ELEKTRONIK PRAXIS

Interview – Mehr Präzision für Echtzeitbetriebssysteme

„Cycle-Resolution Timing bringt bisher ungeahnte Vorteile in Embedded-Anwendungen“

15.12.2021 | Von Sebastian Gerstl

Mit einem in eigenen Worten revolutionären Ansatz will der Embedded-Software-Experte SEGGER den RTOS-Markt aufmischen. embOS-Ultra verspricht höhere Auflösung, größere Präzision und Energiesparen out-of-the-box – ein Echtzeitbetriebssystem mit Nachhaltigkeitsversprechen. Wie das alles funktioniert, erklärt Lead Software Architect Martin Cröll im Exklusivinterview.

Martin Cröll in seinem Büro in Monheim am Rhein. Als Software Lead Architect war er für die Entwicklung des ersten RTOS mit Cycle-Resolution-Timing verantwortlich.
(Bild: Segger)

ELEKTRONIKPRAXIS: Herr Cröll, das Prinzip des RTOS (Real Time Operating System, Echtzeitbetriebssystem) ist inzwischen fast 40 Jahre alt. Nun hat Segger 2021 embOS-Ultra vorgestellt. Welche Verbesserungen soll das in das etablierte RTOS-Umfeld bringen?

Martin Cröll: Eines unserer Unternehmensziele ist es, klimaneutral zu sein und zu bleiben. Dies bedeutet auch, unsere eigenen Produkte – zu nennen sind hier unsere marktführenden Debug-Probes J-Link, J-Trace und unsere Flasher – noch energieeffizienter zu machen, obwohl sie schon heute insgesamt weniger Energie verbrauchen, als mancher Lüfter in Konkurrenzprodukten alleine. Um dieses Ziel zu erreichen, haben wir u.a. auch beim RTOS angesetzt und mit embOSUltra die gewünschten Ergebnisse erzielt, noch mehr Energie einzusparen.

ELEKTRONIKPRAXIS: Sie setzen also Ihr RTOS in Ihren Debuggern und Flashern ein?
Martin Cröll: Richtig, wir verkaufen seit vielen Jahren unser traditionelles RTOS embOS nicht nur an Kunden, sondern setzen es wie auch andere Komponenten unseres All-in-One Embedded-OS emPower OS in unseren J-Links und Flashern ein. Damit sind wir erster Nutznießer jeder Verbesserung im emPower OS und liefern unseren Kunden Produkte, die sich bereits in der Praxis bei uns selbst bewährt haben.

ELEKTRONIKPRAXIS: Wie arbeitet embOS, und worin besteht der Unterschied zum bisherigen embOS-Ultra?
Martin Cröll: embOS arbeitet wie alle bisher bekannten RTOSe seit den 80er Jahren mit einem Systemtick. Dies ist die grundlegende Zeiteinheit. Jede Zeitangabe wird in Vielfachen von Ticks angegeben, wobei Benutzer den Abstand zwischen diesen Ticks individuell konfigurieren können, üblicherweise zum Beispiel auf 1 Millisekunde. Das bedeutet dann, dass ein Hardware Timer so programmiert wird, dass er 1000 Mal pro Sekunde einen Interrupt erzeugt und damit jeweils das Ablaufen einer Millisekunde anzeigt. Obwohl 1 Millisekunde bereits hochauflösend und schnell klingt, sahen wir angesichts der Anforderungen moderner Embedded-Systeme aber trotzdem die Notwendigkeit für Verbesserungen.

ELEKTRONIKPRAXIS: Inwiefern?

Martin Cröll: Unser neues, Taktzyklus-basiertes Scheduling verändert die grundlegende Zeiteinheit des Systems, wodurch die Auflösung des Schedulings drastisch erhöht wird. Anstatt sich auf den traditionellen Systemtick zu verlassen, verwendet embOS-Ultra intern für alle Operationen Taktzyklen. Zeitbasierte Operationen etwa für Task-Verzögerungen oder Software-Timer, die bisher nur in Vielfachen von Ticks angegeben werden konnten, können dadurch nun in Taktzyklen angegeben werden. Zusätzlich können Nutzer von

