MICRO CONSULT

Mehr Gestaltungsspielraum durch Secure Exception Priority Boosting

Die Armv8-M-Architektur bringt grundlegende Sicherheit in Cortex-M-Geräte und ermöglicht so mehr Schutz für IoT-Systeme. Doch wie verhält sich das "Secure Exception Priority Boosting" Feature in der Praxis?

Sicherheitsprobleme im Internet der Dinge (IoT) gehen oft auf unzureichenden Schutz für Geräte am Rand eines Connected Systems zurück. Mangel an Rechenleistung und Speicher und natürlich Kostendruck werden oft als Gründe dafür angeführt, dass diese Geräte nicht angemessen geschützt werden können. Hacker wählen daher zunehmend intelligente Sensoren als Angriffspunkte in das gesamte Netzwerk.

Um anfällige Endpunkte besser zu schützen, wurde auch die Verwendung mehrerer MCUs untersucht, von denen eine oder mehrere ausschließlich Sicherheitsfunktionen wie Verschlüsselung und Authentifizierung übernehmen. Doch leider wirkt sich das auf die Komplexität und Kosten schnell negativ aus.

Der 2015 eingeführten 32-Bit Armv8-M-Architektur hatte Arm unter anderem TrustZone-Sicherheitserweiterungen für Cortex-M-Mikrocontroller (MCUs) hinzugefügt. Mit Sicherheitsfunktionen, die vergleichbar sind mit denen, die sich in vielen Cortex-A-Anwendungsprozessoren finden, bringt der Armv8-M grundlegende Sicherheit in Cortex-M-Geräte und gewährleistet mehr Sicherheit in IoT-Systemen von Anfang bis Ende.

Selbst in den umfangreichen Technical Reference Manuals der Arm-Dokumentation lässt das Thema "Secure Exception Priority Boosting" der neuen v8-M Architektur viel Interpretationsspielraum. Die Frage, wie sich dieses Feature denn in der Praxis wirklich verhält, bleibt offen. Grund genug, einmal genauer hinzuschauen und das Verhalten empirisch durch Experimente mit verschiedenen Controllern zu ermitteln.

Entwickeln in der Secure- und der Non-Secure-Welt

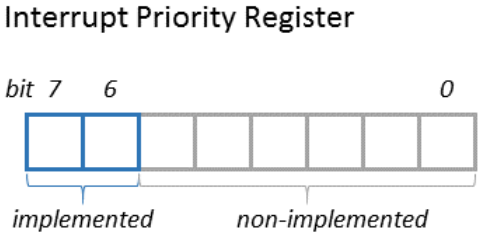
Durch die TrustZone gibt es nun innerhalb eines Controllers einen Secure- und einen Non-Secure-Bereich. Dabei führt der Mikrocontroller ein Programm immer in einer dieser beiden Welten aus und kann auch zwischen ihnen hin- und herspringen. Security Features sorgen dafür, dass ein Programm, das im Non-Secure-Bereich ausgeführt wird, nicht einfach auf den Secure-Bereich zugreifen kann, wo sich dadurch geschützt z.B. Passwörter und Verschlüsselungsprogramme befinden. Auch Interrupts und Exceptions werden in jeweils einer der Welten behandelt beziehungsweise deren Handler ausgeführt.

Die Software entsteht für beide Welten unabhängig voneinander. Als Softwareentwickler entwickelt man daher entweder in beiden Bereichen oder nur in einem.

Anmerkung: In den Arm-Architekturen unterscheidet man zwischen *Exceptions* und *Interrupts* als Bezeichnung für Unterbrechungen. In der Cortex-M-Architektur ist dieser Unterschied zwischen beiden Begriffen sehr gering. Für einen besseren Lesefluss benutze ich im Weiteren nur den Begriff Interrupts, auch wenn meistens beide Begriffe Sinn ergeben würden und der Vollständigkeit halber korrekt wären.

