Zielgenaue Traceauswertungen für Cortex-M3/M4

Fehlerbehebung, Performance-Tuning, Code-Coverage, all das lässt sich für ein Embedded System schnell durchführen, wenn einem die dafür notwendigen Traceauswertungen zur Verfügung stehen. Um für die Cortex-M3/M4 Prozessoren optimale Traceauswertungen anbieten zu können, hat Lauterbach 2011 neue Wege beschritten.

ETM und ITM zusammenführen

Für die Cortex-M3/M4 Prozessoren können Traceinformationen von zwei verschiedenen Quellen generiert werden (siehe Bild 3).

- **ETMv3:** Generiert Informationen zur Instruktionsausführung.
- **ITM:** Generiert Informationen über die Schreib-/Lesezugriffe mittels der *Data Watchpoint and Trigger Unit (DWT)*.

Die ITM-Tracepakete zu den Schreib-/Lesezugriffen enthalten folgende Informationen:

- Datenadresse
- Datenwert, gelesen bzw. geschrieben
- Program Counter

Unter Auswertung des *Program Counter* lassen sich die separat generierten Datenzugriffe nahtlos in den Programmablauf integrieren (siehe Bild 1), was zu einer wesentlichen Vereinfachung der Fehlersuche führt. Die Ursache dafür, dass ein falscher Datenwert auf eine Speicheradresse geschrieben wurde, lässt sich schneller ermitteln, wenn die Schreibzugriffe eingebettet in den gesamten Programmablauf dargestellt werden.

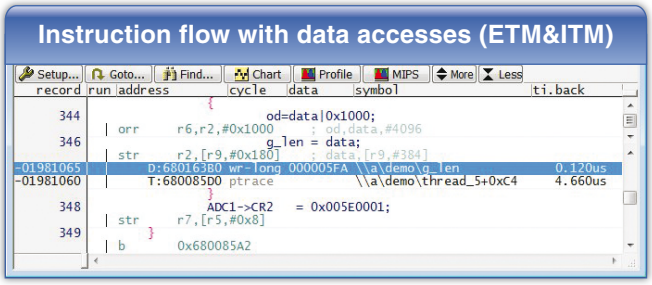

Bild 1: Durch das Zusammenführen von ETM- und ITM-Tracedaten lassen sich Schreib-/Lesezugriffe nahtlos in den Programmablauf integrieren.

OS-aware Tracen

Wenn auf dem Cortex-M3/M4 ein Betriebssystem läuft, sind Taskwechsel-Informationen für die Traceauswertung