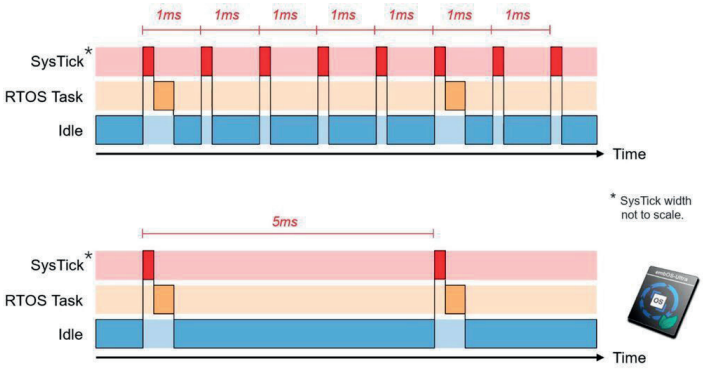

Bild 1: Diese Grafik zeigt den Unterschied zwischen embOS-Ultra (unten) und dem herkömmlichen RTOS embOS (oben): Mit embOS-Ultra bleibt die CPU deutlich länger im Energiespar-Modus und wird weniger oft durch Interrupts (rot) aufgeweckt. Die Folge ist mehr Rechenleistung für die Anwendung und weniger Energieverbrauch.
(Bild: Segger)

embOS-Ultra für zeitbasierte Operationen statt lediglich SystemTicks nun auch systemunabhängige, hochauflösende Einheiten wie Mikrosekunden oder sogar Nanosekunden in ein und dergleichen Applikation verwenden .

ELEKTRONIKPRAXIS: Diese Option bereichert den bestehenden Markt für Echtzeitbetriebssysteme?
Martin Cröll: Absolut. Eine feinere zeitliche Granularität bedeutet, dass die zeitlichen Anforderungen einer Anwendung mit maximaler Genauigkeit erfüllt werden können. Dadurch kann das RTOS-Scheduling in vielen Situationen eingesetzt werden, in denen es bisher wegen der groben Zeitauflösung nicht verwendet werden konnte.

Es gibt zum Beispiel oftmals Situationen, in denen Verzögerungen zwischen 5 und 100 Mikrosekunden erforderlich sind, die mit herkömmlichen tick-basierten Systemen aber nicht oder nur mit erheblichen Nachteilen erreicht werden können. In diesen Fällen mussten die Kunden bislang oftmals nicht unerhebliche Entwicklungs-Mehraufwendungen auf sich nehmen, weil ein RTOS diese Anforderungen nicht erfüllen konnte.

ELEKTRONIKPRAXIS: Was bringt embOS-Ultra aber für Anwendungen, die heute schon ein traditionelles RTOS einsetzen?

Martin Cröll: Da gibt es zwei Punkte zu nennen: Die Präzision und den Energieverbrauch. Zum ersten werden zeitliche Abläufe genauer. In den meisten RTOSen wird z.B. ein Delay(10) den Betrieb des entsprechenden Threads für eine Zeit zwischen 9 und 10 Millisekunden unterbrechen - je nachdem, wie weit der nächste Tick des Systems entfernt ist.

Warum ist das so? Eine programmierte Verzögerung kann nicht zwischen zwei System-Tick-Interrupts enden, sondern erst mit dem nächsten System-Tick-Interrupt, der dann

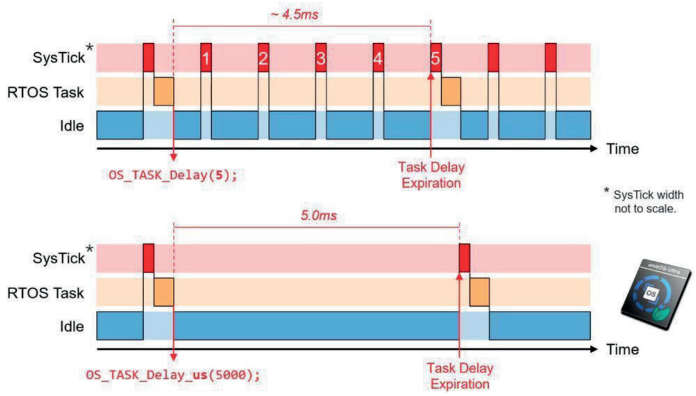

Bild 2: Ein Beispiel für die höhere Präzision von embOS-Ultra. Ein Delay von 5 ms kann bei einem herkömmlichen RTOS in Wirklichkeit nur 4,5 ms dauern, da eine programmierte Verzögerung nicht zwischen zwei System-Tick-Interrupts enden kann, sondern erst mit dem nächsten System-Tick-Interrupt, der dann den Scheduler auslöst (oben). Bei embOS-Ultra wird die programmierte Verzögerung hingegen exakt eingehalten (unten).
(Bild: Segger)

den Scheduler auslöst. Daher können Aufgaben, die für einen Zeitraum unterbrochen werden sollen, der kürzer als ein Systemtick ist, dies nur erreichen, indem sie aktiv mittels Polling des Hardware Timers warten, bis der gewünschte Zeitraum verstrichen ist. In embOS-Ultra mit zyklusbasierter Auflösung führt ein Delay_ms(10) hingegen zu genau 10 ms Unterbrechungszeit.