„PRIS"-Bit komprimiert Prioritäten der Interrupts im Non-Secure-Bereich

In den v8-M-Controllern mit der TrustZone als Security Extension werden Interrupts wahlweise im Secure- oder im Non-Secure-Bereich in Handlern abgearbeitet. Jeder dieser beiden Welten steht die komplette Prioritätslogik zur Behandlung der Interrupts zur Verfügung. Das heißt, es kann sowohl in der Secure- als auch in der Non-Secure-Welt High-Priority- oder Low-Priority-Interrupts geben. Das klingt zunächst schön, wirft aber die Frage auf, wie man als Software-Architekt das System so designen kann, dass die Behandlung von Low-Priority-Interrupts in der Secure-Welt nicht von einem High-Priority-Interrupt aus der Non-Secure-Welt unterbrochen werden kann. Da der M23 nur über zwei Prioritätsbits für nur vier mögliche Prioritäten verfügt, hat mal als Programmierer hier nicht viel Gestaltungsspielraum.



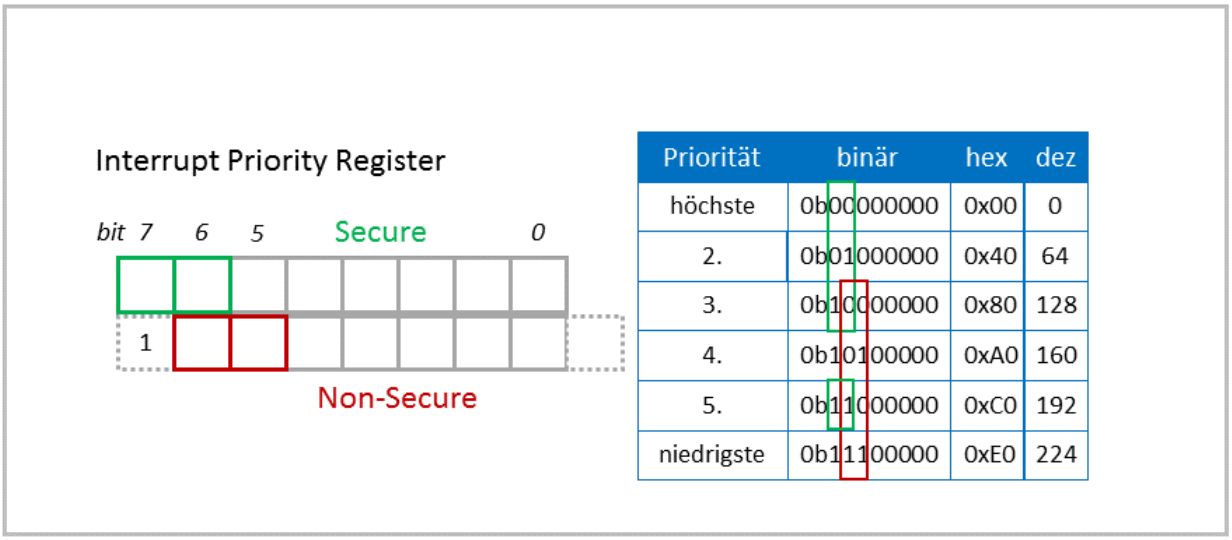
Priorität	binär	hex	dez
höchste	0b00000000	0x00	0
2.	0b01000000	0x40	64
3.	0b10000000	0x80	128
niedrigste	0b11000000	0xC0	192

Wenn Kundenprojekte im Non-Secure-Teil ablaufen, während im Hintergrund Security-Programme unerreichbar und unabhängig ablaufen sollen, wer stellt dann sicher, dass die Interrupts in der Non-Secure-Welt nicht auch hohe Prioritäten zugewiesen bekommen? Genau dafür gibt es das "Secure Exception Priority Boosting" Feature. Es wird durch Setzen des „PRIS"-Bits in der Secure-Welt aktiviert.

Durch die Aktivierung werden alle Prioritäten der Interrupts des Non-Secure-Bereichs einfach in der unteren Hälfte der Prioritätenskala komprimiert. Die Non-Secure-Welt bekommt davon nicht einmal etwas mit. Beim Lesen erscheint die Priorität als hoch, doch in Wirklichkeit liegt sie bei der Auswertung gerade mal im Mittelfeld. Doch noch geht die Rechnung nicht ganz auf.

