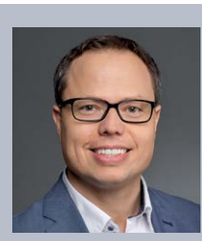
trainiert, mit aggregierten Daten aus mehreren Anlagen, mehreren Standorten. Dieser Trainingsschritt braucht Rechenleistung, historische Daten und Iterationszyklen. Anschließend wird das fertige Modell an die Edge ausgeliefert, dort ausgeführt, und nur die verdichteten Ergebnisse fließen zurück in die Cloud. Diese lassen sich dann für Monitoring oder Modell-Updates nutzen. Die Rollen sind so verteilt, dass die Cloud trainiert und koordiniert, während die Edge entscheidet und reagiert. Wer versucht, beides in der Cloud zu machen, erkauft sich Flexibilität auf Kosten von Latenz und Datensouveränität. Heutige IIoT-Plattformen ermöglichen es den Maschinenherstellern, KI-Modelle direkt in der Maschine zu installieren und zu verwalten. Das bedeutet für Mittelständler, dass die KI bereits eingebaut ist und dass sie keine zusätzliche Infrastruktur benötigen. Der Maschinenbauer liefert die KI mit, und der Fabrikbetreiber profitiert davon vom ersten Tag an.

Was das für Einkäufer und Entscheider bedeutet

Wer heutzutage Maschinen beschafft oder Produktionslinien modernisiert, sollte eine Frage stellen, die vor drei Jahren noch niemand gestellt hat: Welche KI-Fähigkeiten sind in der Maschine bereits enthalten?

Dies ist eine Einkaufsentscheidung, ähnlich wie die Frage nach Sicherheitszertifizierungen oder Wartungsverträgen. Ein Modul ohne integrierte NPU bedeutet, dass jede KI-Funktion auf externe Dienste angewiesen bleibt, mit allem, was dies für Latenz, Kosten und Abhängigkeiten bedeutet. Ein Modul mit integriertem Accelerator bedeutet: Entscheidungen fallen lokal, in Millisekunden, ohne Cloud-Anbindung.

Drei Sekunden Latenz waren für das Leiterplatten-Unternehmen kein technisches Problem. Sie waren das Ergebnis einer Kaufentscheidung, die niemand bewusst getroffen hatte. Das ist der eigentliche Punkt. Edge AI ist keine Zukunftstechnologie, auf die man wartet. Sie ist eine Anforderung, die man heutzutage in die Ausschreibung schreibt – oder eben nicht. (ak)



Dr. Jürgen Krämer, Cumulocity

„Ein Modul ohne integrierte NPU bedeutet, dass jede KI-Funktion auf externe Dienste angewiesen bleibt."

RTOS-Architekturen in heterogenen Prozessoren wie den i.MX-CPUs

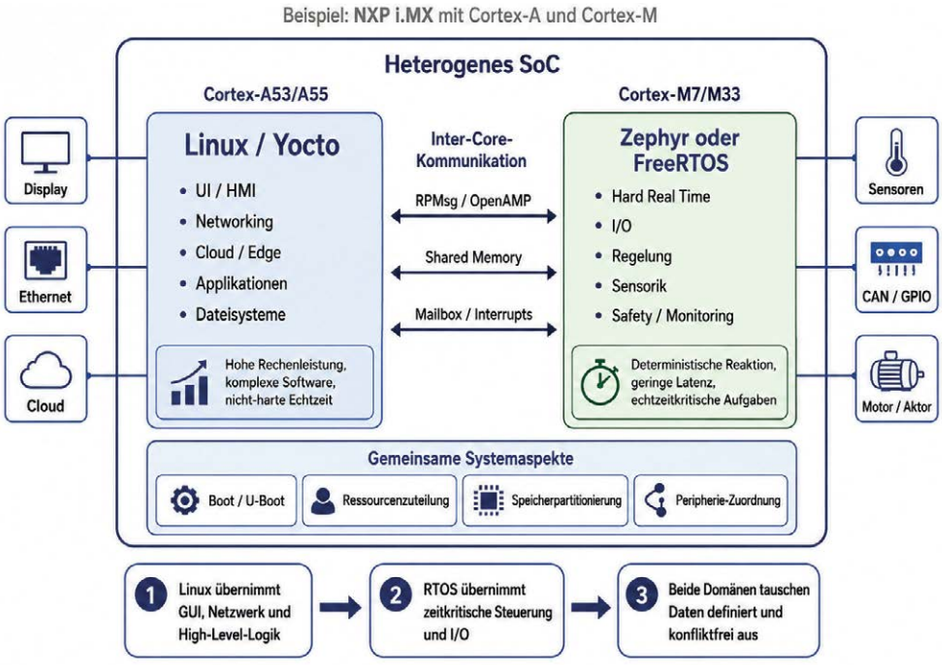
Zephyr OS oder FreeRTOS?