Und dann ist da natürlich der Energiespareffekt. Überlegen Sie mal, welchen Aufwand Halbleiterhersteller Jahr für Jahr betreiben, um noch energiesparendere ULP-Mikrocontroller zu entwickeln. Mit embOS-Ultra bekommen Sie Energieeinsparungen quasi out-of-the-Box.

ELEKTRONIKPRAXIS: Wie genau funktioniert das?
Martin Cröll: Selbst wenn es nur einen Thread gibt, der für mehrere aufeinander folgende Systemticks ausgeführt wird, wird die System-Tick-Unterbrechung immer noch periodisch auftreten und dadurch Rechenzeit „verschwendet“. Dazu müssen zuvor auch noch der Status der CPU, also Registerinhalte und Flags, auf dem Stack gerettet und anschließend wiederhergestellt werden, was ebenfalls Rechenzeit verbraucht. Genauso muss zum Beispiel eine CPU, die sich im Energiespar-Modus befindet, wenn keine Threads ausgeführt werden, jedes Mal wieder in den aktiven Modus überführt werden um die Interrupt-Service-Routine auszuführen- Auch das kostet Rechenzeit und somit Energie, die mit embOS-Ultra eingespart werden kann.

ELEKTRONIKPRAXIS: Hat das nicht Einfluss auf das Taktzyklus-basierte Scheduling?
Martin Cröll: (lacht) Ich hatte gehofft, dass Sie das nicht fragen würden. Aber im Ernst: Weil das System verschiedenste Zeiteinheiten verarbeiten (d.h. in Systemzyklen umrechnen) können muss, statt lediglich in Ticks zu rechnen, ist die Codegröße naturgemäß leicht

erhöht - das sind aber nur wenige hundert Bytes, was für die meisten Systeme unbedeutend ist. Zusätzlich müssen wir natürlich bei der Anzahl von Taktzyklen mit 64-bit-Werten rechnen, statt mit 32-bit-Werten, die für das Mitzählen von System-Ticks noch ausreichen. Allerdings ist der daraus resultierende Performance-Verlust minimal und auf modernen 32-Bit-CPUs in der Praxis nicht signifikant.

ELEKTRONIKPRAXIS: Läuft man dadurch nicht trotzdem Gefahr, dass die 64-bit-Werte mit der Anzahl der Taktzyklen seit Systemstart überlaufen? Stürzt dann das System ab?
Martin Cröll: Nein. Lassen Sie uns mal rechnen und am Anfang von einem schnellen System mit einer Taktfrequenz von 1 GHz ausgehen. Hier tritt ein Überlauf nach 2 ^ 64 Taktzyklen auf, was ungefähr 585 Jahren entspricht. Dies ist in 99,999% der Fälle wohl kein Problem.

Aber selbst wenn dies der Fall wäre und wir annehmen, das System müsste 1.000 Jahre lang laufen, gibt es Lösungen, z.B. die Verwendung eines langsameren Taktes. Wenn Sie statt 1 GHz nur einen 100 MHz-Takt annehmen, erreichen Sie bereits 5.850 Jahre und haben immer noch eine zeitliche Auflösung von 10 Nanosekunden.

ELEKTRONIKPRAXIS: Sie haben zuvor das Mitzählen von SystemTicks erwähnt. Viele Entwickler setzen in ihren Anwendungen auf diesen Wert, weil sie wissen wollen, wie viele Interrupts seit dem Systemstart aufgetreten sind. Wenn embOS-Ultra keinen „Tick Count“ hat, bedeutet das doch, dass diese Anwendungen nicht mehr funktionieren?
Martin Cröll: Der einfache „Tick Count“, der die Anzahl der Timer-Ticks seit dem Start des Systems angibt, ist in embOS-Ultra tatsächlich verschwunden. Beim taktzyklusbasierten Scheduling wird der Timer-Interrupt zwar immer noch verwendet, aber er ist nicht periodisch. Stattdessen handelt es sich um einen Single-Shot-Timer, der so programmiert wird, dass er genau dann einen Timer-Interrupt erzeugt, wenn er benötigt wird.

