NetSNMP Port to RTEMS

Chris Johns (cjohns@cybertec.com.au) 1 April 2003

This document details the port of the NetSNMP <http://www.net-snmp.org/> project to the RTEMS <http://www.rtems.com/RTEMS/rtems.html> real-time embedded operating system.

Contents

1 Status

The port is in its early days and the results are excellent. The agents embedded and operates as normal under RTEMS. The main missing part is the configure and Makefile environment. The standard configure and Makefiles maintained as part of the package have not been used because:

- Proving the agent is able to be ported rather than attempting to support the standard configure and Makefile system was a priority,
- The configure and Makefiles are complex and an initial review showed problems with cross-compliation,
- Changes made to support cross-complication may break important existing targets.

The aim is to have the RTEMS support integrated into the standard package.

The NetSNMP port has been tested with SNMPv1 and SNMPv2 with MIB-II and a custom MIB. SNMPv3 support builds, how-ever it has not been tested.

The RTEMS operating system uses the FreeBSD TCP/IP stack. This stack uses the `sysctl` system call to handle MIB-II requests. RTEMS supports the `sysctl` system call and therefore NetSNMP needed few changes to support MIB-II. The RTEMS 4.6.0pre-release may not show all TCP or UDP connections correctly. This is due to the merging of the lastest FreeBSD `sysctl` code to the older TCP/IP code in RTEMS. A port of the latest TCP/IP code from FreeBSD should resolve this problem.

2 Building NetSNMP

The package is currently based on the NetSNMP 5.0.6 version. It contains code changes needed to support RTEMS, plus Makefile.am files used by Cybertec to build the code. At this point in time you will need to create your own Makefiles if you are using this early release. The Makefile.am files supplied will provide a list of files you need to build. Your Makefiles will need to support your specific cross-compilation flags.

At Cybertec, we build the whole NetSNMP package into a single library that we link our SNMP applications too. It is recommended you also build a library.

The code changes in this package are being slowly sent to the NetSNMP project for inclusion into the standard code base. Having the main code based support RTEMS is a critical goal of the porting project.

The SNMP library and agent code has been ported. The AgentX plus other agent types have not been ported.

At Cybertec we have a an autoconf script that creates the correct header files. The specific peice of autoconf we use is:

```
dnl
dnl Net Snmp package.
dnl
dnl The package is preconfigured for RTEMS. No special configure
dnl magic to select package options.
dnl AC_MSG_CHECKING([for net-snmp])
AS_IF([test -d ${srcdir}/net-snmp],
      [AC_MSG_RESULT([yes])
 AC_SUBST(NET_SNMP_INCLUDES,
        "-I ${srcdir}/net-snmp/include -I `pwd`/net-snmp/include")
 AC_CONFIG_FILES(net-snmp/Makefile
        net-snmp/snmplib/Makefile
        net-snmp/agent/Makefile
        net-snmp/agent/helpers/Makefile
        net-snmp/agent/mibgroup/Makefile
        net-snmp/include/mib_module_inits.h:net-snmp/agent/mibgroup/mib_module_inits.h.in
        net-snmp/include/mib_module_shutdown.h:net-snmp/agent/mibgroup/mib_module_shutdown.h.in
        net-snmp/include/mib_module_dot_conf.h:net-snmp/agent/mibgroup/mib_module_dot_conf.h.in
        net-snmp/include/mib_module_includes.h:net-snmp/agent/mibgroup/mib_module_includes.h.in
        net-snmp/include/net-snmp/agent/mib_module_config.h:net-snmp/include/net-snmp/agent/mib_module_
        net-snmp/include/net-snmp/library/snmpv3-security-includes.h:net-snmp/include/net-snmp/library/
        net-snmp/include/net-snmp/net-snmp-config.h:net-snmp/include/net-snmp/net-snmp-config.h.in
net-snmp/snmplib/snmpsm_init.h:net-snmp/snmplib/snmpsm_init.h.in)],
AC_MSG_RESULT([no]))
```

3 Starting NetSNMP

The NetSNMP agent runs as a single task under RTEMS. The agents main is compiled specifically for RTEMS and is involked by calling the function:

```
int snmpd_init (rtems_task_priority priority, char* cmdline);
```

The command line is the documented in the snmpd man page (<http://www.net-snmp.org/man/snmpd.html>). Select the priority that suites your application. SNMP will use your processing resources and as it is typically used for monitoring a priority lower than your important real-time tasks as well as the networking stack is recommended.