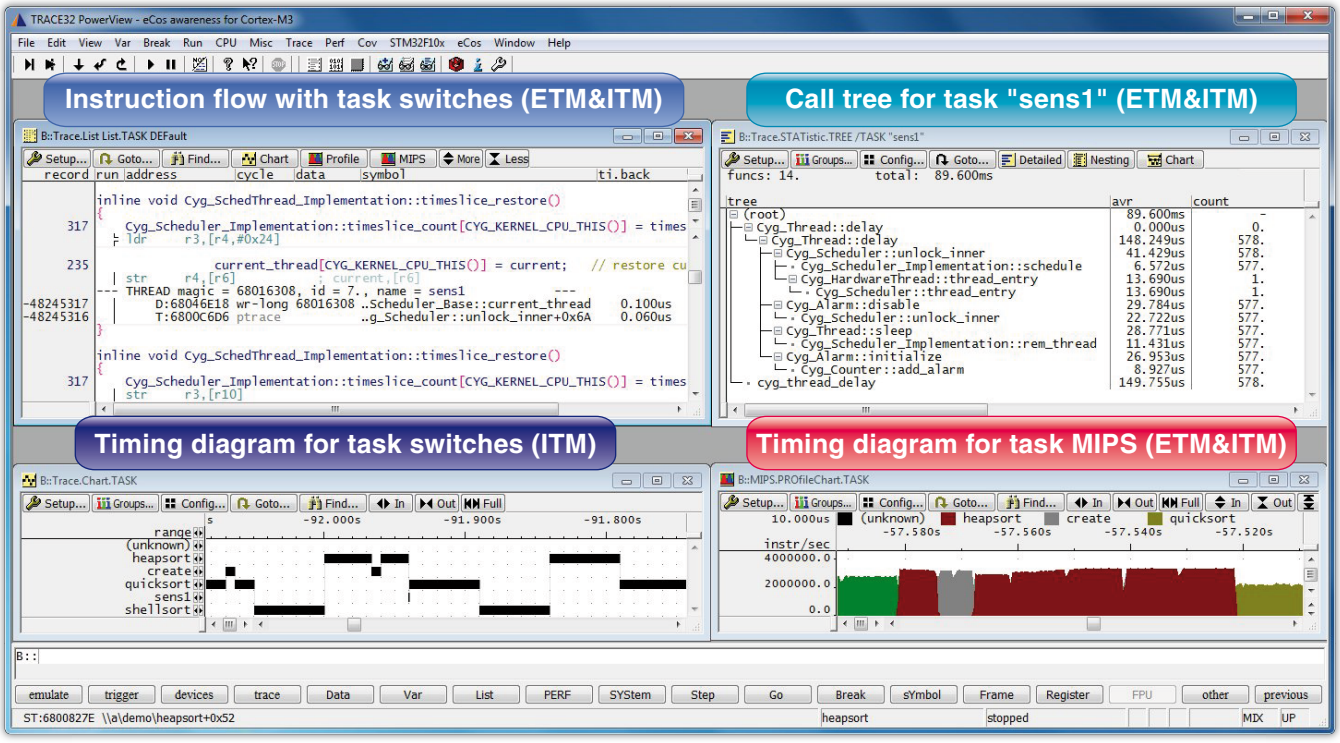

Bild 2: Durch das Zusammenführen von ETM- und ITM-Tracedaten können für das Betriebssystem eCos umfassende Traceauswertungen durchgeführt werden.

unverzichtbar. Eine Möglichkeit, Informationen über Taskwechsel zu erhalten, ist folgende: Mittels der ITM werden Traceinformationen für den Schreibzyklus generiert, in dem der Kernel die Kennung für den aktuellen Task auf die entsprechende OS-Variable schreibt. Als Schreibzugriff lässt sich der Taskwechsel, wie zuvor beschrieben, nahtlos in den Programmablauf integrieren. Dies verbessert die Lesbarkeit des Trace-Listings (siehe Bild 2). Die Integration der Taskwechsel in den Programmablauf bildet auch die Basis für die in Bild 2 gezeigten Laufzeitanalysen.

Drei Aufzeichnungsmodi

Für die Aufzeichnung der von den Cortex-M3/M4 Prozessoren generierten Traceinformationen unterstützt Lauterbach drei Modi:

- **FIFO-Modus:** Abspeichern der Information im 128 MByte Speicher der TRACE32 CombiProbe.
- **STREAM-Modus:** Streamen der Information auf die Festplatte des Hostrechners.
- **Real-time Profiling:** Die Traceinformation wird auf den Hostrechner gestreamt und dort live ausgewertet.

Bei den ersten beiden Aufzeichnungsmodi wird die Traceinformation zunächst erfasst. Die Traceauswertung kann man erst nach Abschluss der Aufzeichnung durchführen.

Jeder Aufzeichnungsmodus hat seine spezielle Zielsetzung. FIFO ist sicher der gebräuchlichste Modus. Er ist schnell und genügt in der Regel für die Fehlersuche und für Laufzeitanalysen.

Die auf Cortex-M3/M4 Prozessoren implementierte ETMv3 verfügt weder über Trigger noch Tracefilter. Daher ist es nicht möglich, die Tracequelle so zu konfigurieren, dass diese nur Informationen für den Programmabschnitt generiert, der von Interesse ist. Dies kann dazu führen, dass für die Fehlersuche Tracedaten eines relativ langen Zeitraums aufgezeichnet und ausgewertet werden müssen. Hier ist der STREAM-Modus die richtige Wahl. Der STREAM-Modus stellt allerdings höhere Anforderungen an die Debug-Umgebung:

- Die große Datenmenge, die durch das Streaming entsteht, erfordert ein 64-Bit TRACE32 Executable. Nur so lassen sich die Traceeinträge adressieren.
- Die Übertragungsrate zwischen CombiProbe und Hostrechner muss ausreichen, um alle Tracedaten verlustfrei zu streamen. Der 128 MByte Speicher der CombiProbe puffert dabei Lastspitzen am Traceport (TPIU) ab.