Neubewertung von Prioritäten hinter den Kulissen

Bei der Untersuchung der Implementierungsdetails fiel auf, dass das Design der Cores abhängig von der Basic- oder Main-Architektur auf zwei unterschiedliche Arten umgesetzt wurden. And the winner is: die Baseline-Architektur des M23. Denn die aus der v6-M Architektur (Cortex-M0, M0+) übernommene Einschränkung der Priority Interrupts auf zwei Bits lässt ein Komprimieren der Interrupts auf die untere Hälfte kaum sinnvoll zu. Deshalb spendiert das Priority Boosting virtuell ein drittes Bit. Jeder der beiden Bereiche behält seine zwei Bits. Während die Secure-Welt weiterhin die Bits 7 und 6 nutzt, werden die beiden Non-Secure-Bits, wenn das „PRIS“-Bit gesetzt ist, bei der Bewertung der Prioritäten um eine Bitposition nach unten verschoben, also auf Bit 6 und 5. Wie schon erwähnt, passiert das hinter den Kulissen.



Priorität	binär	hex	dez
höchste	0b00000000	0x00	0
2.	0b01000000	0x40	64
3.	0b10000000	0x80	128
4.	0b10100000	0xA0	160
5.	0b11000000	0xC0	192
niedrigste	0b11100000	0xE0	224

Damit ist die Aufteilung der effektiven Prioritäten so angelegt, dass die beiden höchsten Prioritäten nur dem Secure-Bereich zur Verfügung stehen. Da bei gleicher Priorität keine Unterbrechung einer Interruptbehandlung stattfindet, ergeben sich folgende Szenarien:

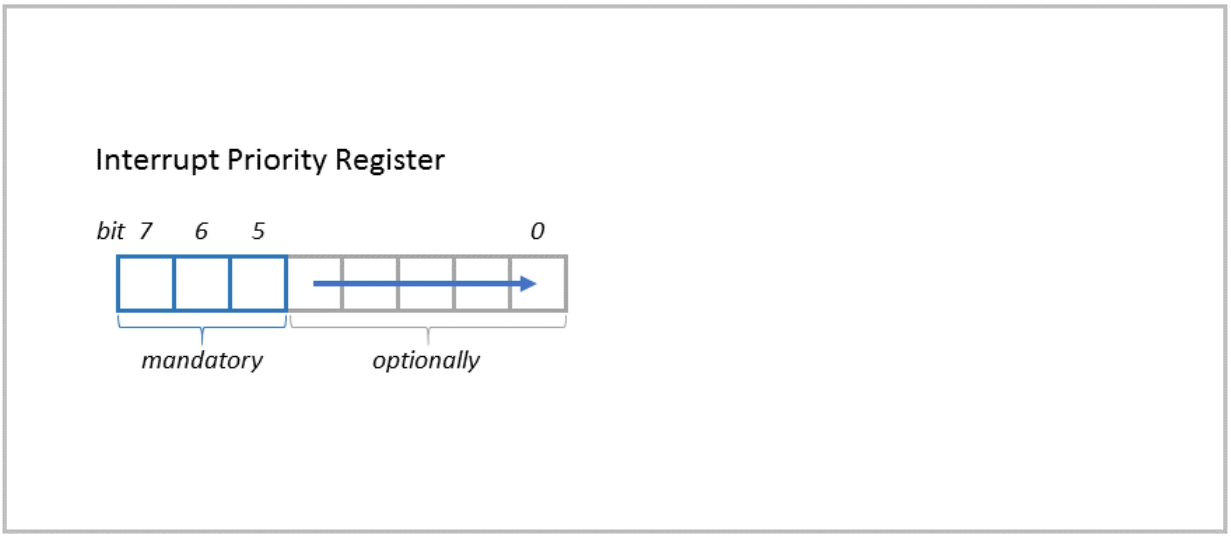
- Ein Interrupt in der Secure-World kann die Prioritäten 0, 64, 128 oder 192 haben.
- Wenn das PRIS-Bit nicht gesetzt und damit das Priority Boosting ausgeschaltet ist, gelten die gleichen Prioritäten auch für Interrupts in der Non-Secure-Welt.

