CAPÍTULO 5

SISTEMAS DE DEPURACIÓN

5.- Sistemas de depuración.

5.1.- Definición de sistema de depuración.

Es un programa que te permite ver qué es lo que está ocurriendo dentro de otro programa mientras este último se ejecuta, o bien qué estaba haciendo el programa en el momento en que dejó de funcionar correctamente.

5.2.- Capacidades de los sistemas de depuración.

Los sistemas de depuración pueden hacer principalmente cuatro cosas diferentes (más algunas otras que se emplean como apoyo de éstas) para ayudar al usuario a localizar los posibles errores:

- Comenzar el programa del usuario, especificando cualquier cosa que pueda afectar a su funcionamiento normal.

- Hacer que el programa del usuario se detenga bajo ciertas condiciones.

- Examinar lo que ha ocurrido, cuando el programa del usuario se ha detenido de forma imprevista.

- Cambiar cosas dentro del programa del usuario, de modo que se pueda experimentar con la corrección de los efectos de un determinado error y sea posible localizar el siguiente.

5.3.- Particularidades de los sistemas de depuración.

En principio, los sistemas de depuración están diseñados de forma que sean capaces de entender las órdenes, rutinas y llamadas a las diferentes funciones que se emplean en un determinado programa.

Sin embargo, los sistemas de depuración presentan una limitación característica; cada sistema de depuración suele "entender" únicamente un determinado lenguaje de programación. Esto ocurre porque los sistemas de depuración son desarrollados generalmente por compañías privadas, que ofrecen a los programadores de sus aplicaciones (sean éstas software o hardware) una herramienta eficaz para elaborar sus programas. Dado que estos sistemas de depuración están orientados a comprobar la programación de programas desarrollados por los usuarios, para dichas aplicaciones de los productos de la empresa, los sistemas de depuración son desarrollados para comprender el mismo lenguaje de programación que se emplea en la aplicación.

Pese a que el programa empleado en la depuración pueda ser muy similar al empleado en la codificación del depurador (como por ejemplo, la semejanza entre los programas escritos con C y C++), cada lenguaje de programación presenta sus propias particularidades; esto es lo que hace que los sistemas de depuración suelan ser específicos para cada lenguaje.

Tenemos de esta forma sistemas de depuración desarrollados en C, C++, Fortran, etc. Dichos sistemas de depuración se emplean para comprobar el correcto funcionamiento de programas que se han desarrollado empleando el lenguaje de programación con que se programó una aplicación o grupo de aplicaciones (puede llegar a ser un depurador que pueda depurar todas las aplicaciones de una empresa) concretas.

5.3.1.- Importación de lenguajes.

Hay, sin embargo, sistemas de depuración desarrollados por medio de un lenguaje de programación (por ejemplo, C) que pueden ser empleados para depurar programas que se hayan escrito empleando un lenguaje diferente (por ejemplo, Fortran).

Esto se hace de la siguiente manera; se desarrollan una serie de archivos que permiten que el depurador pueda alterar sus fuentes, de forma que pueda añadirse información que permita al depurador entender la estructura empleada por el lenguaje de programación de programa que se pretende depurar.

De esta forma, podemos por ejemplo desarrollar un depurador en Fortran con el que se pueden entender las llamadas a rutinas, operaciones de punteros, colas, etc. que se pueden dar en un programa desarrollado en C, mientras que Fortran no funciona de esta manera. Con esto conseguimos un sistema de depuración que puede entender y depurar diversos lenguajes, sin más que añadir los archivos adecuados para permitir la interpretación del nuevo lenguaje.

5.4.- Depuradores intrusivos y no intrusivos.

En cada depurador se establece la comunicación con el sistema que se desea depurar de una forma diferente, pero a grandes rasgos se pueden distinguir dos tipos de depuradores; los depuradores intrusivos y los no intrusivos.

Esta división es importante, y la emplearemos como base para la comparación de las funcionalidades de los distintos depuradores que veremos en el apartado 5.5.

5.4.1.- Depuradores intrusivos.

Un depurador intrusivo es aquel que requiere la presencia de un pequeño programa o aplicación en el sistema objetivo, con el que puede intercambiar información sobre el sistema.

Este pequeño programa se denomina generalmente "monitor", puesto que monitoriza el desarrollo del programa residente en el sistema que se desea depurar.

Normalmente, dicho monitor es capaz de efectuar llamadas por su cuenta al depurador por medio de mensajes, de modo que el depurador puede llevar a cabo la depuración con la información proporcionada por el monitor.

De modo análogo, el depurador puede comunicarse con el monitor si el propio programa o el usuario del programa desean obtener determinados datos en un momento dado.

En ambos casos, el de la comunicación comenzada por el monitor o por el depurador/usuario (para el monitor es indiferente que sea uno u otro), el sistema bajo depuración no tiene porqué detenerse, salvo en los casos de lectura y/o escritura de determinados registros o regiones de memoria que sean necesarios para que el sistema continúe con su ejecución.

