# Mehr als nur ein Cross-Debugger

Andreas Pfeiffer

Die steigende Komplexität von Embedded Applikationen erfordert immer leistungsfähigere Entwicklungswerkzeuge um neue Produkte rechtzeitig auf den Markt zu bringen. Microtec Research ist seit 20 Jahren auf die optimale Unterstützung von Softwareentwicklern in der Embedded Welt spezialisiert und bietet als einziger Hersteller eigene Technologie vom Compiler über unterschiedliche Debugger bis zu Echtzeit-Betriebssystemen. Der XRAY Debugger setzt seit Jahren den Standard bei Cross-Debuggern. Abgestimmt auf die aktuellen Bedürfnisse, ob hostbasierender Simulator oder Cross-Umgebung, sind alle Funktionen implementiert, die einen rascheren Entwicklungszyklus ermöglichen und noch vieles mehr...

## Grundbedürfnisse gestreßter Entwickler

Leistungsfähige Standardfunktionen zeitgemäßer Cross-Debugger sind natürlich auch im XRAY-Debugger implementiert:

- Stack-Traceback: Überblick der Stack-Verschachtelung auf Hochsprachenebene und Einsicht auf die lokalen Variablen auf jeder beliebigen Stack-Ebene
- Single-Step: auf Hochsprachen- und/oder Assembler-Ebene
- Flexible Darstellung und Manipulation von Variablen, Objekten, Register und Codebereichen
- Symbolischer Zugriff auf sämtliche Bereiche der Applikation
- Einheitliche, grafische und intuitiv bedienbare Multi-Windows Bedienoberfläche
- Kompatibilität zu Vorgängerversionen
- Viele Varianten mit durchgängiger Bedienphilosophie (Simulator, Incircuit-Monitor, Incircuit-Emulator, BDM)

Embedded Systeme laufen fast immer stand-alone und müssen daher extrem robust konstruiert werden. Viele gängige Debugger haben den Nachteil, daß sie nur un- bzw. halboptimierten Objektcode des Compilers verarbeiten können, weil z.B. Codeverschiebungen oder Registerlebenszyklen nicht exakt nachverfolgt werden können. Mit dem XRAY Debugger kann immer volloptimierter Objektcode verarbeitet werden. Unliebsame Überraschungen gegen Ende des Entwicklungszyklus werden so von Anfang an vermieden.

Moderne Applikationen basieren auf vielen tausend Zeilen C bzw. C++ Source-Code. Microtec Research hat sehr viel Know-how investiert, um die Arbeit auf Hochsprachenebene zu vereinfachen. Ein Ergebnis daraus ist das Inspektor-Fenster zur übersichtlichen Darstellung von komplexen Datenstrukturen. Die automatische Verarbeitung von dynamischen Datenstrukturen (Listen, Bäume etc.) hilft fehlerhafte Verweise und Einträge sehr einfach zu finden.

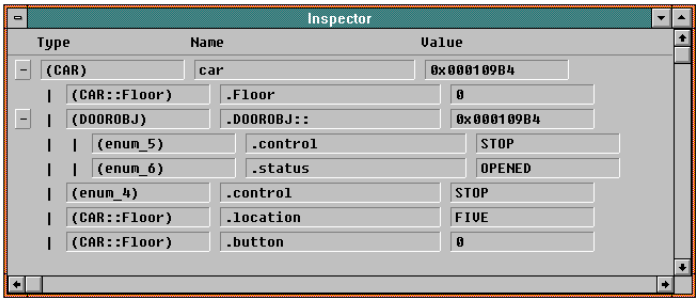


*Bild 1:* **Das XRAY Inspektor-Fenster erlaubt die einfache Darstellung und Manipulation von komplexen Datenstrukturen**

Eine sehr nützliche Eigenschaft von XRAY betrifft die Preprozessor-Unterstützung. Werte, die durch **#define** zugewiesen wurden, können wie Variablen dargestellt werden. Das lange Suchen nach der aktuellen Zuweisung in vielen Header-Dateien hat somit ein Ende.

Einzigartig bei XRAY ist, daß das schrittweise Durchlaufen der Applikation befehlsorientiert anstatt zeilenorientiert erfolgt. **if**, **while** und **do** Konstrukte werden exakt durchschritten und Breakpoints können in der entsprechenden Spalte auf Hochsprachenebene gesetzt werden. Bei vielen Debuggern muß noch auf Assembler-Ebene gewechselt werden, um Breakpoints z.B. auf den Bedingungs-Teil einer **for**-Schleife zu setzen.

