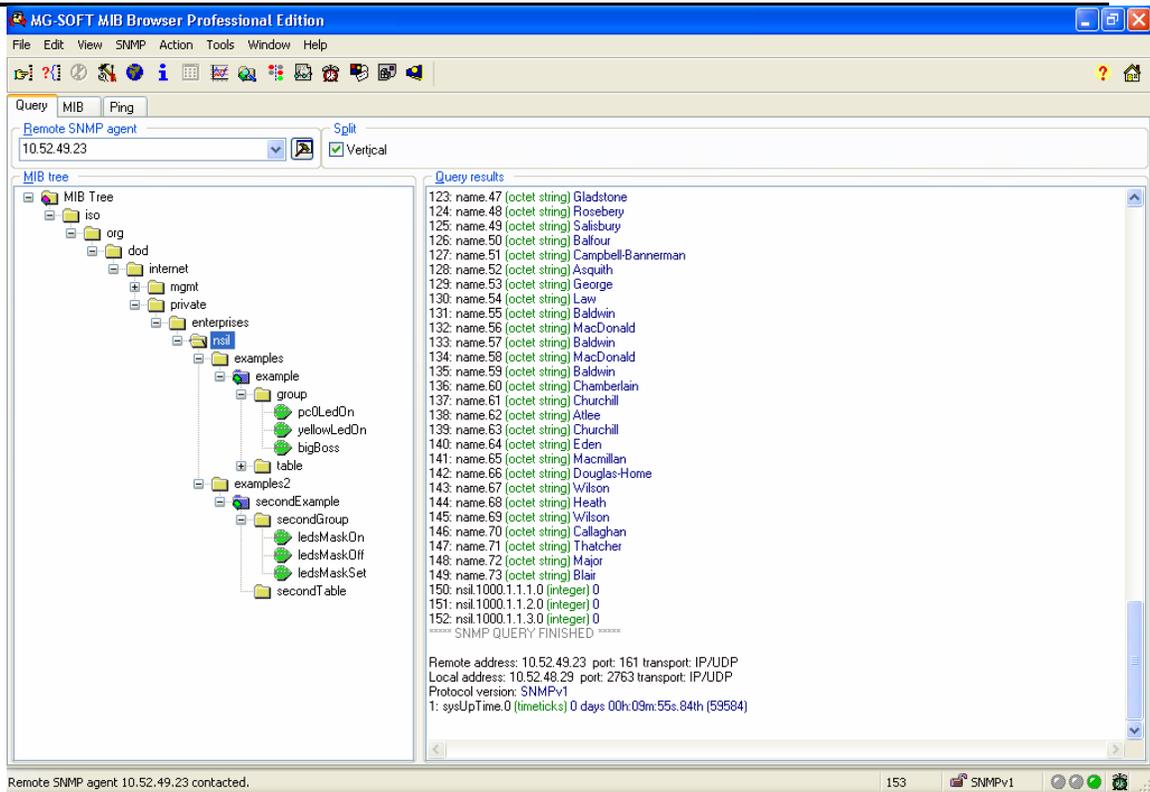
Press any key to continue . . .
Successful parsing
Successful parsing
Successful parsing
Successful parsing
Reading SNMPv2-SMI.
Reading SNMPv2-TC.
Reading EXAMPLE.
Reading OTHERLEDS.
The file EXAMPLEAction.c already exists and has not been overwritten.
You must delete or rename the file to generate a new one.
The file OTHERLEDSAction.c already exists and has not been overwritten.
You must delete or rename the file to generate a new one.

C:\NETOS60_GH361\src\examples\namib>
```

Note the additional statement from MIBMAN 'Reading OTHERLEDS' and the statement that OTHERLEDSAction.c already exists.

2. Next, rebuild the application in the Green Hills IDE (or at the command line by invoking the builder from \32b – build image.bld).
3. Compile the new MIB, and register it with your MIB browser.
4. Download and start the application.
5. Start the MIB browser when the application has started.

Your new MIB should resemble this:



OTHERLEDS.SM2

The first three items in the file otherleds.sm2 are the Name, Imports statement and, MODULE-IDENTITY. For information about the format of the MIB module and the syntax of the statements, see 'Understanding SNMP MIBs'. The first two chapters describe the format of the MIB specification, provide some history, and explain the organization of the IOD. (That's more than enough to get started.)

The next statements set some important information that identifies this object. First is the nsil object identifier (common with example.sm2), followed by identities for 'examples2', 'secondGroup and secondTable. (differentiate this object and its components from 'example').