Evidentemente, los sistemas intrusivos requieren de un espacio de memoria libre en el sistema que se pretende depurar, lo cual tiene como principal consecuencia la reducción del espacio de memoria disponible para la programación del sistema objetivo de la depuración. Esta desventaja sólo se presenta como tal en sistemas microprocesadores con poca memoria, por lo que se presentan como una opción útil y deseable al permitir un control casi exhaustivo del sistema.

En nuestro sistema en concreto, la tarjeta de evaluación EVB2107 de Motorola, este tipo de depuradores no se pueden utilizar por varias razones:

- Una de las condiciones de nuestro proyecto es que el interfaz no debe ocupar espacio de memoria en el sistema objetivo.
- El monitor ocupa un espacio en la memoria del sistema objetivo que disminuye en gran manera las posibilidades de programación del sistema (disminuye el tamaño de los programas que se pueden introducir en el sistema).

5.4.2.- Depuradores no intrusivos.

Los depuradores no intrusivos son aquellos en los que no es necesaria una aplicación residente en el sistema que se desea depurar, al contrario que en los sistemas intrusivos.

Dichos depuradores aprovechan determinados puertos "dedicados" del sistema objetivo, comunicándose a través de uno de ellos. Sin embargo, estos puertos "dedicados" emplean un protocolo de comunicación que suele ser diferente de los empleados por la plataforma en que se encuentra el programa depurador. Por esta razón dichos depuradores no intrusivos hacen uso de unos dispositivos hardware (denominados "dongle" en los círculos profesionales) que adaptan los protocolos entre determinados puertos de la plataforma del depurador y el puerto dedicado del sistema objetivo.

Hay que señalar que los puertos dedicados de que hablamos no son puertos absolutamente estandarizados; para empezar, no requieren que el sistema se detenga cuando se esté accediendo a ellos desde el exterior (desde el depurador), y pueden modificar determinados registros sin necesidad de hacer uso del registro de instrucción (IR) del sistema. Los registros de que dispone el puerto dedicado permiten además la alteración de la programación del sistema, así como la detención del mismo cuando se cumpla una determinada condición (que se acceda a un determinado rango de memorias, por ejemplo).

En el presente proyecto, el sistema JDONG se comporta como una especie de "dongle" que realiza la adaptación de protocolos del puerto serie de la plataforma del depurador y del puerto OnCE/JTAG (que uno de los puertos "dedicados" de que hemos hablado).

5.5.- Comparativa de depuradores.

En este apartado veremos algunos de los depuradores comerciales o de distribución gratuita que podemos encontrar, y estudiaremos sus funcionalidades o características más destacables, estableciendo una comparativa entre las ventajas de emplear uno u otro depurador en la depuración de la tarjeta de evaluación EVB2107 empleada para la realización del presente proyecto.

5.5.1.- Depurador GDB, de GNU.

Este depurador es el que se ha utilizado como base para el desarrollo del presente proyecto. Su funcionamiento interno está explicado en mayor detalle en el Capítulo 4: "Introducción a GDB"; en este apartado se pretenden destacar las principales funcionalidades del mismo, estableciendo la base para una comparativa entre los diferentes depuradores del mercado.

Sus principales características son las siguientes:

- Funciona sobre plataforma Linux, Solaris y UNIX.
- Es de fuente abierta, lo cual nos permite la modificación e implementación de funcionalidades nuevas y más adecuadas a nuestro sistema objetivo.

- Permite la depuración de sistemas microprocesadores de diferentes empresas y/o familias, como Intel x86, Motorola M·Core, Motorola 680x0, Z80, etc.
- Es gratuito.
- Hay muchos grupos de programación en la red, de los que se puede obtener información sobre determinados aspectos de GDB que pueden ser útiles para la programación de nuevas aplicaciones de depuración. Permite la introducción de puntos de ruptura, puntos de vigilancia y puntos de enganche.
- Permite la ejecución paso a paso.
- Permite la introducción de puntos de ruptura condicionales.
- Funciona en plataformas Linux, UNIX y Solaris.
- Permite la simulación de pequeños sistemas microprocesadores, como el microprocesador Zilog Z8000 ó Hitachi, por ejemplo.
- Permite la emulación ("in-circuit emulation") de microprocesadores de Hitachi, usando el emulador E7000 para desarrollar código para los microprocesadores Hitachi SH ó H8/300H.
- Permite la observación de la pila y determinadas variables del sistema.
- Permite la depuración de hilos y procesos.
- Es de tipo intrusivo, si se emplea el protocolo estándar de comunicación de GDB.
- Por medio del comando *target*, se puede cambiar el objetivo de la depuración, eligiendo entre depuración de procesos locales, ficheros locales, sistemas independientes a través de un puerto serie, o incluso sistemas en tiempo real a través de una conexión TCP/IP.
- *Formato de Código de Objeto:* Permite el cambio dinámico de formatos de tipo de fichero. Actualmente es posible este cambio dinámico entre ficheros de los siguientes tipos: COFF, ELF, a.out, Intel 960 b.out, MIPS ECOFF, HPPA SOM,

y S-records; los ficheros se pueden leer como fichero .o, bibliotecas, o salidas de tipo “core”.