Für heterogene Prozessoren wie die Arm-basierten i.MX-CPUs von NXP Semiconductors bieten sich in Embedded-Anwendungen die Echtzeitbetriebssysteme FreeRTOS und Zephyr OS an. Welches der beiden Real-Time Operating Systems (RTOS) eignet sich nun für welche Applikationen am ehesten?

Von Andreas Kopietz,
Key Account Manager
bei F&S Elektronik Systeme

Aktuelle i.MX-Prozessoren von NXP kombinieren Arm-Cortex-A-Kerne mit Arm-Cortex-M-Kernen für Echtzeit- und Low-Power-Aufgaben. Beim i.MX 8M Plus stehen Cortex-A53 und Cortex-M7 zur Verfügung, beim i.MX 93 bis zu zwei Cortex-A55 mit 1,7 GHz sowie ein Cortex-M33 mit bis zu 250 MHz. CAN FD und zwei Gigabit-Ethernet-Controller, davon einer mit TSN, zeigen die Ausrichtung auf Applikations-, Kommunikations- und Echtzeitaufgaben. Typischerweise übernimmt Linux in den Cortex-A-Kernen HMI, Netzwerk, Dateisystem und komplexe Applikationen. Ein Cortex-M kann davon entkoppelt beispielsweise Sensorik, Aktuatoransteuerung oder deterministische Regelkreise ausführen. Diese Aufteilung entspricht Asymmetric Multiprocessing (AMP), bei dem mehrere Betriebssysteminstanzen mit explizit zugeordneten Ressourcen parallel arbeiten.

Linux mit PREEMPT_RT, einem Patch-Set für Echtzeit und mittlerweile Teil des Kernels, kann geringe Latenzen erreichen, ersetzt aber nicht automatisch einen separaten Echtzeitkern. Bei AMP lassen sich CPU-Zeit, Interrupts, Speicher und Peripherie klar ei-



Asymmetric Multiprocessing (AMP): Zusammenspiel von Cortex-A- und Cortex-M-Kernen. (Bilder: F&S Elektronik Systeme)

nem Betriebssystem zuordnen. Echtzeit, übrigens laut DIN-Norm 44300 definiert als Betrieb eines Rechnersystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind, wird so weniger von der Linux-Systemlast abhängig.

FreeRTOS: schlank und direkt

FreeRTOS konzentriert sich auf den Kernel, Scheduler und klassische RTOS-Tasks wie Queues, Semaphoren, Mutexes und Software-Timer. Für kompakte Control-Firmware ist dieser Ansatz optimal: Die Applikation bleibt nah an den Treibern, und die Zahl der Abstraktionsschichten ist gering.

In einem i.MX 8M Plus kann Linux in den Cortex-A53-Kernen etwa HMI, OPC UA und Datenlogging ausführen, während der Cortex-M7 zyklische I/O-Verarbeitung und Regelalgorithmen übernimmt. Ist die Firmware auf wenige deterministische Tasks begrenzt, bleibt FreeRTOS mit dem SDK MCUXpresso eine gut beherrschbare und schlanke Lösung.

Die NXP-Treiber und Middleware zusammen mit den Tools von F&S lassen sich eng mit der Anwendung verbinden. Dies vereinfacht kleine Systeme, bedeutet aber zugleich, dass Hardwareabstraktion, Variantenmanagement und Build-Struktur stärker projekt- oder SDK-spezifisch bleiben.

Zephyr: Betriebssystemplattform statt Kernel

Das Zephyr-Project wurde 2016 von der Linux Foundation ins Leben gerufen. Gründungsmitglieder und aktive Supporter sind unter anderem Intel, Wind River und NXP.