Aber wie Sie sagen, gibt es Fälle, wo es nützlich war, die Anzahl der SystemTicks abzufragen , sei es, um sie in einer Webschnittstelle anzuzeigen oder in kurzen Schleifen mit Timeouts zu verwenden. Wenn Sie einen solchen periodischen Interrupt jede Millisekunde benötigen, gibt es mit embOS-Ultra eine Möglichkeit, den traditionelle Tick Count zu ersetzen.

Um den Tick Count zu replizieren, können Sie nämlich die zyklusbasierte Zeit abfragen und sie durch die Taktfrequenz teilen, z.B. bei einem 400-MHz-System OS_TIME_GetCycles() / 400000 berechnen. Um das Ganze noch mehr zu vereinfachen, gibt es hierfür sogar eine

API-Funktion, die diesen Wert liefert und die praktischerweise OS_TIME_GetTime_ms() heißt.

ELEKTRONIKPRAXIS: Mit welchem Timer-Konzept arbeitet embOS-Ultra? Das alles, was Sie beschreiben, klingt ja relativ komplex.
Martin Cröll: Die meisten einfachen RTOSe verwenden einfach einen Hardware-Timer, der so konfiguriert ist, dass er periodische Interrupts erzeugt, typischerweise eben einmal pro Millisekunde.

embOS-Ultra verwendet im Grunde zwei Hardware-Timer. Ein Timer wird für die Langzeitstabilität verwendet. Dieser Timer läuft im kontinuierlichen Modus und erzeugt keine Interrupts.
Der andere Timer arbeitet im Single-Shot-Modus, d.h. er zählt von einem konfigurierten Wert auf 0 herunter oder von 0 bis zu einer konfigurierten Zählergrenze und erzeugt dann einen einzelnen Interrupt. Wird ein Timer verwendet, der zunächst von 0 bis zu einer konfigurierten Grenze zählt, und dann von dort aus weiter bis zum nächsten konfigurierbaren Grenzwert, so genügt sogar ein Hardware-Timer für den Betrieb von embOS-Ultra.

Die meisten Embedded-Systeme haben jedoch mehr als genug Hardware-Timer zur Verfügung, so dass auch die Verwendung zweier Timer üblicherweise kein Problem darstellt.

ELEKTRONIKPRAXIS: Die Arm-Architektur ist ja heute der quasi-Standard in der Embedded-Welt und wird selbstredend unterstützt. Ist embOS-Ultra auch für andere Architekturen einsetzbar?
Martin Cröll: embOS-Ultra ist für viele CPU- und Compiler-Kombinationen, darunter natürlich ARM Cortex-A/R/M oder auch RISC-V, verfügbar. Eine embOS-Ultra-Portierung enthält zudem eine Vielzahl Board-Support-Packages für verschiedene Devices und Evaluierungsboards. So können Nutzer embOS-Ultra ohne zusätzlichen Aufwand direkt in Betrieb nehmen.

ELEKTRONIKPRAXIS: Angenommen, jemand setzt bereits ein herkömmliches RTOS ein und möchte nun zu embOS-Ultra migrieren – wie groß ist der Aufwand?
Martin Cröll: In embOS-Ultra haben wir die bestehende API im Vergleich zu embOS unverändert gelassen. Bestehende Funktionen verhalten sich im neuen taktzyklus-

basierten embOS-Ultra daher genauso wie im traditionellen embOS. Das bedeutet, dass API-Funktionen wie OS_TASK_Delay() immer noch das gleiche, auf Millisekunden ausgerichtete Timing ergeben, wodurch sichergestellt wird, dass sich das Timing einer Anwendung, die von embOS auf embOS-Ultra migriert wird, nicht ändert.

Um jedoch die Vorteile der neuen Funktionalität zu nutzen, wurde die API um Funktionen wie OS_TASK_Delay_ms(), OS_TASK_Delay_us(), OS_TASK_Delay_Cycles() erweitert, die eine viel genauere Zeitmessung ermöglichen. Das Gleiche wurde auch für die vom RTOS bereitgestellten Software-Timer getan.

Das Ergebnis ist das Beste aus beiden Welten: ein genaueres Timing für geänderte und / oder erweiterte Anwendungen, bei gleichzeitig 100%iger Kompatiblität für Anwendungen, die nicht modifiziert werden sollen.

Herr Cröll, vielen Dank für das Gespräch.