- Se puede emplear para depurar programas escritos en C, C++, Modula-2 de GNU,

- Permite el uso de diferentes interfaces gráficos (Insight, DDD, etc), también de distribución gratuita. El interfaz gráfico DDD, por ejemplo, presenta este aspecto:

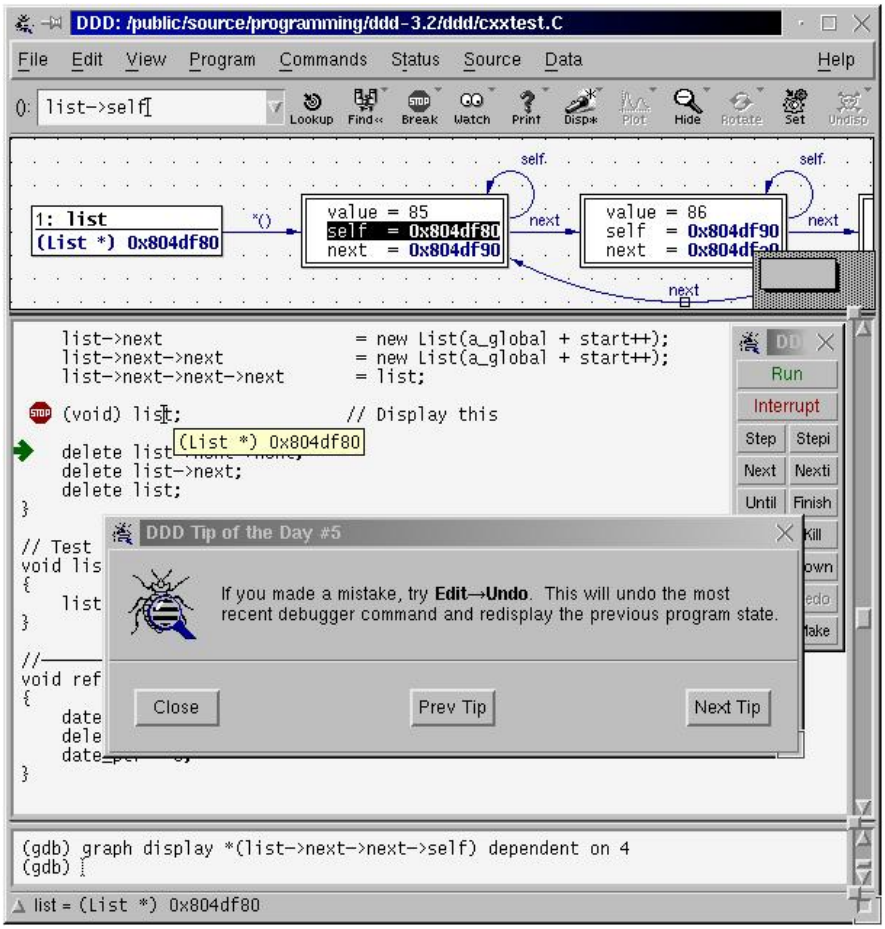


Figura 5.1.- DDD: Interfaz gráfico de usuario para GDB.

Las principales desventajas del depurador de código GDB son:

- Requiere un largo proceso de aprendizaje, debido a los conocimientos necesarios para instalarlo y ejecutarlo con éxito.

- Existen varias versiones algunas de las cuales no son estables.
- Hay alguna documentación sobre el uso del depurador, pero no hay ninguna sobre cómo realizar aplicaciones propias para GDB, debiendo alargar el proceso de aprendizaje todavía más, debido a la necesidad de leer el código fuente para comprender el funcionamiento del programa.
- Pese a indicar que es posible la ejecución paso a paso, ésta es una funcionalidad que no se encuentra actualmente implementada para la depuración de ningún sistema remoto, reduciendo la eficiencia de la depuración.
- Hay muy pocos sistemas para los que se han desarrollado totalmente las funcionalidades de GDB; en concreto, hay muchas funcionalidades que sólo están disponibles para un muy reducido grupo de microprocesadores.
- Cuando se comunica con un sistema que desea depurar por el puerto serie (como es el caso de nuestro proyecto), procede a emplear un protocolo de comunicación de tipo intrusivo, que puede no ser el adecuado para el sistema que se desea depurar (como ocurre en nuestro caso). Esto obliga a :
    - Desarrollar una aplicación propia que realice la comunicación con nuestro sistema.
    - Para ello, es necesario crear nuevos archivos y comandos para GDB, lo cual requiere reenlazar y recompilar todo el depurador.
    - Requiere asimismo la modificación de ciertos ficheros empleados por el protocolo de comunicación estándar de GDB, de forma que nuestro propio protocolo interactúe con el mismo de forma efectiva, reduciendo la longitud del código a desarrollar.

Como podemos ver, algunas de estas desventajas presentan grandes problemas cuya resolución representa un aumento importante del tiempo requerido para poder sacar provecho del depurador (¡e incluso para empezar a utilizarlo!).