```
examples2 OBJECT IDENTIFIER ::= {nsil 1000 }
```

```
-- scalar items
```

```
-- Now define a group and a table
```

```
secondGroup OBJECT IDENTIFIER ::= {secondExample 1}
```

```
secondTable OBJECT IDENTIFIER ::= {secondExample 2}
```

-- scalar items

ledsMaskOn OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This object is used to turn on multiple LEDs"

DEFVAL {0}

::= {secondGroup 1}

ledsMaskOff OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This object is used to turn off multiple LEDs"

DEFVAL {0}

::= {secondGroup 2}

ledsMaskSet OBJECT-TYPE

SYNTAX Integer32

MAX-ACCESS read-write

STATUS current

DESCRIPTION

"This object is used to turn on and off multiple LEDs"

DEFVAL {0}

::= {secondGroup 3}

END

OTHERLEDS.CONFIG

The file specifies some needed action routines to MIBMAN. MIBMAN is the second tool that the gen.bat file invokes.

Defining the ledsMaskOn, ledsMaskOff and ledsMaskSet objects in the otherleds.sm2 file gave MIBMAN enough information to define some variables and structures (for the management interface, among others) to implement these objects. MIBMAN also creates action routines for the variables.

One of the messages that appears in the console window when running the batch file is:

“The file xxxxxxAction.c already exists and has not been overwritten.” This feature protects your action routine sources from being replaced by the template MIBMAN creates for these variables. However, we need the templates that MIBMAN creates, so we will rebuild (after renaming the existing OTHERLEDSAction.c). After rebuilding, we can integrate whatever code we wanted from the original source (OTHERLEDSAction.c-save) with the newly-generated template OTHERLEDSAction.c.

MIBMAN uses the contents of otherleds.config to determine which action routines to generate. We want action routines for all the objects (ledsMaskOn, ledsMaskOff and ledsMaskSet) to be able to set the values of these variables through the MIB browser.

Add specifications to otherleds.config

The identification of the routines by their target objects requires us to look in the file otherleds.h to retrieve the OIDs for these objects.

```
#define OTHERLEDS_ledsMaskOn "1.3.6.1.4.1.901.1000.1.1.1"  
#define OTHERLEDS_ledsMaskOff "1.3.6.1.4.1.901.1000.1.1.2"  
#define OTHERLEDS_ledsMaskSet "1.3.6.1.4.1.901.1000.1.1.3"
```

are present.

Open the otherleds.config file and verify that the statements:

```
GenerateWriteActionRoutine 1.3.6.1.4.1.901.1000.1.1.1  
GenerateWriteActionRoutine 1.3.6.1.4.1.901.1000.1.1.2  
GenerateWriteActionRoutine 1.3.6.1.4.1.901.1000.1.1.3
```

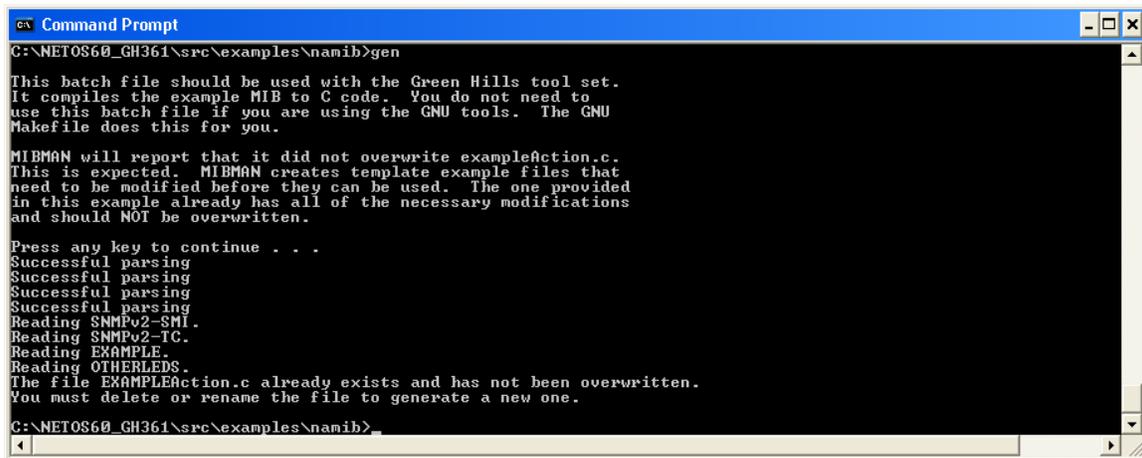
are present.

Recreate the OTHERLEDSAction.c source file

Next, either rename the OTHERLEDSAction.c file or delete it to allow MIBMAN to generate the action routine templates on the next pass.

“Execute the gen.bat command again, and then examine the contents of OTHERLEDSAction.c. You will find an action routine for each of the two statements in the otherleds.config that we just entered.

The output displayed is:



```

C:\NETOS60_GH361\src\examples\namib>gen
This batch file should be used with the Green Hills tool set.
It compiles the example MIB to C code. You do not need to
use this batch file if you are using the GNU tools. The GNU
Makefile does this for you.

MIBMAN will report that it did not overwrite exampleAction.c.
This is expected. MIBMAN creates template example files that
need to be modified before they can be used. The one provided
in this example already has all of the necessary modifications
and should NOT be overwritten.

