EMANATE®/Lite SNMP for OS-9

Version 15.3

www.radisys.com

World Headquarters
5445 NE Dawson Creek Drive • Hillsboro, OR
97124 USA
Phone: 503-615-1100 • Fax: 503-615-1121
Toll-Free: 800-950-0044

International Headquarters
Gebouw Flevopoort • Televisieweg 1A
NL-1322 AC • Almere, The Netherlands
Phone: 31 36 5365595 • Fax: 31 36 5365620

RadiSys Microware Communications Software Division, Inc.
1500 N.W. 118th Street
Des Moines, Iowa 50325
515-223-8000

Revision F
December 2001

Copyright and publication information

This manual reflects version 15.3 of EMANATE®/Lite SNMP for OS-9 from SNMP Research, Inc. Reproduction of this document, in part or whole, by any means, electrical, mechanical, magnetic, optical, chemical, manual, or otherwise is prohibited, without written permission from RadiSys Microware Communications Software Division, Inc.

Disclaimer

The information contained herein is believed to be accurate as of the date of publication. However, RadiSys Corporation will not be liable for any damages including indirect or consequential, from use of the OS-9 operating system, Microware-provided software, or reliance on the accuracy of this documentation. The information contained herein is subject to change without notice.

Reproduction notice

The software described in this document is intended to be used on a single computer system. RadiSys Corporation expressly prohibits any reproduction of the software on tape, disk, or any other medium except for backup purposes. Distribution of this software, in part or whole, to any other party or on any other system may constitute copyright infringements and misappropriation of trade secrets and confidential processes which are the property of RadiSys Corporation and/or other parties. Unauthorized distribution of software may cause damages far in excess of the value of the copies involved.

December 2001

Table of Contents

Chapter 1: Getting Started

This chapter describes using the Microware implementation of EMANATE®/Lite SNMP from SNMP Research International, Inc. It includes the following sections:

- **Introduction**
- **Configuring the Target System**

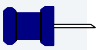
RadiSys.
MICROWARE SOFTWARE

Introduction

EMANATE®/Lite SNMP for OS-9 is third-party software from SNMP Research International, Inc., that Microware has a license to resell. Source files for examples using EMANATE/Lite are included.

Documentation from SNMP Research International, Inc., is included on your Enhanced OS-9 CD.

Configuring the Target System



Note

Your target system must be running Microware OS-9 SoftStax and the LAN Communications Pak in order to run EMANATE®/Lite SNMP for OS-9.

To configure your target system for EMANATE/Lite for OS-9, you must include the appropriate module as part of the OS-9 boot in your ROM image.

EMANATE/Lite agent provides MIB-II capabilities, and subagent extensions. However, in order to add an extension subagent, you have to have access to the source code, and compile and link in the new extensions to the agent. You only need to include one module in your ROM image:

- `snmpd` - EMANATE/Lite agent

Use the OS-9 Wizard during the configuration process to include the binaries by completing the following steps.

Step 1. From the Wizard main configuration window, in Advanced Mode, select `Configure -> Bootfile -> Network Configuration`.

Step 2. Select **SoftStax Options** Tab

Step 3. Select the SNMP check box.

Step 4. Click `OK`.

After all of the above procedure has been performed, you need to include only **one** of the following configuration files on a disk on the target system:

- `v1.cnf` if you want to run SNMPv1-only agent

- `v1c.cnf` if you want to run SNMPv1 and SNMPv2c agent (Also known as “bilingual” version)
- `v1c3.cnf` if you want to run SNMPv1, SNMPv2c and SNMPv3 agent (Also known as “trilingual” version)

The files are located in the following directory on the host side:

`/MWOS/SRC/SPF/SNMPLITE/EXPORT/CONFIG/AGT`

This file is used to configure the EMANATE/Lite agent for the target system. You can modify the file as needed.

You must place the file in the following location on the target:

`/dd/ETC/SRCONF/AGT`

You can also put the agent configuration file in another directory. However, you must set the `SR_AGT_CONF_DIR` environment variable in your target to match the directory where the file is stored. For example, if you want to put the file in the `/dd/SNMP/AGT` directory, you have to set the `SR_AGT_CONF_DIR` environment variable as shown below:

```
setenv SR_AGT_CONF_DIR /dd/SNMP/AGT
```

so the agent can figure out the location of the configuration file.

After you place the appropriate configuration file, copy or rename the file to `snmpd.cnf`, so the agent can see it and run appropriately.