Sin embargo, es el depurador más extendido basado en plataforma Linux, lo cual hace de él la mejor opción a la hora de decantarse por un depurador de propósito genérico cuando trabajamos con dicha plataforma.

5.5.2.- Depurador CodeWarrior, de Metrowerks.

Este depurador es el que la propia empresa Motorola propone para la depuración de código de sus sistemas microprocesadores.

Sus principales características son las siguientes:

- Es el depurador *"oficial"* de Motorola para la depuración de sus microprocesadores.
- Funciona en Win32 (W95, W98, Millennium, XP, W2000 ó NT 4.0) y sistemas Macintosh, así como una versión para Linux (aunque ésta está limitada a los micros ARM, x86, MIPS y PowerPC) .
- Permite emplear emuladores de tipo "in-circuit" y emuladores ROM. Al emplear estos sistemas, elimina la necesidad de emplear un monitor ROM en el sistema objetivo, aumentando de este modo la velocidad de descarga. Para poder hacer esto requiere la presencia de una hardware adicional.
- Permite emplear monitores ROM en el sistema objetivo, en caso de querer una solución más eficiente que la anterior en relación con el coste (aunque ocupa espacio de memoria en el sistema objetivo).
- Cuando se emplear el hardware adicional, puede depurar sistemas basados en PowerPC anidados a través de un puerto BDM, empleando un dispositivo de Macraigor Systems. Los sistemas que puede depurar de este modo son: PowerPC 8xx, 5xx, 403 y PowerQuicc II (8260).
- Permite la depuración de micros de las series V83x y VR de NEC, empleando emuladores ROM de Kyoto Microcomputer.
- Como podemos inferir de las características anteriores, es un sistema que puede funcionar de modo intrusivo o no, según el hardware y software de que hagamos uso.
- Puede funcionar de modo intrusivo empleado monitores ROM de libre distribución, como el PMON.

- Si funciona de modo intrusivo (por medio del monitor ROM MetroTRK que se facilita con el paquete básico de CodeWarrior), sus principales características son las siguientes:

    - Controlar las fuentes del procesador y de la tarjeta.
    - Lectura y escritura en registros individuales.
    - Observación y alteración de la memoria.
    - Carga de programas desde la plataforma de depuración.

- Permite la depuración de los siguientes sistemas: ARM, Motorola (68K, Coldfire, Suite 56 DSP 563xx/DSP 566xx, HC05, HC(S)08, HC11, HC(S)12, HC16, M·CORE, MPC500/5500/5200, MPC60x/7xx/74xx, PowerQUICC I/II/III, StarCore).
- Formatos específicos de arquitecturas (ELF, STABS, S-Records para Motorola, DWARF).
- Depurador de nivel de aplicación de Linux Anidado (para determinadas versiones).
- El depurador soporta ensamblador, C y C++.
- MetroTRK incluido con las fuentes para muchas tarjetas de evaluación comunes.
- Depuración multi-proceso y multi-hilo.
- Desensamblador disponible en el IDE.
- Depuración en modo fuente, ensamblador o mezcla de ambos, que además puede hacerse en modo gráfico.
- Tiene puntos de ruptura condicionales, aparte de los normales.
- Admite evaluación de expresiones.
- Puede mostrar y editar valores de variables, registros y rangos de memoria.
- El control de ejecución ejecuta líneas de código o instrucciones de ensamblador en cualquier localización.
- Permite conexiones de tipo Ethernet y protocolos de puerto serie para la conexión con sistemas externos.

Como podemos ver, en lo que se refiere a nuestro proyecto este sistema presenta los siguientes inconvenientes:

- La versión que utiliza Linux está limitada a unos pocos procesadores entre los cuales no se encuentra la familia que nos interesa, el M·CORE.
- Requiere de la monitorización ROM (MetroTRK) ocupando parte de la memoria del objetivo, lo cual va en contra de las condiciones de partida del proyecto. En caso de emplear un "dongle" hardware, podríamos eliminar esta desventaja, pero entonces subiría el precio.
- No es gratuito.

Sin embargo, podemos ver que es el que más se adecuaría a nuestras necesidades en caso de poder trabajar en una plataforma diferente (Win32 en lugar de Linux), aunque a un cierto precio. También podemos observar que el periodo de aprendizaje, el mayor problema en el caso del depurador GDB, se convierte en un problema menor cuando se emplea este depurador, debido al soporte de la propia empresa, así como a la gran cantidad de opciones que ofrece.

5.5.3.- Depurador UPS.

El depurador UPS es un depurador de código para C, C++ y Fortran, desarrollado de forma gratuita inicialmente por Mark Russell en la Universidad de Kent, en Canterbury.

Actualmente se encuentra en su versión 3.37.

Las principales características de este depurador son las siguientes:

- Establecimiento de puntos de ruptura: Se pueden establecer puntos de ruptura de la manera usual, asociados a la utilización de una determinada función, por ejemplo.

