

Debugging mit Cortex-M3-Mikrocontrollern

Die Cortex-M3-Mikrocontrollerfamilie ist mit der neuen Debug- und Trace-Einheit „ARM CoreSight“ ausgestattet. Gegenüber bisherigen On-Chip-Debug-Lösungen weist CoreSight eine ganze Reihe neuer Funktionen auf, die fast an einen traditionellen In-Circuit-Emulator heranreichen. Trotzdem geht CoreSight ökonomisch mit den knappen externen Anschlüssen des Chips um.

Von Reinhard Keil

Der Cortex-M3-Prozessor von ARM ist mit der Debug- und Trace-Technik „ARM Core-Sight“ ausgestattet, die die Funktionen des klassischen On-Chip-Debugging-Blocks „Embedded ICE“ deutlich vergrößert. Zusätzlich zum normalen Debugging mit Programmablaufsteuerung wartet die CoreSight-Technik mit On-the-Fly-Speicherzugriff, Daten-Tracing, Event-Tracing und Instrumentation-Tracing über einen herkömmlichen, preisgünstigen JTAG-Anschluss auf. Zusammen mit Entwicklungs-Tools nach neuestem Stand der Technik sorgen diese neuen Methoden für eine erhebliche Beschleunigung der Software-Verifikation und des Debuggings von Programmen.

Bevor es das On-Chip-Debugging gab, setzten die meisten Software-Entwickler teure In-Circuit-Emulatoren (ICE) zum Testen von Applikationen auf Mikrocontrollern ein. Anspruchsvolle Emulatoren wurden über komplexe Adapter angeschlossen und boten außerdem weitreichende Befehls- und Daten-Tracing-Fähigkeiten mit komplexen Triggerfunktionen.

Diese Emulatoren basierten auf speziellen Bond-out-Bausteinen, die von den Bauelementen aus der Serienproduktion abwichen und im Vergleich zu normalen Mikrocontrollern sehr teuer waren. Die sehr hohen Taktfrequenzen moderner Mikrocontroller und die miniaturisierten Gehäuse mit Unmengen von Anschlüssen machen die Nutzung der traditionellen ICE-Technologie mittlerweile unmöglich.

Moderne Mikrocontroller sind mit On-Chip-Debugging-Logik ausgestattet, die den Speicher, die CPU-Register und die Programmverarbeitung einer Beobachtung zugänglich macht. Diese Logik ist in allen Bausteinen aus der Serienfertigung enthalten und deshalb in ihrem Umfang begrenzt, um keine Mehrkosten entstehen zu lassen. Da der Zugriff jedoch meist über ein standardmäßiges JTAG-Interface erfolgt, fehlt es zudem an der für das Instruction-Tracing erforderlichen zusätzlichen Bandbreite, und so beschränkt sich der Großteil der On-Chip-Debug-Implementierungen auf die einfache Ablaufsteuerung des Programms mit einem begrenzten Umfang an Breakpoint-Features. Mikrocontroller auf Basis des Cortex-M3-Prozessors sind dagegen mit ARMs Core-Sight-Debug-Technologie ausgestattet, die – über einen konventionellen JTAG-Anschluss und ohne teure Hardware – nützliche Trace-Informationen zur Verfügung stellt (Tabelle).

Feature	Embedded ICE (ARM7, ARM9)	CoreSight Cortex-M3
Debug-Interface	JTAG	JTAG oder Serial Wire
Hardware-Breakpoint	2 für Programm-Ausführung oder 1 für Speicherzugriffe	8 für Programm-Ausführung und 4 für Speicherzugriffe (siehe Hinweis)
Software-Breakpoint	begrenzt	unbegrenzt
Speicherzugriff während der Programm-Ausführung	ja, mittels des Real-Time Agent in der Anwender-Applikation	ja, von der Hardware unterstützt
Befehls-Tracing	per ETM	per SWD
Daten-Tracing	per ETM	ja
Event-Tracing	nein	ja
Instrumentation-Tracing	nein	ja

Tabelle: Die Debug-Features im Vergleich. Hinweis: Da die Anzahl der Break-Register bei CoreSight variabel ist, sind in einigen Mikrocontrollern u.U. weniger Hardware-Breakpoints verfügbar.