WARNING

Failing to include the `snmpd.cnf` file on the target will result in an error when you try to run the EMANATE/Lite agent. Also, don’t forget to put a `/dd` alias on the disk that contains the file. An example is:

```
alias /dd /h0
```

A fixed drive, such as a hard disk, is preferred to the RAM disk, since it can maintain the data across a reboot.

After you transfer all the necessary files to the target, start the EMANATE/Lite agent by entering the following command at the shell prompt:

```
snmpd <>>>/nil&
```

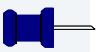
Chapter 2: MIB-II Subagent Extensions

This chapter introduces some example MIB subagent extensions that can be added to the basic EMANATE/Lite master agent. These subagent extensions are designed to support logical groups of MIB variables, including an entire MIB, selected MIB families, or single MIB variables.

Overview of MIB-II Subagents

The format of the agents information has been standardized. Each agent contains the following headings and corresponding information, if applicable:

Extension Name	Identifies the MIB-II subagent extension name.
Description	Presents a narrative detailing the uses, features, and specific instructions for each subagent.
Example Patch	Presents a guide on how to make the necessary adjustment to build each subagent extension properly.



Note

The **Example Patch** section only serves as an example, and it will not give you real-time data as in the real SNMP agents. You have to build your own retrieval functions to get the data you want.

See Also	Provides a reference to related subagent(s).

For More Information

Refer to SNMP Research Inc.'s *SNMP Research OID Configuration* and *SNMP Research EMANATE®/Lite Developer Documentation* documentations for more detailed information regarding how the subagents operate.

Recompiling Subagent Binaries

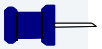
The EMANATE/Lite agents with extensions source code can be found in the following directory:

`/MWOS/SRC/SPF/SNMPLITE/SRC/SNMPD/EXTEND/EXAMPLES`

When an extension subagent is added, the compiler will build a new EMANATE/Lite agent binary with the extension name following. For example, if you add the Temperature subagent extension, the compiler will build a `snmpd_thrm` binary. This binary will have all the basic EMANATE/Lite capabilities, plus the subagent capabilities.

WARNING

You **cannot** have two agents running at the same time on the target. If you accidentally load two agents, the first agent will be used. For example, if you load `snmpd` and `snmpd_thrm` at the same time, `snmpd` will be used, while `snmpd_thrm` is ignored.



Note

To recompile the binaries for the agents and other SNMP binaries (e.g. management utilities), use the `bmake` utility instead of `os9make`. The utility is included in your Enhanced OS-9 CD. Following is an example of using the `bmake` utility:

```
bmake -f Makefile.os9 TARGET=pppc
```

You can also uncomment the desired TARGET in the `target.tpl` file, located in the `/MWOS/SRC/SPF/SNMPLITE/SRC` directory.

Note

Current targets supported for EMANATE/Lite include the following: `k68k`, `p386`, `pppc`, `arm4`, `sh`, `sh4` and `mip32`.

WARNING

When recompiling the subagent binaries for EMANATE/Lite, the first run of the compile **always** fails. This is not a bug, but is a result of some variables that have not been initialized yet (Because of the `-DNOT_YET` define in the makefiles). You have to supply your own value for the variables. Refer to the **Example Patch** section of each agent below for information about the variables. After the variables are initialized your recompile will build properly.

In addition, during the unsuccessful first compile attempt, a warning will appear and inform you to modify the necessary file to make it compile properly.

When compiling, the compiler generates a data file that contains all the information related to the subagents (for example the Chronometer MIB generates a `chroinfo.dat`). The compiler then merges the data file into the default `snmpinfo.dat` file, which is located in the following directory:

`/MWOS/SRC/SPF/SNMPLITE/EXPORT/CONFIG/MGR`

The `snmpinfo.dat` file enables the management utilities (for example: `getmany`, `getone`) to read from and write to the agent. If `snmpinfo.dat` does not exist, an error will occur when you try to use the utilities. You can remake a new `snmpinfo.dat` file by executing the Makefile in the following directory:

`/MWOS/SRC/SPF/SNMPLITE/SRC/MIBS/COMMON`

chro

Chronometer MIB

Name

`snmpd_chro`

Description

The Chronometer MIB demonstrates the `DateAndTime` scalar object types. It describes the management information for a timekeeping device, such as a clock or computerized appointment book application.

