Objektorientierte Programmierung…

Die Popularität von objektorientierten Programmiermethoden macht auch vor Embedded Applikationen nicht halt. Natürlich muß ein leistungsfähiger Cross-Debugger auch in dieser Welt zu Hause sein. So kennt der XRAY Debugger sämtliche aktuellen Spracheigenschaften von C++, wie überladene Operatoren und Funktionen, Konstruktoren und Destruktoren und Vererbung. Soll ein Breakpoint auf eine überladene Funktion gesetzt werden, so bietet XRAY eine Liste der möglichen Funktionen zur Auswahl an. Die exakte oft komplizierte Parameter-Syntax braucht man jetzt nicht mehr im Kopf behalten.

…und wenn's sein muß, auch Assembler

Embedded Applikationen beinhalten nach wie vor auch Assembler-Anweisungen. Im XRAY Debugger kann Hochsprache gemischt mit der Assemblerebene dargestellt werden (Bild 2). Der Debugger sorgt für die genaue Zuordnung der beiden Welten. Anstatt Objektcode einfach nur zu disassemblieren, verwendet XRAY Debug-Informationen die vom Assembler generiert wurden. Die Arbeit auf Maschinenebene ist dadurch wesentlich vereinfacht, weil sie symbolisch unterstützt ist. Das StepOver-Kommando verhindert, daß immer wieder Makros oder Repeat-Blöcke schrittweise durchwandert werden müssen.

Hardware noch nicht komplett?

PC-Programme zu bearbeiten ist einfach, die Hardwareumgebung ist immer gleich. Bei Embedded Applikationen ist die Peripherie jedoch meist sehr unterschiedlich aufgebaut. Ein sehr guter Cross-Debugger muß dem Entwickler die Möglichkeit geben, schon vor Fertigstellung der Hardware Programme möglichst vollständig auszutesten. Mit einem Instruction-Set Simulator, wird der Zielprozessor auf der Host-Maschine nachgebildet. Natürlich muß hier periphere Hardware berücksichtigt werden. XRAY bietet eine Vielfalt an Simulationsmöglichkeiten für Input/Output-Ports oder Interrupts (Bild 4).

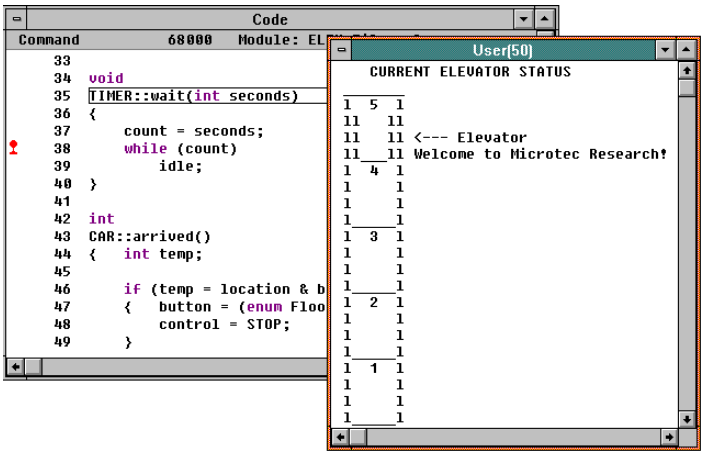

Bild 4: **Hardwarenachbildung im XRAY-Debugger: Die Funktionalität dieser C++ Fahrstuhlsteuerung läßt sich mit Semi-Grafik-Unterstützung noch einfacher überprüfen**

Hardwarenahe Programmteile, die zum aktuellen Zeitpunkt noch nicht so interessant sind, können in XRAY umgangen werden, ohne eine Zeile am Quellkode zu manipulieren.

Umgekehrt können Funktionen der Applikation wie ein Debugger-Kommando einzeln im Zielsystem zur Ausführung gebracht werden (Target-Function-Calls).

Makrosprache und noch viel mehr…

Der XRAY Debugger kann mittels Makrosprache funktionell beliebig erweitert werden. Die Sprache ist nicht kompliziert und neu, sondern basiert auf normaler C-Syntax und kann auf sämtliche Symbole der Applikation zugreifen (BILD5). Makro-Anwendungen reichen von automatisierten Debugging-Läufen, über Test-Scripts, bis zu neuen Debugger-Funktionen. Mittels Makros können Codebereiche einfach auf Debugger-Ebene gepatcht werden, ohne fehleranfällig im Source-Code manipulieren zu müssen.

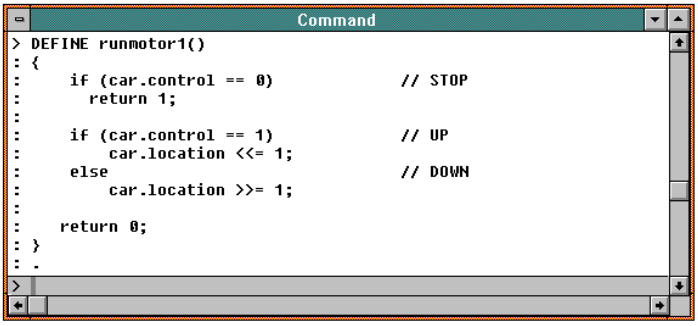

Bild 5: **Die Makro-Sprache ist in simpler C-Syntax gehalten**

Sicher in die Zukunft

Über Debugger-Features hinaus bietet XRAY aber noch viel mehr Funktionalität, um ein Embedded-Software Design zu optimieren. Stichworte wie Performance- und Code-Coverage-Analyse oder RTOS-System-Level-Debugging wären Themen für eigene umfangreiche Abhandlungen. Der XRAY-Debugger kann auch im Batch-Mode betrieben werden und z.B. im Feld in der Schaltung über längere Zeiten Daten loggen, die dann im Labor z.B. am Simulator ausgewertet werden.

Erhältlich ist der XRAY-Debugger in zahlreichen Simulator- und In-Circuit-Varianten für unterschiedliche Prozessorfamilien, wie Motorola 68K, PowerPC, Intel 80x86-kompatible und läuft auf den Host-Plattformen SUNSparc, HP9000/700 und PC-386/486/Pentium.

Für weitere Auskünfte kontaktieren Sie bitte

Herrn Schuster
Microtec Research GmbH
Haidgraben 1c
D-85521 Ottobrunn/München
Tel.: +49-89-609 00 81

□