Die On-Chip-Debug-Technologie von ARM

Mit der Einführung des Prozessors ARM7TDMI stellte ARM die On-Chip-Debug-Lösung „Embedded ICE“ vor. Es handelt sich hierbei um einen kostengünstigen Hardware-Block, der umfassende Run-Control-Features mit zwei hardwaremäßigen Break-Registern zur Verfügung stellt, die entweder durch die Programmausführung oder durch Speicherzugriffe getriggert werden können. Ein zusätzlicher „Debug Communication Channel“ (DCC) ermöglicht während der Programmausführung den Datenaustausch mit der Applikation des Anwenders. Der Embedded ICE ist die standardmäßige Debug-Einheit in allen ARM7TDMI oder ARM9-Mikrocontrollern, die es derzeit von zahlreichen Halbleiteranbietern gibt. Von der Tool-Industrie wird der Embedded ICE in großem Umfang mit standardisierten, kostengünstigen JTAG-Adaptoren unterstützt, die den Verzicht auf teure Hardware-Adaptionen ermöglichen.

Da der Embedded-ICE-Block als On-Chip-Debug-Hardware allerdings keine Daten oder Befehle aufzeichnen kann, enthalten einige der ARM-Mikrocontroller auch die „Embedded Trace Macrocell“ (ETM). Die mit dem Befehls-Tracing einhergehenden hohen Datenraten sind jedoch auf Datenausgangsleitungen zusätzlich zu den standardmäßigen JTAG-Pins angewiesen. Ein spezieller ETM-Emulator dient dem Anschluss an diese ETM-Datenausgänge und interpretiert die Trace-Informationen.

In Mikrocontrollern belegt die Embedded Trace Macrocell I/O-Leitungen, die eigentlich von der Anwender-Applikation benötigt werden, sodass die ETM von den Ingenieuren häufig nicht genutzt werden kann. Mit dem Ziel, die Zahl der für das Debugging erforderlichen I/O-Pins zu minimieren, stellt die neue CoreSight-Lösung an einem herkömmlichen JTAG-Anschluss zusätzliche Betriebsarten zur Verfügung:

- Einen Standard-JTAG-Modus für den Anschluss an eine JTAG-Chain oder für JTAG-Adapter bisheriger Bauart. Diese Betriebsart benötigt fünf Pins.
- Einen Serial-Wire-Modus, der für das Run-Control-Debugging mit nur zwei I/O-Pins auskommt. Der Serial-Wire-Modus stellt eine abweichende Betriebsart des JTAG-Ports dar, deren Kommunikation ausschließlich über die Pins TCLK und TDI abläuft.
- In der Serial-Wire-Betriebsart kann ein zusätzlicher SWV-Ausgang (Serial Wire Viewer) auf der TDO-Leitung zur Ausgabe von Daten-Trace-, Befehls-Trace- und Instrumentation-Trace-

Informationen dienen.

CoreSight ist die in Mikrocontrollern auf Basis des Cortex-M3-Prozessors eingesetzte Debug-Technologie. Für den Anschluss an die Debug-Einheit wird lediglich ein preisgünstiger JTAG-Adapter (z.B. Keil ULINK2) benötigt. Abgesehen von den Trace-Features, implementiert die CoreSight-Einheit zusätzliche Break-Register und erlaubt Speicher-Zugriff auch während laufender Programmausführung, ohne dass zusätzlicher Software-Aufwand entsteht.

Live-Zugriff auf den Speicher

Mikrocontroller auf der Basis von ARM-Prozessoren werden von der Entwicklungs-Tool-Industrie in großem Umfang unterstützt. Zum Beispiel bietet das ARM RealView Microcontroller Development Kit (MDK) von ARM/Keil bausteinspezifischen Support für mehr als 260 Standard-Mikrocontroller. Das Kit kombiniert den ARM-RealView-Compiler mit dem µVision Debugger/IDE und dem RTX-Echtzeit-Kernel (Bild 1).

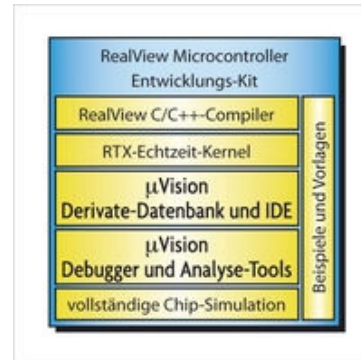


Bild 1. Die Komponenten des RealView Microcontroller Development Kit.

Der µVision Debugger wird über den USB-JTAG-Adapter ULINK2 an Mikrocontroller auf Basis des Cortex-M3-Prozessors angeschlossen (Bild 2). Der ULINK2 ermöglicht die Flash-Programmierung sowie das Hardware-Debugging und unterstützt sämtliche CoreSight-Betriebsarten. Mit der Kombination aus MDK und ULINK2 erhält der Anwender eine komplette Software-Entwicklungsumgebung für Projekte auf Cortex-M3-Basis.