- Puntos de ruptura condicionales y bajo comando: Al igual que cuando se emplea el depurador de GDB.
- Ejecución paso a paso.
- Emplea un interfaz gráfico de usuario:

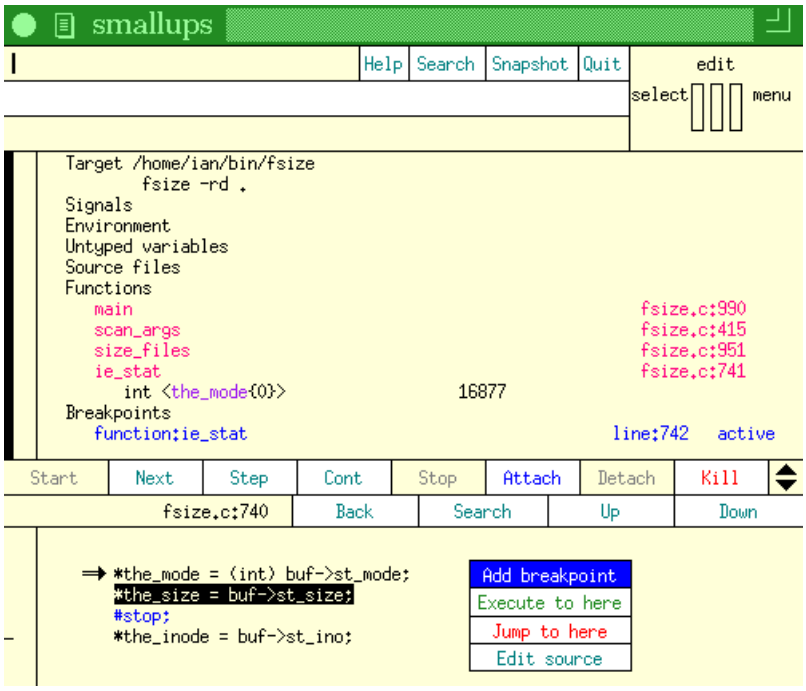


Figura 5.2: Interfaz gráfico del depurador UPS.

- Examen de variables.
- Muestra de los cambios en la pila: Se almacenan los cambios en la pila cada vez que se produce uno de ellos, de forma que cuando se detiene el programa, observando la pila podemos reconstruir los pasos dados por el programa.
- Permite la congelación de variables: Introduciendo un valor en una variable a través del interfaz gráfico, podremos ejecutar el programa completo con esa variable congelada en ese valor, y ver qué ocurre durante la ejecución.
- Posee un intérprete de C, que te permite añadir código sin más que escribir en la ventana correspondiente. Con esto se pueden añadir funciones de llamada de tipo printf sin necesidad de recompilar, reenlazar ni reiniciar el programa original.

- También funciona para plataformas Solaris, Fortran, SunOS en SunSPARC y FreeBSD.
- No tiene simulador ni emulador, sólo se emplea como enlace con la plataforma.
- Es de tipo intrusivo, puesto que presenta su propio protocolo de comunicaciones por el puerto serie, el cual requiere de un monitor en el sistema objetivo.

La principal ventaja de este depurador es que se puede obtener de forma gratuita, y está en continua evolución. La principal desventaja es que por estar en continua evolución, el programa puede ser poco fiable, por lo que adolece del mismo defecto que el depurador GDB.

Además, las características de este depurador indican que su finalidad tiene más relación con la depuración entre sistemas similares que para sistemas que no sean parecidos entre sí. No tiene implementada ninguna utilidad o capacidad que le permita depurar sistemas que no tengan implementada de alguna manera un monitor ROM, de modo que tampoco nos resulta de utilidad en este aspecto.

Por último, el programa tiene una difusión y seguimiento mucho menor que el depurador GDB, lo cual hace que sea más difícil encontrar documentación completa sobre el mismo, o nuevas funcionalidades desarrolladas por usuarios profesionales para este depurador.

5.5.4.- Entorno de Desarrollo Integrado MULTI®, de Green Hills Software Inc.

El depurador MULTI de Green Hills es un potente depurador de código en entorno gráfico que permite la carga del programa, la ejecución, control de ejecución, y monitorización.

Las principales funcionalidades de este sistema son las siguientes:

- Provee depuración en los lenguajes C, C++, C++ anidado, Ada 95, Fortran, y en código ensamblador.
- Permite la visualización de datos durante la ejecución, enlaces directos al Constructor de Proyectos MULTI, Buscadores, Comprobación de Errores en tiempo real, menús configurables, botones de comandos, y un interfaz abierto para editores y software de control de versiones.
- Además, este depurador es recomendado por la propia empresa Motorola para la depuración de código de sus sistemas microprocesadores.
- Permite la depuración de programas escritos en ensamblador para los siguienes sistemas microprocesadores: PowerPC, M·CORE, ARM/Thumb, ST100, i960, Alpha, RH32, 680x0, 683xx, StarCore, StrongARM, V8xx, SH, RAD6000, FR, CPU32, ColdFire, MIPS, x86/Pentium, TriCore, SPARC/SPARCLite.
- Puntos de ruptura condicionales y bajo comando: Al igual que cuando se emplea el depurador de GDB.
- Ejecución paso a paso.
- Examen de variables.
- Establecimiento de puntos de ruptura: Se pueden establecer puntos de ruptura de la manera usual, asociados a la utilización de una determinada función, por ejemplo.
- Muestra de los cambios en la pila: Se almacenan los cambios en la pila cada vez que se produce uno de ellos, de forma que cuando se detiene le programa, observando la pila podemos reconstruir los pasos dados por el programa.
- El interfaz gráfico que presenta es bastante sencillo, y tiene la siguiente estructura:

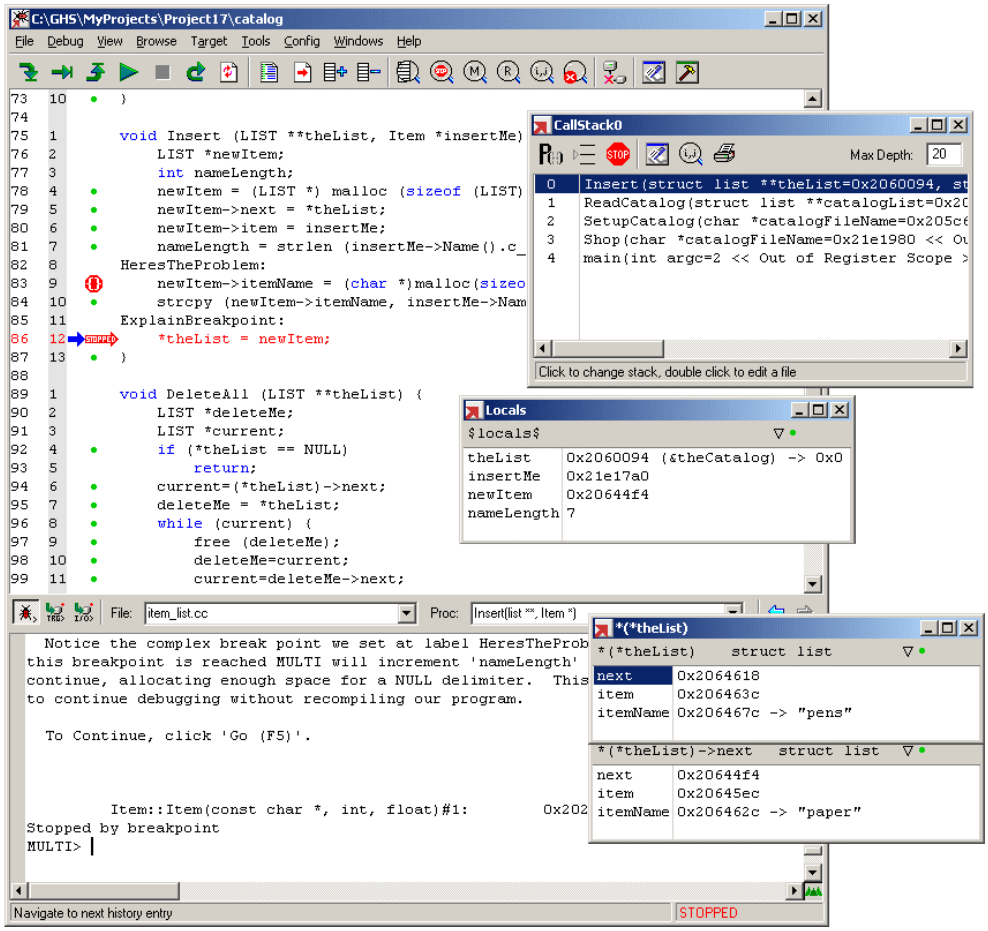


Figura 5.3.- Depurador del IDE MULTI, de Green Hills Software.

- Ventanas de exploración: Pulsar en una variable crea una ventana que muestra la variable, su tipo, y el valor actual. Esta variable puede ser de cualquier tipo. La ventan se actualiza de forma automática cada vez que el programa se detiene. Si la ventana contiene un puntero, una doble pulsación en el mismo abrirá una nueva ventana en la que se mostrará la variable a la que se apunta con el mismo.
- Depuración incremental.
- No tiene emulador ni simulador (a menos que se disponga de los servidores ISS adecuados para conectarlos de la siguiente manera, los cuales pueden ser desarrollados por los usuarios, o se pueden utilizar los que ya han sido desarrollados por Green Hills Software):

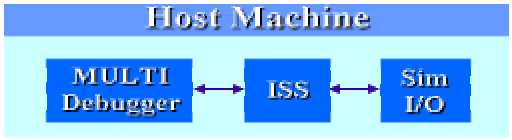


Figura 5.4.- Esquema básico de conexión con emulador y simulador en el IDE MULTI, de Green Hills Software.

- Funciona bajo plataformas Linux, SPARC/Solaris, Windows, PA-RISC/HP-UX.
- No es de distribución gratuita.
- Es de tipo intrusivo, puesto que requiere la presencia de un programa monitor ROM en el sistema objetivo, conectado con el servidor ISS.
- Puede ser de tipo no intrusivo (en el caso de los micros de la familia M·CORE), si se hace uso de una sonda de pruebas de procesadores de Agilent Technologies:

Figura 5.5.- Sonda de prueba de microprocesadores E5900B, de Agilent Technologies

- Funciona en plataformas UNIX, W95,W98, NT, W2000.
- La sonda de pruebas también puede emplearse para conseguir la emulación del sistema, al conectarlo a un analizador lógico por medio de la misma.