Im weiteren Falle gilt, wenn das PRIS-Bit gesetzt ist und damit das Priority Boosting eingeschaltet ist:

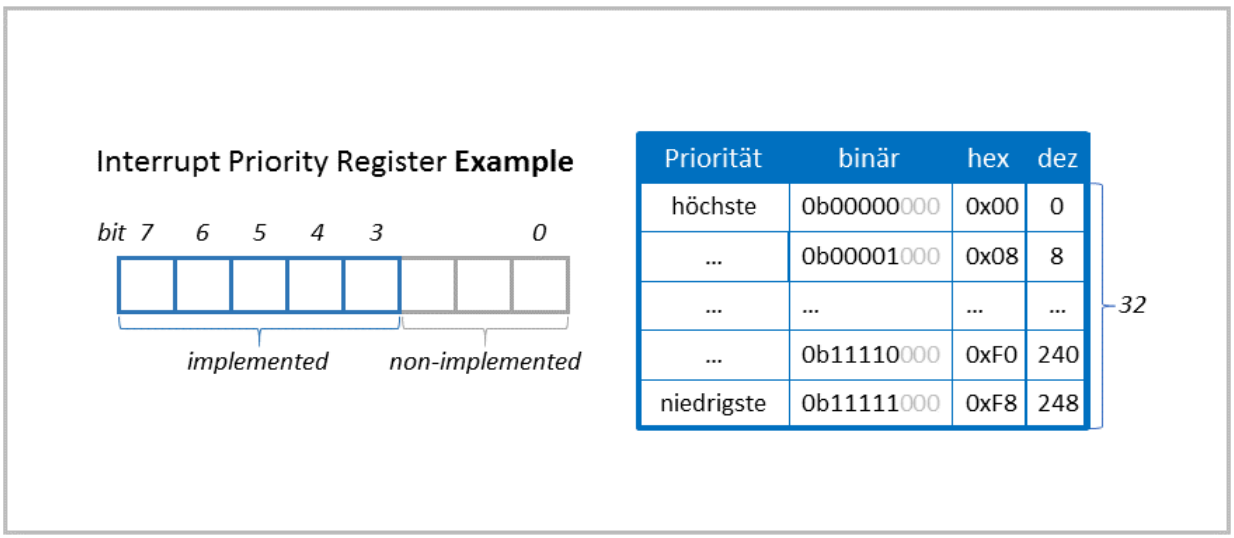
- Ein Interrupt in der Non-Secure-Welt kann die Prioritäten 128, 160, 192 oder 224 haben.
- Ein Interrupt in der Secure-Welt mit einer der beiden höchsten Prioritäten wird immer unabhängig vom Non-Secure-Bereich ausgeführt. Ggfs. wird dazu eine aktuelle Non-Secure-Behandlung unterbrochen.

- Ein Interrupt im Secure-Bereich kann nur durch einen Interrupt im Non-Secure-Bereich unterbrochen werden, wenn dieser selbst die unterste der vier Prioritäten hat (192) und der „Unterbrecher“ aus der Non-Secure-Welt die höchste oder zweithöchste (128 oder 160).
- Ein Interrupt im Secure-Bereich kann durch eine aktuell laufende Non-Secure-Unterbrechungsbehandlung verzögert werden, wenn beide die gleiche Priorität besitzen. Dieser Fall kann bei den Prioritäten 128 und 192 auftreten. Der jeweils spätere hängt sich mit dem Tail Chaining Feature an die Behandlung des aktuellen Interrupts hinten an.

In der **Main-Architektur** des M33, M35 und M55 stehen mehr Prioritätsbits zur Verfügung. Analog zur v7-M Architektur hat der Chip-Designer die Wahl zwischen mindestens drei und maximal acht Bits.