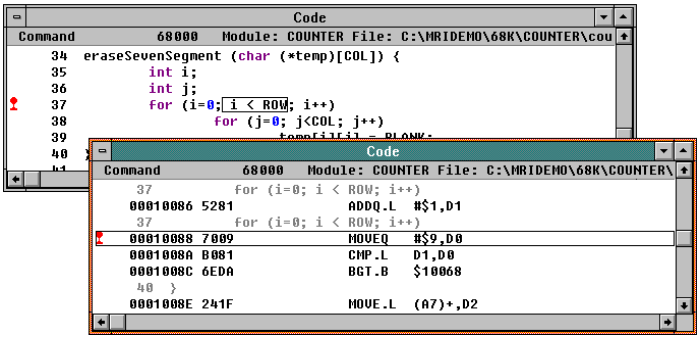


*Bild 2:* **Der XRAY-Debugger arbeitet befehlsorientiert. Breakpoints können auch innerhalb von for-Anweisungen gesetzt werden.**

## Breakpoints: Wie hätten Sie's denn gerne?

Einfache Instruktions-Breakpoints sind die Grundvoraussetzung für die Arbeit mit einem Debugger. Breakpoints bei Speicherzugriffen sollten auch nach Schreib- oder Lesezyklus unterschieden werden können. Natürlich muß auch eine mögliche Hardwareunterstützung durch den Zielprozessors (BP-Register) Verwendung finden.

Die gefürchtetsten Bugs sind jene, die nur sehr selten auftreten. Die Triggerbedingungen in der Umgebung solcher Fehler sind meist sehr komplex. Der XRAY-Debugger kann über eine leistungsfähige Makrosprache Abbruchbedingungen quasi frei programmierbar und beliebig komplex qualifizieren.

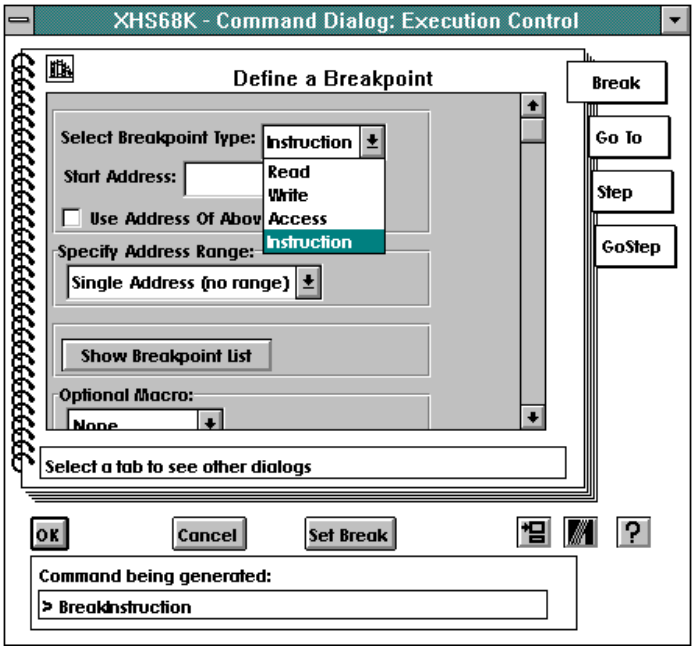


*Bild 3*: **Mit grafischer Unterstützung durch das Notebook-Interface können auch komplexe Breakpoints sehr einfach gesetzt werden**

Und oft möchte man die Applikation nur schnell bis zu einer bestimmten Zeile laufen lassen. In XRAY werden diese temporären Breakpoints per Point-and-Click angelaufen.

## 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).

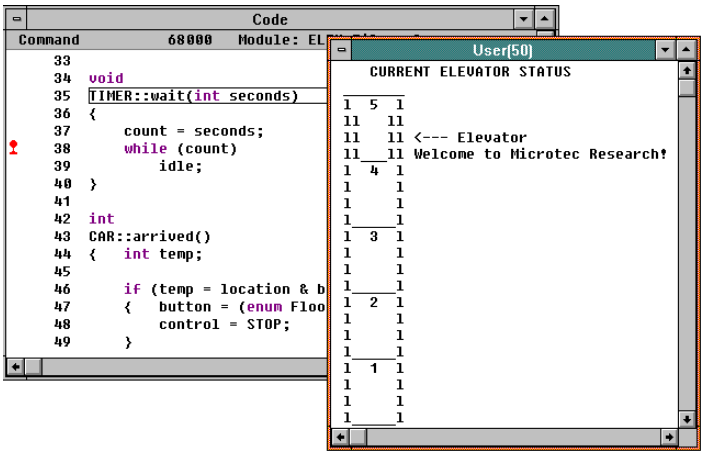


*Bild 4*: **Hardwarenachbildung im XRAY-Debugger: Die Funktionalität dieser C++ Fahrstuhlsteuerung läßt sich mit Semi-Grafik-Unterstützung noch einfacher überprüfen**

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.

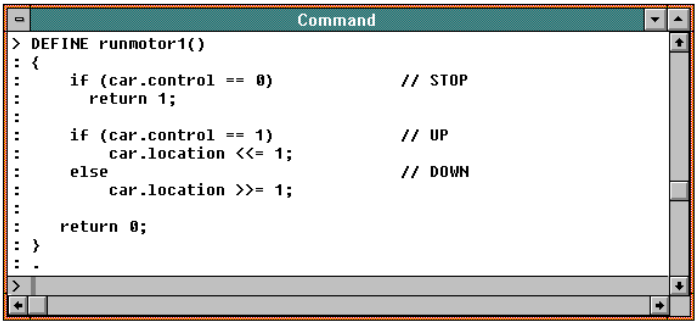


*Bild 5:* **Die Makro-Sprache ist in simpler C-Syntax gehalten**

## 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*

□