Example Patch

You need to add the patch to the following file to make the agent properly:

`CHRO/RELS/<processor>/<version>/k_chro.c`

The code snippet below shows an example of adding the patch. The variable value can be modified as appropriate. The bold parts are the example variable values.

```
...
chronometer_t *
k_chronometer_get(int serialNum, ContextInfo *contextInfo,
                  int nominator)
{
#ifdef NOT_YET
   static chronometer_t chronometerData;
...
   chronometerData.chronometerCurrentTime = MakeOctetStringFromText("18:03:24”);
   SET_ALL_VALID(chronometerData.valid);
   return(&chronometerData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
```

cmbo

Combo MIB

Name

snmpd_cmbo

Description

The Combo MIB is a combination of the Thermometer and Thermostat MIBs. It lives in SNMPD/EXTEND/COMBO/THRMTHST directory.

Example Patch

You need to add the patch to the following file to make the agent properly:

COMBO/RELS/<processor>/<version>/k_cmbo.c

Refer to the thrm and thst section for the appropriate patch.

log

Log Output MIB

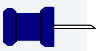
Name

`snmpd_log`

Description

The Log MIB demonstrates SNMP Research Library support for log messages.

SNMP Research programs can generate log messages with specific attributes. The Log Output MIB describes the management information for the output of the SNMP Research log messaging system.



Note

To compile this agent, you have to include the `DEBUG` define in the makefile, otherwise, this agent will not compile properly.

Example Patch

You need to add the patch to the following file to make the agent properly:

`LOG/RELS/<processor>/<version>/k_logo.c`

For this example, you only need to copy the `k_logo.fin` from the `log` directory as `k_logo.c` in the `RELS/<processor>/<version>` directory.

Name

`snmpd_mirr`

Description

The Mirror MIB demonstrates how to retrieve a MIB object automatically from elsewhere in the agent. It demonstrates SNMP Research library function `GetVar()`, which can be found in section 14.4 of the *EMANATE®/Lite Developer Documentation* documentation. The Mirror MIB is structures, therefore the use of `GetVar()` can be explained and the potential pitfalls exhaustively explored.

Example Patch

Add the patch to the following file to make the agent properly:

`MIRROR/RELS/<processor>/<version>/k_mirr.c`

The code snippet below shows an example of adding the patch. The variable value can be modified as appropriate. The bold parts are the example variable values.

```
...
mirrorLoopbackTargetBefore_t *
k_mirrorLoopbackTargetBefore_get(int serialNum, ContextInfo *contextInfo,
                                 int nominator)
{
#ifdef NOT_YET
    static mirrorLoopbackTargetBefore_t mirrorLoopbackTargetBeforeData;
...
    mirrorLoopbackTargetBeforeData.mirrorLoopbackInteger1 = 18;
    mirrorLoopbackTargetBeforeData.mirrorLoopbackOctetString1 =
MakeOctetStringFromText("Microware OS-9");
    mirrorLoopbackTargetBeforeData.mirrorLoopbackOID1 = MakeOIDFromDot("1.3.6.1");
    SET_ALL_VALID(mirrorLoopbackTargetBeforeData.valid);
    return(&mirrorLoopbackTargetBeforeData);
#else /* NOT_YET */
    return(NULL);
#endif /* NOT_YET */
}

mirrorSource_t *
k_mirrorSource_get(int serialNum, ContextInfo *contextInfo,
```

```
                    int nominator)
{
#ifdef NOT_YET
   static mirrorSource_t mirrorSourceData;
...
   mirrorSourceData.mirrorTarget =
MakeOIDFromDot("1.3.6.1.4.1.99.12.19.2.1.1.1.2");
   mirrorSourceData.mirrorGetVarSearchType = 1;
   mirrorSourceData.mirrorGetVarResult =
MakeOIDFromDot("1.3.6.1.4.1.99.12.19.2.1.1.1.2");
   mirrorSourceData.mirrorGetVarStatus = 11;
   mirrorSourceData.mirrorInteger = 18;
   mirrorSourceData.mirrorOctetString = MakeOctetStringFromText(“Microware OS-9”);
   mirrorSourceData.mirrorOID = MakeOIDFromDot("1.3.6.1");
   SET_ALL_VALID(mirrorSourceData.valid);
   return(&mirrorSourceData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
...
mirrorLoopbackTargetAfter_t *
k_mirrorLoopbackTargetAfter_get(int serialNum, ContextInfo *contextInfo,
                                int nominator)
{
#ifdef NOT_YET
   static mirrorLoopbackTargetAfter_t mirrorLoopbackTargetAfterData;
...
   mirrorLoopbackTargetAfterData.mirrorLoopbackInteger2 = 78;
   mirrorLoopbackTargetAfterData.mirrorLoopbackOctetString2 =
MakeOctetStringFromText("with SNMP Agents");
   mirrorLoopbackTargetAfterData.mirrorLoopbackOID2 =
MakeOIDFromDot("1.3.6.1.2.1");
   SET_ALL_VALID(mirrorLoopbackTargetAfterData.valid);
   return(&mirrorLoopbackTargetAfterData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
```