Real-time Profiling ist besonders geeignet, um *Statement* und *Condition Coverage* durchzuführen, da man die Analyse live auf dem Bildschirm mitverfolgen kann. Für bedingte Anweisungen, für die bisher nur der Fail-Zweig

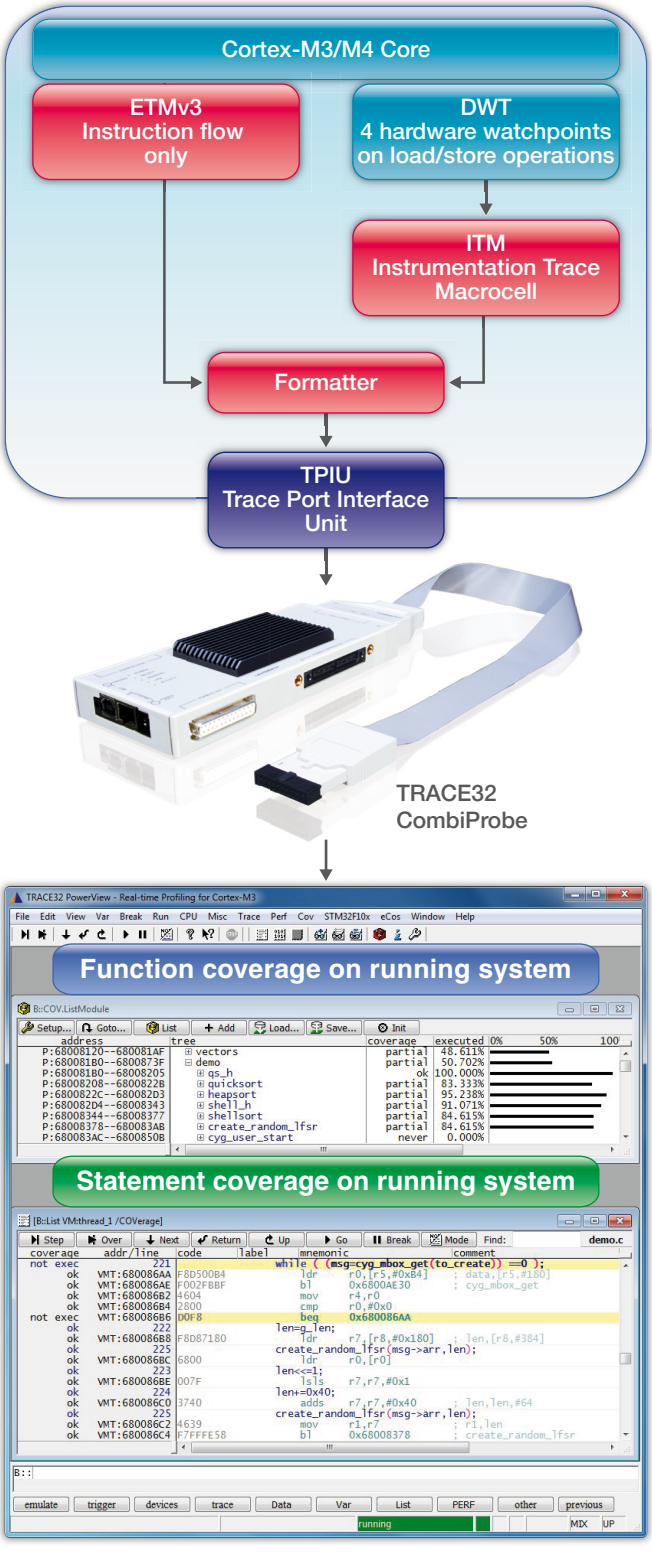

Bild 3: Real-time Profiling erlaubt es, die Code-Coverage-Analyse live auf dem Bildschirm mitzuverfolgen.

durchlaufen wurde (gelb unterlegt in Bild 3), kann man das Embedded System gezielt so stimulieren, dass auch der Pass-Zweig zur Ausführung kommt.