This document will refer to `snmpd` as the NetSNMP agent running as an RTEMS task.

The NetSNMP uses a number of files. The most important of these is the `/etc/snmpd.conf`. You need to arrange to have this file created before you start `snmpd`. Many ways exist in RTEMS to create this file. For example code to write the file, coping directly from non-volatile storage to the IFMS, reading from a disk drive, or using the `untar` call.

The standard NetSNMP configure script creates a list of modules that the agent supports. The modules active when `snmpd` is running is controlled by command line options. This makes sense on a Unix or Windows host. Here you build into the agent all that is supported by the host and the system administrator enables what is needed. On an embedded system this approach results in an executable that is much larger than required. To keep the image size to what you need a function called by the agent that specifies the list of agent modules you require needs to be provided. An RTEMS application may look like:

```
#include <stdio.h>
#include <rtems.h>
#include <rtems/untar.h>

void init_system_mib();
void init_sysORTable();
void init_at();
void init_interfaces();
void init_snmp_mib();
void init_tcp();
void init_icmp();
void init_ip();
void init_udp();
void init_vacm_vars();
void init_setSerialNo();
void init_snmpEngine();
void init_snmpMPDStats();
void init_usmStats();
void init_usmUser();
void init_snmpNotifyTable();
void init_snmpNotifyFilterTable();
void init_snmpNotifyFilterProfileTable();
void init_snmpTargetAddrEntry();
void init_snmpTargetParamsEntry();
void init_target_counters();
void init_nsTransactionTable();
void init_nsModuleTable();
void init_override();
void init_var_route();
void init_vacm_context();

void init_mib_modules ();
int  snmpd_init (rtems_task_priority priority, char* cmdline);
```

```
/*
 * These are created by the linker when linking a binary file.
 */
extern int _binary_net_snmp_files_tar_start;
extern int _binary_net_snmp_files_tar_size;

/*
 * We need to supply this function. It is call by the
 * the snmpd agent. List the modules we need. They will
 * be linked from the NetSNMP library.
 */
void init_mib_modules ()
{
  init_system_mib();
  init_sysORTable();
  init_at();
  init_interfaces();
  init_snmp_mib();
  init_tcp();
  init_icmp();
  init_ip();
  init_udp();
#if 0
  init_vacm_vars();
  init_setSerialNo();
  init_snmpEngine();
  init_snmpMPDStats();
  init_usmStats();
  init_usmUser();
  init_snmpNotifyTable();
  init_snmpNotifyFilterTable();
  init_snmpNotifyFilterProfileTable();
  init_snmpTargetAddrEntry();
  init_snmpTargetParamsEntry();
  init_target_counters();
  init_nsTransactionTable();
  init_nsModuleTable();
  init_override();
  init_var_route();
  init_vacm_context();
#endif
}

/**
 * Main line.
 */
int Init ()
```

```
{
  /*
   * Place the snmpd files into the IMFS.
   */
  Untar_FromMemory ((unsigned char *) (&_binary_net_snmp_files_tar_start),
                    (int) &_binary_net_snmp_files_tar_size);

  /*
   * Start the BSD TCP/IP stack.
   */
  rtems_bsdnet_initialize_network ();

  /*
   * Start the Net SNMP server.
   */
  if (!snmpd_init (150, "-f -L"))
  {
    printf ("Net SNMP did not start\n")
   return 1;
  }

  rtems_task_delete (RTEMS_SELF);
}
```