Also, you need to change line in the `k_mirrorSource_set()` with the `COMMIT_FAILED_ERROR` message to `NO_ERROR` message. This is important because without this change, you will not be able to modify any read-write/read-create variables in the subagent.

Name

`snmpd_note`

Description

The Note MIB demonstrates how a standard MIB containing a `write-only` ACCESS clause can be supported by SNMP Research tools and method routines.

The Note MIB describes the management information for a message delivery system, such as e-mail.

Example Patch

Add the patch to the following file to make the agent properly:

`NOTE/RELS/<processor>/<version>/k_note.c`

The code snippet below shows an example of adding the patch. The variable value can be modified as appropriate. The bold parts are the example variable values.

```
...
int noteNumber = 1;
...
note_t *
k_note_get(int serialNum, ContextInfo *contextInfo,
           int nominator)
{
#ifdef NOT_YET
   static note_t noteData;
...
   noteData.noteNumber = noteNumber;
   SET_ALL_VALID(noteData.valid);
   return(&noteData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
...
int
k_note_set(note_t *data,
           ContextInfo *contextInfo, int function)
{
```

```
    char msg1[30],
         msg2[30];

    strcpy(msg1, "MCSD, RadiSys Corp.");
    strcpy(msg2, "OS-9 with SNMP");

    data->noteRecipient = MakeOctetString((unsigned char *) msg1, 19);
    data->noteBody = MakeOctetString((unsigned char *) msg2, 14);

    /* Increment noteNumber so we know that we write something */
    noteNumber++;

    return NO_ERROR;
}
```

Be sure to change the line in the `k_note_set()` with the `COMMIT_FAILED_ERROR` message to `NO_ERROR` message. This is important because without this change, you will not be able to modify any read-write/read-create variables in the subagent.

Name

`snmpd_room`

Description

The Room MIB demonstrates SNMP Research Library support for multi-dimension table creation and maintenance.

This example describes the management information of a hotel by generating a conceptual table. The table contains the floor plan of the hotel, the Smoke Detector and Motion Detector status on each floor and room.

Example Patch

For this example, you only need to copy the `k_room.fin` from the `room` directory as `k_room.c` in the `RELS/<processor>/<version>` directory.

thrm

Thermometer MIB

Name

`snmpd_thrm`

Description

The Thermometer MIB contains only `read-only INTEGER` scalar object types.

This example describes the management information for a simple thermometer, which reports two variables. The variables for this MIB can not be changed.

Example Patch

Add the patch to the following file to make the agent properly:

`THRM/RELS/<processor>/<version>/k_thrm.c`

The code snippet below shows an example of adding the patch. The variable value can be modified as appropriate. The bold parts are the example variable values.

```
...
thermometer_t *
k_thermometer_get(int serialNum, ContextInfo *contextInfo,
                  int nominator)
{
#ifdef NOT_YET
   static thermometer_t thermometerData;
...
   thermometerData.thrmTemperature = 80;
   thermometerData.thrmUnits = 2;
   SET_ALL_VALID(thermometerData.valid);
   return(&thermometerData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
```

See Also

thst

thst

Thermostat MIB

Name

`snmpd_thst`

Description

The Thermostat MIB contains only `read-write INTEGER` scalar object types.

This example describes the management information for a simple thermometer, which reports two variables. Both variables for this MIB can be set.