Press any key to continue . . .
Successful parsing
Successful parsing
Successful parsing
Successful parsing
Reading SNMPv2-SMI.
Reading SNMPv2-TC.
Reading EXAMPLE.
Reading OTHERLEDS.
The file EXAMPLEAction.c already exists and has not been overwritten.
You must delete or rename the file to generate a new one.
C:\NETOS60_GH361\src\examples\namib>

```

Action Routine Source Modification

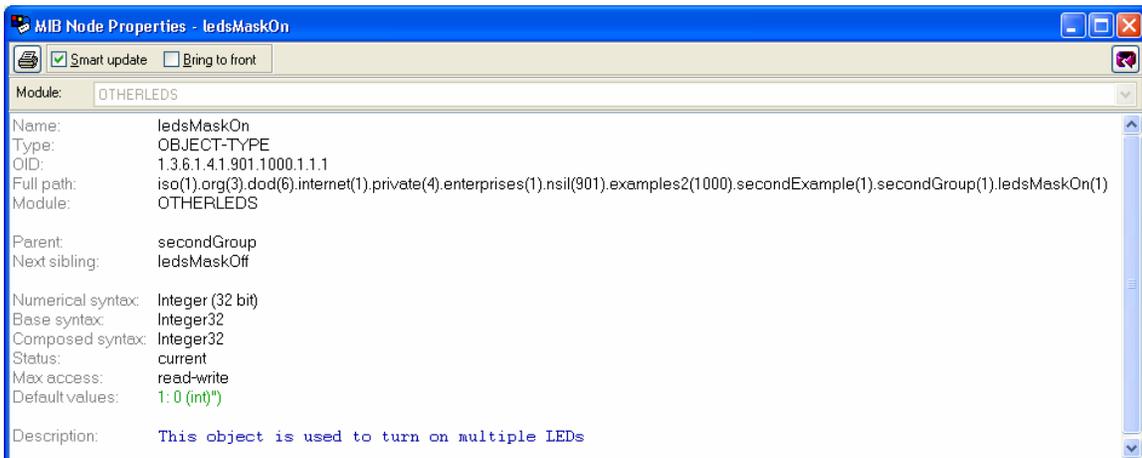
The action routines that MIBMAN generatesy MIBMAN are stubs. You need to add code to these routines to perform the needed work. An example of turning the yellow LED on and off is available in the EXAMPLEAction.c file.

ADD NEW MIB TO DATABASE AND VERIFY

See the section: “Compile the sample namib MIB” and the documentation you received with the MIB Compiler and browser for instructions about compiling and registering the new MIB.

After the OTHERLEDS MIB has been registered, you can browse its contents.

Here is a sample MIB browser output that describes the ledsMaskOn variable’s properties:



```

MIB Node Properties - ledsMaskOn
[Smart update] [Bring to front]
Module: OTHERLEDS
Name: ledsMaskOn
Type: OBJECT-TYPE
OID: 1.3.6.1.4.1.901.1000.1.1.1
Full path: iso(1).org(3).dod(6).internet(1).private(4).enterprises(1).nsil(901).examples2(1000).secondExample(1).secondGroup(1).ledsMaskOn(1)
Module: OTHERLEDS
Parent: secondGroup
Next sibling: ledsMaskOff
Numerical syntax: Integer (32 bit)
Base syntax: Integer32
Composed syntax: Integer32
Status: current
Max access: read-write
Default values: 1: 0 (int)
Description: This object is used to turn on multiple LEDs

```

Project Build and Debug

You need to add the new C sources that were created to your project's build files.

- Navigate to the 'project.bld' file. From the 'Project' drop-down menu, select 'Add File to Project.Bld'. Add the files OTHERLEDS.c and OTHERLEDSAction.c files to the project.
- Navigate to the 'image.bld' file and perform a build (Rebuild All is not necessary).
- After successfully building, you can download and run the application.

VERIFY THE OTHERLEDS MIB IN MIB BROWSER

After you install or register the newly-compiled MIB with the MIB browser and start the application running, you can use the MIB browser to interact with the agent running in your target.

At this point, it should be easy to verify the OIDs, types and names of all the variables.

To see the OID for each variable, see the generated .c and .h files (EXAMPLE.c, EXAMPLE.h, OTHERLEDS.c, OTHERLEDS.h).

SAMPLE FILES

NetSilicon recommends that you save a copy of your working source code tree before you make these changes.

Extract the contents of the zip file into the project root directory with these options selected:

- All files
- Overwrite existing files

These files contain all the edits described above.

The code, MIB specifications, and configuration files for this sample are provided in two zip files.

Unzip the [netos60-with-private-mibs-rev0.1.zip](#) file into your project root directory; for example: C:\NETOS60_GH361\netsilicon_netos_source\netos\src\examples\namib.

Additionally, to support the Yellow LED and pc0 LED on and off calls, you need to extract the contents of: [netos60-with-private-mibs-bsp-mods.zip](#) into your platform bsp directory. The file is gpio.c. Find the directory in your source tree where that file is located, and replace it with the gpio.c in the zip file then rebuild the bsp before rebuilding the application.