Por las características que nos muestra este sistema de desarrollo, podemos ver que es una alternativa viable a CodeWarrior en caso de emplear un sistema Win32, o bien una alternativa viable a GDB en el caso de sistemas basados en Linux. Tiene prácticamente las mismas características de los otros dos depuradores, con la ventaja de

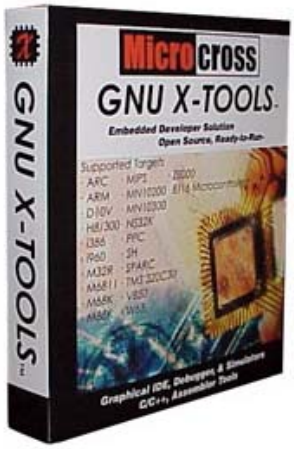
que además tiene ya implementados los elementos necesarios para trabajar con los microprocesadores de la familia M·CORE.

En el caso de compararlo con CodeWarrior, podemos ver que en ambos depuradores se puede realizar la depuración de modo no intrusivo o intrusivo, de pendiendo de que se emplee una sonda especial o no, respectivamente. Sin embargo, esto presenta un coste asociado. Además, presenta la ventaja de ser de fuente abierta, al contrario que el CodeWarrior, lo cual permite una mayor flexibilidad en la programación de nuevas aplicaciones.

Si lo comparamos con GDB, nos encontramos con que disponemos de gran parte del proceso de desarrollo ya realizado, un tiempo de aprendizaje menor, y una documentación más extensa y precisa. Como contrapartida, GDB nos ofrece más formas de acceder al sistema, aunque requiere que el usuario realice la programación por sí mismo.

En general, parece que este depurador es el más polivalente de todos; puede funcionar en diferentes plataformas y con menos limitaciones que las que tiene CodeWarrior, nos da una gran cantidad de documentación precisa (al contrario que GDB, en el que la información se encuentra esparcida por toda la red, y cuya cantidad y exactitud no son del todo claras), nos permite la emulación y simulación (adquiriendo los elementos necesarios a un precio adicional, al igual que en el CodeWarrior), y por último nos permite realizar nuestras propias aplicaciones con mayor libertad, pro ser de fuente abierta.

5.5.5.- GNU X-Tools, de Microcross.



Este paquete de software se puede obtener en dos versiones; la primera de ellas es la versión 2.0, y la segunda es la versión 3.0.

- La versión 2.0 permite el manejo de fichero de formato ELF y COFF muchos sistemas, entre ellos los microprocesadores de Intel i386/i486/Pentium/i960, Atmel, ARC, NEC V850x, Panasonic MN10200 y MN10300, Sparc, SparcLite, UltraSparc, Motorola M·CORE, etc.

- La versión 3.0 igualmente se emplea para otros sistemas microprocesadores, como Motorola/IBM Power PC, Motorola CPU32, M68K y ColdFire, MIPS 16/32, Intel Xscale y ARM/Thumb, para los mismos formatos.

- Emplea un interfaz gráfico para facilitar la tarea del programador.

- Permite realizar la introducción de puntos de ruptura

- Permite examinar variables

- Permite un seguimiento continuo de los valores de la pila.

- Las fuentes son abiertas, de forma que cualquier modificación que sea requerida por el usuario puede ser implementada por éste siguiendo el tutorial de diseño de nuevos módulos que presenta este programa.

- Sin embargo, no es de distribución gratuita, debido a que el apoyo que una empresa presta a sus usuarios es muy superior al que pueda dar la comunidad GNU a un único programador, además de ser más fiable.
- Es de tipo intrusivo, pero si se utilizan herramientas como EPI MAJIC JTAG para la depuración hardware a través del puerto JTAG, se puede convertir en no intrusivo, al menos en el caso de los microprocesadores ARM
- No permite la simulación, salvo en el caso de los microprocesadores ARM, que son los únicos que han sido implementados por el momento para este IDE.
- En principio, no permite la emulación, a menos que se emplee hardware adicional, como el dispositivo BDI1000 de Abatron:

Figura 5.6.- Dispositivo de emulación BDI1000, de Abatron.

- Funciona en plataformas Linux, Solaris y Windows.

Al igual que al comentar las características y funcionamiento del entorno de desarrollo integrado (IDE) MULTI de Green Hills Software, nos encontramos con otro paquete que nos ofrece las mismas características y capacidades. En general, podemos realizar las mismas afirmaciones en este caso, al compararlo con los depuradores CodeWarrior y GDB, con una excepción. El X-Tools presenta apoyo para el depurador GDB dentro de su estructura, lo cual posibilita la utilización del depurador GDB en determinadas aplicaciones, mientras que el MULTI no presenta tal posibilidad.

5.5.6.- Depurador XRAY, de Mentor Graphics.