Example Patch

Add the patch to the following file to make the agent properly:

`THST/RELS/<processor>/<version>/k_thst.c`

The code snippet below shows an example of adding the patch. The variable value can be modified as appropriate. The bold parts are the example variable values.

```
...
int thst_temp, thst_unit;
...
int
k_thst_initialize(void)
{
    thst_temp = 25;
    thst_unit = 1;
    return 1;
}
...
thermostat_t *
k_thermostat_get(int serialNum, ContextInfo *contextInfo,
                 int nominator)
{
#ifdef NOT_YET
   static thermostat_t thermostatData;
...
   thermostatData.thstTemperature = thst_temp;
   thermostatData.thstUnits = thst_unit;
   SET_ALL_VALID(thermostatData.valid);
   return(&thermostatData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
...
int
k_thermostat_set(thermostat_t *data,
                 ContextInfo *contextInfo, int function)
{
   thst_temp = data->thstTemperature;
   thst_unit = data->thstUnits;
   return NO_ERROR;
}
...
```

Be sure to change the line in the `k_thermostat_set()` with the `COMMIT_FAILED_ERROR` message to `NO_ERROR` message. This is important because without this change, you will not be able to modify any read-write/read-create variables in the subagent.

See Also

`thrm`

tmon

Trap Monitor MIB

Name

`snmpd_tmon`

Description

The Trap Monitor MIB contains only `read-only OBJECT IDENTIFIER` scalar object types. it is also one of two example MIBs that demonstrate how to handle traps. The other one is the Surge-Protector MIB.

This example has a variable that contains the `OBJECT INDENTIFIER` of the last trap that was sent by the agent.

Example Patch

For this example, you only need to copy the `k_tmon.fin` from the `tmon` directory as `k_tmon.c` in the `RELS/<processor>/<version>` directory.

Name

`typeagt`

Description

The Typesetter MIB contains only `read-only OCTET STRING` scalar object types.

This example describes the management information for a simple typesetting device, such as a printer or computer terminal.

Example Patch

Add the patch to the following file to make the agent properly:

`TYPE/RELS/<processor>/<version>/k_type.c`

The code snippet below shows an example of adding the patch. The variable value can be modified as appropriate. The bold parts are the example variable values.

```
...
static typesetter_t typesetterData;
int
k_type_initialize(void)
{
   typesetterData.tsBoldfaceBegin = MakeOctetStringFromText("Ctrl-B");
   typesetterData.tsBoldfaceEnd = MakeOctetStringFromText("Ctrl-Shift-B");
   typesetterData.tsNegativeBegin = MakeOctetStringFromText("Ctrl-N");
   typesetterData.tsNegativeEnd = MakeOctetStringFromText("Ctrl-Shift-N");
   typesetterData.tsUnderlineBegin = MakeOctetStringFromText("Ctrl-U");
   typesetterData.tsUnderlineEnd = MakeOctetStringFromText("Ctrl-Shift-U");
   SET_ALL_VALID(typesetterData.valid);
   return 1;
}
...
int
k_type_terminate(void)
{
   FreeOctetString(typesetterData.tsBoldfaceBegin);
   FreeOctetString(typesetterData.tsBoldfaceEnd);
   FreeOctetString(typesetterData.tsNegativeBegin);
   FreeOctetString(typesetterData.tsNegativeEnd);
   FreeOctetString(typesetterData.tsUnderlineBegin);
```

```
   FreeOctetString(typesetterData.tsUnderlineEnd);
   return 1;
}

typesetter_t *
k_typesetter_get(int serialNum, ContextInfo *contextInfo,
                 int nominator)
{
#ifdef NOT_YET
   return(&typesetterData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
```

Name

`snmpd_vcr`

Description

The VCR MIB demonstrates two-dimensional table row creation in the SNMP agent.

This example describes the management information for a programmable video cassette recorder, with a conceptual table that contains information about the VCR programming.

There are two versions of this subagent, **auto** and **byhand**. The binary that is shipped is built from the `byhand` version. If you prefer the `auto` version, change the makefile to point to the `vcr/auto` directory.

Example Patch

Add the patch to the following file to make the agent properly:

`VCR/BYHAND/RELS/<processor>/<version>/k_vcr.c`

The code snippet below shows an example of adding the patch. The variable value can be modified as appropriate. The bold parts are the example variable values.

```
...
#define MAX_ENTRIES 5
static vcrProgramEntry_t vcrProgramEntryData[MAX_ENTRIES];

#define GET_CHANNEL_6() 6
#define GET_POWER() D_vcrPowerState_on

int
k_vcr_initialize(void)
{
   int i;

   for (i = 0; i < MAX_ENTRIES; i++) {
       vcrProgramEntryData[i].vcrProgramStatus = D_vcrProgramStatus_destroy;
   }

   /* Initialize Entry 2 */
   vcrProgramEntryData[2].vcrProgramTableIndex = 2;
   vcrProgramEntryData[2].vcrProgramChannel = 18;
   vcrProgramEntryData[2].vcrProgramStartTime = MakeOctetStringFromText("02.1.1,
07:00");
   vcrProgramEntryData[2].vcrProgramStopTime = MakeOctetStringFromText("02.1.1,
09:00");
   vcrProgramEntryData[2].vcrProgramSpeed = 500;
   vcrProgramEntryData[2].vcrProgramStatus = D_vcrProgramStatus_active;

   /* Initialize Entry 4 */
   vcrProgramEntryData[4].vcrProgramTableIndex = 4;
   vcrProgramEntryData[4].vcrProgramChannel = 3;
   vcrProgramEntryData[4].vcrProgramStartTime = MakeOctetStringFromText("02.1.1
07:00");
   vcrProgramEntryData[4].vcrProgramStopTime = MakeOctetStringFromText("02.1.1
09:00");
   vcrProgramEntryData[4].vcrProgramSpeed = 1000;
   vcrProgramEntryData[4].vcrProgramStatus = D_vcrProgramStatus_active;

   return 1;
}
...
vcrGeneral_t *
k_vcrGeneral_get(int serialNum, ContextInfo *contextInfo,
                 int nominator)
{
#ifdef NOT_YET
   static vcrGeneral_t vcrGeneralData;
...
   vcrGeneralData.vcrChannel = GET_CHANNEL_6();
   vcrGeneralData.vcrPowerState = GET_POWER();
   SET_ALL_VALID(vcrGeneralData.valid);
   return(&vcrGeneralData);
#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
```

```
...
vcrProgramEntry_t *
k_vcrProgramEntry_get(int serialNum, ContextInfo *contextInfo,
                      int nominator,
                      int searchType,
                      SR_INT32 vcrProgramTableIndex)
{
#ifdef NOT_YET

   /* Remove the default vcrProgramEntryData here */
   int i;

   for (i = vcrProgramTableIndex; i < MAX_ENTRIES; i++) {
       if (vcrProgramEntryData[i].vcrProgramStatus != D_vcrProgramStatus_destroy) {
           break;
       }
   }
   if (searchType == EXACT) {
       if (i != vcrProgramTableIndex) {
           return NULL;
       }
   }
   if (i >= MAX_ENTRIES) {
       return NULL;
   }

   SET_ALL_VALID(vcrProgramEntryData[i].valid);
   return(&vcrProgramEntryData[i]);

#else /* NOT_YET */
   return(NULL);
#endif /* NOT_YET */
}
...
int
k_vcrProgramEntry_test(ObjectInfo *object, ObjectSyntax *value,
                       doList_t *dp, ContextInfo *contextInfo)
{
    vcrProgramEntry_t *data = (vcrProgramEntry_t *) (dp->data);

    if (data->vcrProgramTableIndex >= MAX_ENTRIES) {
        return NO_CREATION_ERROR;
    }
    else if (data->vcrProgramTableIndex <= 0) {
        return NO_CREATION_ERROR;
    }
    else {
        return NO_ERROR;
    }
}
...
int
k_vcrProgramEntry_set(vcrProgramEntry_t *data,
                      ContextInfo *contextInfo, int function)
```

```
{
   int i = data->vcrProgramTableIndex;

   vcrProgramEntryData[i].vcrProgramTableIndex = i;
   vcrProgramEntryData[i].vcrProgramChannel = data->vcrProgramChannel;
   vcrProgramEntryData[i].vcrProgramStartTime =
CloneOctetString(data->vcrProgramStartTime);
   vcrProgramEntryData[i].vcrProgramStopTime =
CloneOctetString(data->vcrProgramStopTime);
   vcrProgramEntryData[i].vcrProgramSpeed = data->vcrProgramSpeed;
   vcrProgramEntryData[i].vcrProgramStatus = data->vcrProgramStatus;

   return NO_ERROR;
}
...
```

Be sure to change the line in the `k_vcrProgramEntry_set()` with the `COMMIT_FAILED_ERROR` message to `NO_ERROR` message. This is important because without this change, you will not be able to modify any read-write/read-create variables in the subagent.

Product Discrepancy Report

To: Microware Customer Support

FAX: 515-224-1352

From:_____

Company:_____

Phone:_____

Fax:_____Email:_____

Product Name:

Description of Problem:

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Host Platform_____

Target Platform_____

RadiSys.

MICROWARE SOFTWARE