Zephyr bietet ebenfalls einen prioritätsbasierten Echtzeit-Scheduler, ergänzt ihn jedoch um ein umfassendes Systemmodell. Der Devicetree beschreibt Hardware und Ressourcen, Kconfig steuert Features und Abhängigkeiten, CMake und West organisieren Build und Workspace. Alle Treiber werden über ein einheitliches Device Model eingebunden.

Mit Hardware Model V2 kann Zephyr heterogene SoCs mit mehreren Kernen und unterschiedlichen Architekturen direkt abbilden. Boards, SoCs, CPU-Cluster und einzelne Kerne werden dabei klar beschrieben. Sysbuild erleichtert den gemeinsamen Build mehrerer Software-Images. Das vereinfacht die Strukturierung komplexer AMP-Systeme.

Besonders relevant wird das, wenn eine Firmware mehrere Boardrevisionen, CAN FD, Ethernet, Diagnose oder unterschiedliche Peripherievarianten unterstützen muss.

Zephyr in Cortex-A

Technisch interessant und der größte Unterschied zu FreeRTOS ist Zephyr direkt in Cortex-A-Kernen. Aktuell unterstützt F&S den Cortex-A-Support unter anderem für i.MX 93, i.MX 8M Plus, i.MX 8M Mini, i.MX 8M Nano und i.MX 95. Für das i.MX-8M-Plus-EVK existieren Cortex-A53-Targets für einen einzelnen Core und als SMP-Variante; beim i.MX 95 werden Cortex-A55- und Cortex-A55-SMP-Varianten beschrieben.

Im i.MX 93 bindet Zephyr im Cortex-A55 unter anderem GICv3, CCM, IOMUX/Pinctrl, LPUART, RGPIO, LPI2C, Timer und Ethernet ein. Damit ist eine vollwertiges RTOS in einem leistungsfähigen Cortex-A-Kern möglich, ohne zwingend ein vollständiges Linux bereitstellen zu müssen.

Linux und Zephyr dynamisch auf Cortex-A-Cores verteilen

Grundsätzlich sind zwei Bootpfade möglich, die sich auf den jeweiligen Boards von F&S auch nutzen lassen: Zephyr kann bereits in U-Boot über cpu release beziehungsweise go gestartet oder später von Linux über remoteproc geladen und kontrolliert werden.

Bei remoteproc gehören zunächst alle Cortex-A-Cores zum SMP-Linux. Für den Start von Zephyr wird ein Core per CPU-Hotplug aus Linux entfernt, heruntergefahren und anschließend mit der RTOS-Firmware neu gestartet. Nach dem Stoppen kann derselbe Core wieder in das Linux-SMP zurückkehren. Der Core wird damit zur dynamisch zuweisbaren Rechenressource.

Dies setzt allerdings sauberes Resource Partitioning voraus. Der Linux-Devicetree definiert über fsl, cpus-bits, welcher Core einer Remoteproc-Instanz gehört; memory-region reserviert den RTOS-Speicher. Peripherie, die Zephyr exklusiv nutzt, muss natürlich auf der Linux-Seite deaktiviert werden.

Auch der Interrupt-Controller erfordert Koordination. Mehrere Betriebssysteme teilen sich den GICv3-Distributor und dürfen dessen globale Konfiguration nicht überschreiben. NXP sieht dafür CONFIG_GIC_SAFE_CONFIG=y in Zephyr und CONFIG_GIC_GENTLE_CONFIG=y im Linux-Kernel vor. Der Aufwand eines AMP-Systems liegt damit vor allem in der eindeutigen Zuteilung von Cores, RAM, Interrupts und Peripherie.

Kommunikation zwischen den Welten

Für Cortex-A/Linux und Cortex-M/RTOS wird in i.MX häufig RPMsg über Shared Memory eingesetzt. Auf Zephyr-Seite kommt typischerweise OpenAMP zum Einsatz, bei FreeRTOS in NXP-Projekten RPMsg-Lite. Shared Memory allein genügt jedoch nicht, wenn Cache-Kohärenz, Ownership und Synchronisation nicht eindeutig geregelt sind. Genau hier liegt die Herausforderung für Entwickler.

