RTEMS Debugging with GDB

Till Straumann,
2005/5/31
Cross-Debugging

Host

GUI-Frontend ('ddd')

Cross-GDB ('powerpc-rtems-gdb')

Host Filesystem

Source Files

Object Files

Target

GDB Remote Protocol

serial, TCP/IP

Target Agent

Target Filesystem

Executable ('rtems.nxe')
Target Agent

- Server for GDB remote protocol
  - memory R/W
  - register R/W
  - breakpoints/exceptions
  - Loadable modules (Cexp): propagate info about loaded object files and section addresses
- Host GDB translates e.g., 'stack trace' into low-level reqs. to agent: read SP, read memory blocks.
Target Agent Types

High Level

- Operates at 'task' level
- Minimal intrusion (only tasks hitting a breakpoint are stopped)
- Needs a functional system.
- No ISR debugging

Low Level

- Operates at 'ISR' level
- Maximal intrusion; complete system is stopped
- Can do ISR debugging
General Comments

- RTM (GDB, rtems-agent, DDD docs)

http://sources.redhat.com/gdb/documentation
http://www.gnu.org/manual/ddd

- Some behavioral/semantical particularities can be understood historically: GDB is mainly a native debugger for unix processes (but also cross-debugger for many different hosts and targets).
  - breakpoint stops all threads of a process (maps to 'low-level' agent model)
  - search path
Host Setup

- Compile with \(-g\). Optimization may still be used.
- Link with \(-g\) (don't use \(-s\)); don't strip debugging info (epics workaround: make 'OP_SYS_LDFLAGS=')
- \texttt{PATH} must point to cross-gdb and all target executables; there is no way of changing this from within a gdb session. (Could be changed.)
- Sources are usually found automatically; search paths can be changed (gdb \texttt{dir} command), however.
Target Setup

• "rtems-gdb-stub.obj" must be loaded and started

```c
rtems_gdb_start(int priority, char *serial_device)
```

• If no serial device is specified, the agent listens on TCP port 4444.

• PATH should point to all modules (required for gdb rtems load command to work).
Starting a GDB Session

- Start GUI and GDB
  ```
  ddd --debugger powerpc-rtems-gdb rtems.nxe
  ```
- Attach to target (command window)
  ```
  (gdb) target rtems-remote <ioc>:4444
  ```
  - connects to target
  - obtains list of all loaded modules with addresses
- Detach from target (releases all stopped threads)
  ```
  (gdb) detach
  ```
Breakpoints and Threads

- **GDB semantics (history):** target is either running or stopped (breakpoint, exception/signal). GDB can only talk to stopped target. Context (registers, stack) belongs to a 'current thread' (thread cmd).

- **RTEMS agent:**
  - only threads that hit a breakpoint or incur an exception or threads that are explicitly stopped (gdb thread cmd) are stopped.
  - All stopped threads remain stopped until gdb continues, steps or detaches.
  - agent provides 'helper thread'. Context used when attaching or interrupting (<Ctrl-C>).
Breakpoints and Threads (Cont.)

- breakpoints are only physically inserted while target is running (i.e., after \texttt{gdb continue} or while stepping). No thread will hit an active breakpoint while \texttt{gdb} is waiting for user input (prompt visible in the command window) since the target is stopped at this point (\texttt{gdb} semantics).

- \texttt{Use gdb info threads} command to get a list of threads (crashed threads are 'suspended').

- \texttt{GDB thread IDs needed for switching context.}
Module Synchronization

• Debugging involves re-compilation and re-loading of modules on target. GDB's module/object list and the target's must be in sync!

either

  – unload/reload modules from Cexp prompt: must use `gdb rtems sync-objs` command to refresh GDB's object list.

or

  – unload/reload modules through GDB: use `gdb rtems load` command (target PATH must be set correctly).
Caveats

- Beware of deadlocks: must not set breakpoints in code used by critical sections of gdb agent (in particular: networking).
- Objects used by GDB and target must be in sync.
- Optimizer changes flow of code (weird jumps in source code, variables disappear or have strange values).
- Alpha code. Feedback needed.