Der µVision Debugger kann Speicherinhalte und Variablen in mehreren Daten-Formaten darstellen. Da Speicherinhalte und Variablen auch während der Programmausführung laufend aktualisiert werden, hat der Anwender stets den Überblick über den jeweiligen Programmstatus. Zusätzlich können Breakpoints gesetzt werden, deren Triggerung beim Zugriff auf bestimmte Variable mit oder ohne Wert erfolgt.

Ein Beispiel hierzu:

```
BS write my_value
/* stop on write to my_value */
```

Der ULINK2 kann so programmiert werden, dass er den Ausgangspin des Serial Wire Viewer nutzt. In dieser Betriebsart kann der Anwender Trace-Informationen über folgende Abläufe erhalten:

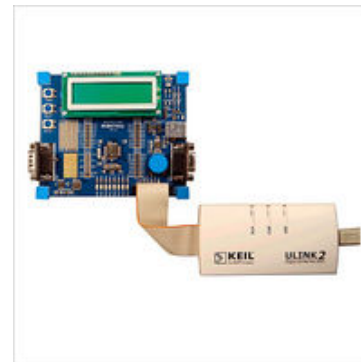


Bild 2. JTAG-Adapter ULINK2, angeschlossen an die Ziel-Hardware.

- Daten-Lese- und Schreibzugriffe auf ausgewählte Variablen (optional mit Zeitstempel und Programmzähler-Werten) können in einem Logikanalysator-Fenster überprüft werden.
- Ereigniszähler geben Auskunft über die CPU-Zyklusstatistik, aus der wiederum die erforderlichen Wait-States oder Leerlaufzeiten des Bausteins abgelesen werden können.
- Exception- und Interrupt-Verarbeitung mit Timing-Statistiken, die beim Optimieren der Interrupt-Funktionen helfen.
- Periodisches Abfragen des Programmzählers zum Aufspüren des Punkts, an dem ein Programm in eine Endlosschleife gerät.
- User-Trace-Daten, die über 32 ITM-Register (Instrumentation Trace Macrocell) ausgegeben werden können und für die Timing-Analyse oder einfaches Debugging nach der printf-Methode dienen können.

Der nicht-invasive Serial-Wire-Viewer-Modus kommt ohne Monitor-Software oder zusätzliche Wait-Zyklen der CPU aus. Um den Bandbreitenbedarf der Trace-Informationen zu reduzieren, kann die Datenerfassung selektiv aktiviert werden. Überprüfen lassen sich die erfassten Trace-Daten im µVision-Trace-Records-Fenster (Bild 3), das außerdem zusätzliche Filter für die Datenausgabe bereithält.

Bild 4 zeigt die typische Ausgabe des integrierten µVision-Logikanalysators, der Auskunft über Werteänderungen von bis zu vier ausgewählten Variablen über die Zeit gibt. Wenn Programmzählerstände in den Trace-Informationen enthalten sind, lässt sich durch Anklicken des Buttons „Code Show“ der Quellcode darstellen, der die Variablenmodifikation anstößt.

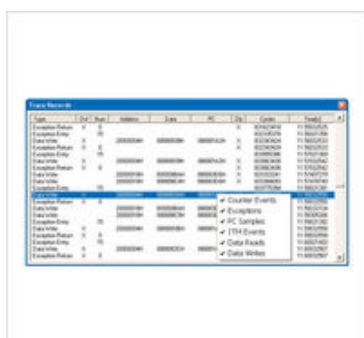


Bild 3. Trace-Records-Fenster zur selektiven Prüfung sämtlicher Trace-Informationen des Cortex-M3. Logikanalysator-Darstellung des Variablen-Tracings.

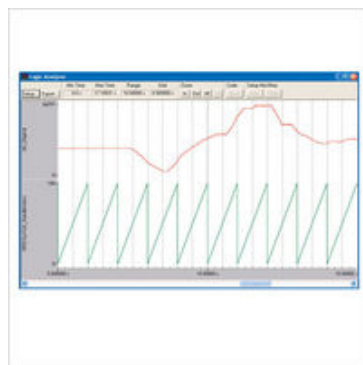


Bild 4. Das Tracing zeichnet z.B. auf, wie sich Variablen ändern. In einer Logikanalysator-Darstellung kann der Variableninhalt visualisiert werden.