CRA-Readiness für RTOS

Für den Cyber Resilience Act (CRA) der EU kann eine stärker standardisierte Softwareplattform Vorteile bringen. Der CRA fordert unter anderem Security-by-Design, dokumentierte Software-Komponenten, wirksame Schwachstellenbehandlung und Sicherheitsupdates über den Supportzeitraum. Zephyrs zentral gepflegte Treiber und Subsysteme, reproduzierbare Build-Konfiguration über Kconfig/Devicetree sowie klar versionierte Module erleichtern dabei Nachverfolgbarkeit, Erfassung der Softwarekomponenten und das gezielte Einspielen von Fixes. Dies kann Aufwand für SBOM, Vulnerability Management und Update-Prozesse reduzieren. FreeRTOS kann grundsätzlich dieselben regulatorischen Ziele erfüllen, benötigt dafür aber häufiger projektspezifische Prozesse und zusätzliche Komponenten. Die Wahl von Zephyr erzeugt per se keine CRA-Konformität, kann deren technische Umsetzung und Pflege jedoch strukturell deutlich besser unterstützen.

Fazit

FreeRTOS ist eine gute Wahl, wenn ein Cortex-M wenige klar abgegrenzte Echtzeitaufgaben mit direktem Hardwarezugriff ausführt. Zephyr punktet, sobald Firmware, Hardwarevarianten und Multi-Core-Topologien selbst Teil der Plattformarchitektur werden.

Der NXP-Ansatz sowie die Implementierung seitens F&S zeigen, wie weit diese Entwicklung geht: Zephyr kann nicht nur in Cortex-M-, sondern auch in Cortex-A-Cores laufen; einzelne Cortex-A-Cores lassen sich dynamisch zwischen SMP-Linux und RTOS umverteilen. Damit entscheiden nicht mehr allein CPU-Variante oder der Scheduler über die Architektur des Systems, sondern Resource Partitioning, Interrupt- und Speichermodell, Peripherie-Ownership, Rechenleistung und Wartbarkeit des Gesamtsystems. F&S bietet zu diesem Thema diverse Workshops und Unterstützung für Embedded-Entwickler an. (ak)

Kriterium	FreeRTOS	Zephyr
Grundphilosophie	Schlanker RTOS-Kernel mit Fokus auf Einfachheit und geringem Overhead	Vollständige Embedded-OS-Plattform mit breitem Funktionsumfang
Einstieg	Einfach, kurze Einarbeitungszeit	Höhere Lernkurve durch mehr Konzepte und Werkzeuge
Scheduler	Fixed-Priority, präemptiv, optional Time-Silcing	Prioritätsbasiert, präemptiv und kooperativ, verschiedene Scheduler-Optionen
Hardwarebeschreibung	SDK- oder projektspezifisch, häufig Code-basiert	Devicetree zur Beschreibung der Hardware
Konfiguration	Über FreeRTOSConfig.h und Projektkonfiguration	Kconfig mit Abhängigkeiten und Build-Zeit-Optimierung
Build-System	Abhängig von SDK und Projekt	Standardisiert mit CMake und West
Treibermodell	Meist direkt an Hersteller-SDK gekoppelt	Einheitliches Device Model über alle unterstützten Plattformen
Portabilität	Stark projekt- und hardwareabhängig	Hohe Portabilität durch Abstraktion und standardisierte APIs
Geeignet für	Kleine, klar abgegrenzte Echtzeitaufgaben mit wenigen Tasks	Komplexe Firmware mit mehreren Protokollen, Varianten und Middleware
Ressourcenbedarf	Sehr geringer RAM- und Flash-Bedarf	Höher, abhängig von aktivierten Features
Kommunikation	Zusätzliche Libraries oder Middleware je nach Protokoll	Integrierte Netzwerkstacks und Protokoll-APIs
NXP i.MX Cortex-M	Sehr gut etabliert, enge Integration über MCUXpresso SDK	Gute Unterstützung, wachsendes Ökosystem
NXP i.MX Cortex-A	Möglich, aber plattformabhängig	Offiziell unterstützt (z. B. i.MX 933: Cortex-A55)
Bestehende Codebasis	Beste Walh bei vorhandenem FreeRTOS-Code	Migration möglich, aber Aufwand einplanen
Langfristige Skalierbarkeit	Gut für stabile, kleine Systeme	Sehr gut für wachsende und variantenreiche Produkte
	FreeRTOS: Ideal für schlanke, deterministische Echtzeitlösungen mit geringem Overhead.	**Zephyr:** Ideal für komplexe, vernetzte und langfristig skalierbare Embedded-Systeme.

Direkter Vergleich der beiden Echtzeit-Betriebssysteme FreeRTOS und Zephyr.