Debugging mit SE2K

Peter Kainberger

August 26, 2004

1 Möglichkeiten (SE2K ab firmware-release 4)

- Schnittstelle für den Debugger-Anschluss mit variabler Baudrate (statt bisher fix 19200).
- Unterbrechung des laufenden Programms über Eingabe von <CTRL-C> sowhl im debugger als auch auf einer direkt angeschlossenen Tastatur.
- 2 Triggerausgänge am Diagnoseport sind mit Events verknüpfbar (per Property).
- 1 Komparator-Register kann (per Property) mit einer Adresse belegt werden, bei der dann ein Interrupt erzeugt wird, der widerum in den Debugger führt (also eigentlich ein Break/Watch-point in Hardware). Es kann zwischen Lese- und Schreibzugriffen unterschieden werden.

2 Einschränkungen

- Der Break/Watch-point in Hardware kann nicht über den gdb (bzw ddd68) eingestellt werden. Es ist leider nicht ohne größeren Aufwand möglich die gdb-Konfiguration so einzurichten, dass HW-Break/Watch-points verwendet werden können. Bei allen Möglichkeiten, die zur Verfügung stehen überwiegen die Nachteile diesen einen Vorteil.
- Zur Sicherheit sind die Unterbrechungsmöglichkeiten für ein laufendes Programm *per default* erst mal DISABLED, sie müssen explizit per Property ENABLED werden.
- Die Baudrate (19200 oder *variabel*) muss im gdb bzw ddd68 explizit richtig eingestellt werden (*set remotebaud 19200)* und hängt jetzt von CPU (FIO8130, SE2K oder Gruppen-Micros) und firmware-release ab. Per default ist die Baudrate der SE2K ab release 4 auf 115200 und bei allen anderen SEs auf 19200 eingestellt.
- Leider gibt der ddd68 eine Unterbrechungsanforderung per <CTRL-C> erst dann über die Schnittstelle an den frontend weiter, wenn er schon

mal mit diesem debugging-frontend geredet hat. Also muss man erst mal die debugging-Verbindung herstellen, dann den frontend rennen lassen und danach erst kann man mit <CTRL-C> anhalten. Also ein laufendes Programm ohne Vorbereitung des debuggers einfach anhalten *iss nich.*

3 Erweiterungen

Wenn der debugger enabled ist, kann über die angeschlossene Tastatur das laufende Programm durch Eingabe von <CTRL-C> unterbrochen werden (über die Umleitung mit VMETERM geht das auch, wie immer muss aber die Eingabe mit <RETURN> abgeschlossen werden).

Eine neue Property DEBUG stellt folgende Funktionen zur Verfügung:

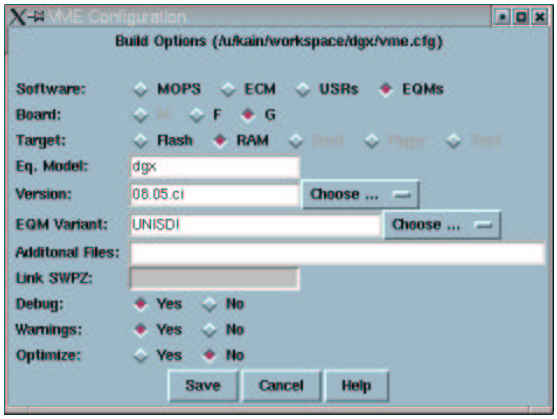
1. *enable/disable* des debuggers (Watchdog aus, <CTRL-C>-Interrupt an)
2. *break* durchführen, damit wird Kommunikation mit gdb bzw ddd68 möglich
3. *set baudrate* (Baudrate der debugging-Schnittstelle einstellen)
4. *set/clear addressstop* (Adresse ins Komparator-Register eintragen bzw. löschen)
5. *set/clear triggerout* (Triggerausgänge mit Events belegen bzw. Einträge löschen). Da das Eventfilter meist auf 8-Bit-breit eingestellt ist, kann nicht nach Beschleunigernummern unterschieden werden. Vorerst ist der Pulsmodus auf *singlepulse* gestellt, die Option *framepulse* wir derzeit nicht zur Auswahl angeboten.

Die Funktionsnummer wird als Parameter und die notwendigen Daten werden als Daten der Property transportiert. Zur Vereinfachung gibt es ein kleines NODAL-Programm (SEDEBUG), das die Handhabung erleichtert.

4 Kurzanleitung zum debugging

- Rechner mit SE verbinden: Der Rechner von dem aus man debuggen will, muss über eine seiner seriellen Schnittstellen mit der seriellen Schnittstelle der SE verbunden werden (Nullmodem nicht vergessen). Bei den 'alten' SEs gibt's nur eine Schnittstelle und bei den neuen (SE2K) ist es die COM1.
- Im vmeconfig-Menü des Target *RAM* und die Option *debug* anwählen. Will man breakpoints setzen können, muss man unbedingt das Target auf *RAM* setzen und ohne *debug* kann man nicht symbolisch debuggen. Wenn man den Programmcode durchsteppen will, dann empfiehlt sich die Abwahl von *optimize.* Mit *optimize* ist es oft schwierig, breakpoints auf Programmzeilen zu setzen, denn nach der Codeoptimierung ist die Verwandtschaft des erzeugten Codes mit dem ursprünglichen Programm

oftmals nicht mehr sehr gross und die Zuordnung zwischen C-Programm und erzeugtem Code ist nicht mehr eindeutig.



- Software downloaden
- Software im RAM starten mit
  vmeboot -r rechnername
- debugger auf dem angeschlossenen linux-Rechner starten mit
  ddd68 filename.sym
  (ddd68 ist ein alias für *ddd –debugger m68k-unknown-coff-gdb*)
- falls nötig, Baudrate einstellen mit
  set remotebaud 19200 (bei 'alten' SEs und neuen SE2K firmware-release < 4)
  oder
  set remotebaud 115200 (bei SE2K mit firmware-release $>= 4$)
- Verbindung zur SE herstellen mit
  target remote /dev/ttyS0 (ttyS0 für COM1 des linux-Rechners und ttyS1 für COM2)
- SE in den debugmodus bringen über eine der folgenden Möglichkeiten:
  1. im Nodalprogramm sedebug zuerst *enable debugger* und dann *break* ausführen
  2. an einem direkt angeschlossenen Terminal <CTRL-C> drücken
  3. an der SE direkt den *ABORT*-Knopf drücken
  4. die SE mit debug-option starten: *vmeboot -rd rechnername*
- wenn die Verbindung einmal hergestellt ist, kann man auch im ddd68-Eingabefenster <CTRL-C> direkt eingeben

Und nun viel Spass und Erfolg beim debuggen!