El depurador XRAY de Microtec/Mentor Graphics es un depurador comúnmente empleado para la depuración de sistemas de 32 y 64 bits.

Las principales características de este depurador son las siguientes:

- Presenta un prototipo de aplicación en un simulador con un juego de instrucciones definido, para realizar pruebas de software.
- OCD (Depuración en micro): Dispositivo de depuración a través del puerto JTAG por medio del uso de diferentes dispositivos hardware, como la sonda Agilent, la sonda de Macraigor Systems, la de Abatron o la MAJIC, de las que ya hemos hablado en los anteriores depuradores (aunque la elección de la sonda depende del sistema que se desee depurar).
- Programa monitor software residente en el sistema objetivo, que facilita la depuración del programa deseado.
- Co-verificación Seamless hardware/software.
- Soporta lenguajes C y C++, así como el ensamblador de determinados micros de 32 y 64 bits.
- Permite control absoluto sobre la ejecución de programas.
- Permite la manipulación de datos.
- Se pueden examinar variables del código bajo depuración, y modificarlas en alto (C/C++) o bajo nivel (ensamblador).
- Se puede emplear una lista de comandos o bien un interfaz gráfico como el que mostramos a continuación:

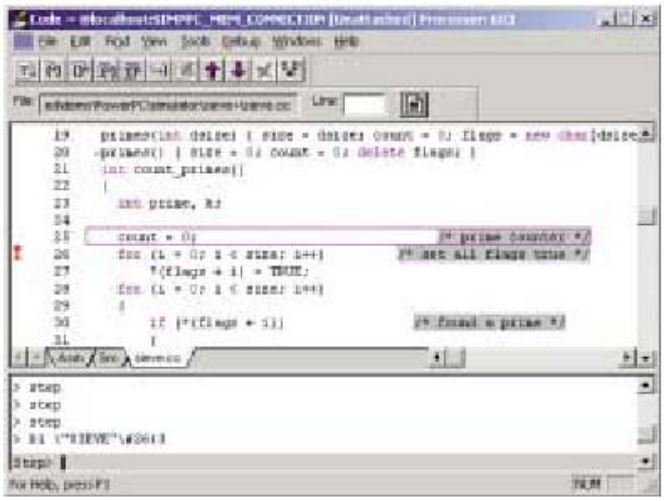


Figura 5.7.- Pantalla del editor de textos del código fuente.

- Permite la observación de la pila, memoria y registros por medio de ventanas, todos los cuales son accesibles al programador para su modificación.
- Permite la descarga de imágenes ejecutables, incluyendo la descarga de símbolos sólo para la memoria ROM, o para la memoria FLASH.
- Permite la ejecución paso a paso tanto a nivel de instrucción como a nivel de código fuente.
- Permite la vuelta a estados anteriores de la pila.
- Permite puntos de ruptura software y hardware.
- Permite la deshabilitación/habilitación temporal.
- Permite la introducción de puntos de ruptura condicionales.
- Permite la corrección de errores de forma inmediata por medio del editor de textos.
- Este depurador se puede instalar en las siguientes plataformas: Sun Solaris, HP UX, W95/98/2000/NT.
- Permite la emulación de micros.

Como podemos deducir de la relación de funcionalidades, de nuevo nos encontramos con un depurador capaz de funcionar como intrusivo o no, según empleemos un hardware adicional. Asimismo, vemos que presenta aproximadamente

las mismas capacidades que otros sistemas de depuración profesionales ya comentados, gracias al empleo del paquete de apoyo de microprocesadores de Motorola, Seamless Co-verification, que le permite añadir todos los micros importantes de Motorola (680x0, M·CORE, ColdFire, etc). Asimismo, presenta otros paquetes de apoyo para los micros de las diferentes empresas fabricantes.

5.6.- Conclusión de la comparativa.

En general, todos los depuradores de la comparativa nos ofrecen prestaciones similares, entre las cuales las más importantes son si la depuración es intrusiva o no y si permite la emulación y simulación de los sistemas bajo depuración.

En el caso de la depuración intrusiva, nos encontramos con que todos funcionan por defecto con algún tipo de monitor ROM en el sistema objetivo, aunque se presenta la opción (en la mayor parte de los casos) de convertirla en no intrusiva por medio de la adición de hardware adicional por el puerto serie, conectándolo al puerto JTAG del sistema objetivo.

En el caso el presente proyecto, la utilización de tal elemento hardware adicional no es posible, debido a la presencia del sistema JDONG, que actúa como tal; Puesto que todos los dispositivos hardware propuestos en la comparativa emplean el puerto serie para conectarse a un PC y el puerto JTAG del sistema para conectarse al sistema objetivo, pero el JDONG requiere esos mismos puertos imposibilitando la utilización de los mismos.

Todo esto nos deja en su mayor parte con sistemas intrusivos, salvo aquellos que disponen de fuentes abiertas, lo cual nos permitiría realizar una interfaz software entre el resto del software del entorno de desarrollo y el JTAG a través del sistema JDONG, que es lo que ha hecho en el presente proyecto.