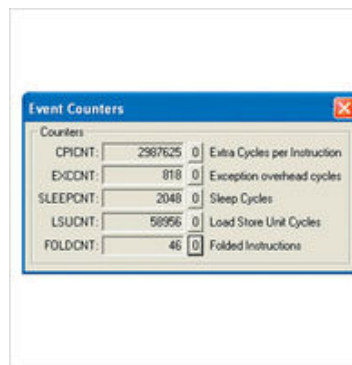


Bild 5. Der Event-Counters-Dialog gibt Auskunft über Timing-Statistiken.

Trace-Ereignisse und Instrumentierungs-Trace-Register

Der Cortex-M3-Prozessor liefert Verarbeitungs-Statistiken, die Hilfestellung beim Ermitteln der Leistungsfähigkeit der jeweiligen Hard- und Software-Implementierung leisten. Der in Bild 5 gezeigte Event-Counters-Dialog gibt neben der Information über die Gesamt-Verarbeitungszeit auch die folgenden Detailangaben aus:

- Zusätzliche, in Wait-States verbrachte Zyklen (beim Warten auf langsame Speicher),
- Zusatzaufwand durch CPU-Exceptions,
- Sleep- oder Idle-Phasen des Bausteins,
- Load/Store-Einheiten und gefaltete Instruktionen zur Beschleunigung der Verarbeitung.

Die ITM-Einheit (Instrumentation Trace Macrocell) implementiert 32 Stimulus-Register, mit deren Hilfe die Ausgabe zusätzlicher Trace-Daten über den Serial Wire Viewer möglich ist. Der hierfür entstehende Mehraufwand in der Anwender-Applikation ist minimal, da lediglich ein Schreibzugriff auf ein ITM-Register erforderlich ist. Die ITM-Trace-Ausgabe kann optional mit Timing-Informationen versehen werden, sodass diese Code-Instrumentierung auch zum Analysieren von Verarbeitungszeiten herangezogen werden kann.

Die ITM-Einheit lässt sich für unterschiedliche Informationen nutzen. Im ITM-Viewer-Fenster ist die Übertragung von ASCII-Textstrings über die ITM-Register möglich. Bild 6 zeigt das Beispiel eines Debuggings nach dem printf-Prinzip, das lediglich die Implementierung der folgenden Routine für die serielle Ausgabe erfordert:

```
int SendChar (int ch)
{
    /* Write serial output to ITM */
    while (ITM_Port32(0) == 0);
    ITM_Port8(0) = ch;
    return (ch);
}
```

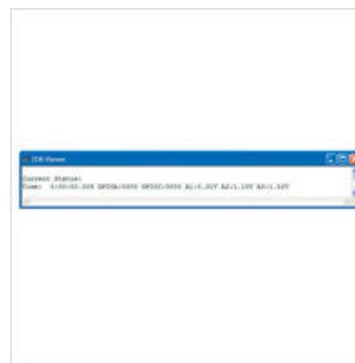


Bild 6. printf-Ausgabe mithilfe der ITM-Register.

Auf den Spuren der Interrupts

Durch Freigabe des Exception Tracing wird veranlasst, dass der Serial-Wire-Viewer-Ausgang Informationen über die Verarbeitung von Interrupt-Routinen in der Applikation ausgibt. Auch Angaben über die Zahl der Aufrufe, minimale und maximale Verarbeitungszeiten sowie minimale und maximale Zeitabstände zwischen Interrupt-Aufrufen lassen sich – mit Zeitstempeln versehen – ausgeben.

Die Trace-Informationen, die über den Serial Wire Viewer ausgegeben werden, lassen sich mittels eines Konfigurations-Dialogs selektiv freigeben. Die selektive Trace-Ausgabe stellt einerseits gezielt die zum Analysieren des anstehenden Problems benötigten Informationen zur Verfügung und reduziert andererseits den Bandbreitenbedarf. Indem sie mit nur drei I/O-Pins eine umfassende Debug- und Trace-Verbindung zur Verfügung stellt, erfüllt die CoreSight-Technologie somit die Anforderungen bei der Entwicklung von Mikrocontroller-Software. *jk*

Autor:



Reinhard Keil

studierte Elektrotechnik an der Fachhochschule München. Anschließend war er als Software-Entwickler bei Siemens tätig. 1985 gründete er zusammen mit seinem Bruder die Keil Elektronik GmbH. Er ist Co-Entwickler des Keil-C51-Compilers, der die Grundlage für den weltweiten Erfolg von Keil Software ist. Nach der Übernahme von Keil Software durch ARM ist er jetzt Direktor für Microcontroller Development Tools bei ARM.

Reinhard.Keil@arm.com

© 2008 WEKA FACHMEDIEN GmbH
Alle Rechte vorbehalten

Verwandte Webseiten:
www.magnus.de * www.franzis.de * www.funkschau.de