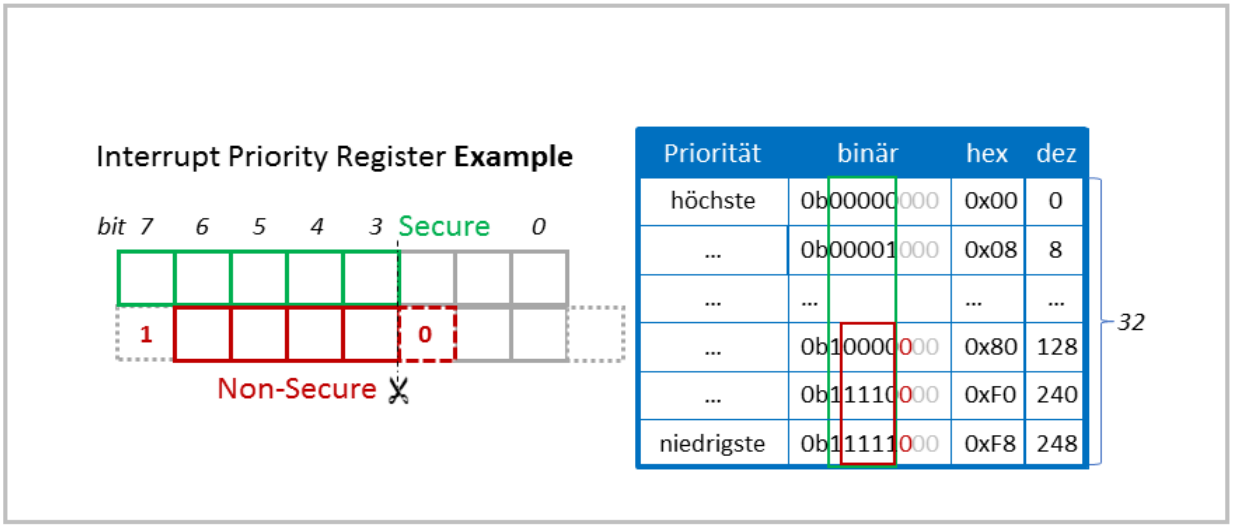
Hier ein Beispiel mit fünf implementierten Bits und den sich daraus ergebenden 32 möglichen Prioritäten:



Priorität	binär	hex	dez
höchste	0b00000000	0x00	0
...	0b00001000	0x08	8
...	...	...	...
...	0b11110000	0xF0	240
niedrigste	0b11111000	0xF8	248

32

Und auch hier werden durch das PRIS-Bit die Prioritäten in der Non-Secure-Welt um eine Bitposition nach rechts geschoben, das niederwertigste implementierte Bit geht dabei verloren und wird unabhängig vom gesetzten Bitwert als 0 gewertet:



Unerwünschtes Systemverhalten kostet ein Prioritätsbit

Doch dieses Verfahren birgt ein Problem: Durch Setzen oder Löschen des PRIS-Bits im Secure-Bereich wird das Preemption-Verhalten im Non-Secure-Bereich verändert. Bei ungünstigen Konstellationen führt das dazu, dass ein Non-Secure-Interrupt einmal einen anderen unterbricht und sich ein anderes Mal hinten anstellen muss. Aus der Non-Secure-Sicht handelt es sich dabei um ein nicht deterministisches Verhalten, da noch nicht einmal abgefragt werden kann, ob denn das PRIS-Bit auf der Secure-Seite gesetzt ist oder nicht.

Wie kommt es zu diesem unerwünschten Systemverhalten, und wie lässt es sich verhindern? Die gute Nachricht zuerst: Es lässt sich verhindern – jedoch auf Kosten eines Prioritätsbits, das auf der Non-Secure-Seite nicht benutzt werden darf.

Ein Beispiel zum besseren Verständnis:

- In unserem Beispielchip sind fünf Bits implementiert.
- Auf der Non-Secure-Seite sollen alle fünf Bits als Group Bits benutzt werden, d.h. keines der fünf Bits soll als Subprioritätsbit konfiguriert sein. Dies ist bereits ein v7-M Architektur-Feature und wird daher hier als bekannt vorausgesetzt (die Group Priority muss auf einen der Werte 5:3, 6:2 oder 7:1 eingestellt werden).
- Ein Interrupt A soll auf der Non-Secure-Seite mit der zweitniedrigsten Priorität behandelt werden, also mit Priorität 240.
- Ein Interrupt B soll auf der Non-Secure-Seite mit der niedrigsten Priorität behandelt werden, also mit Priorität 248.
- Der Interrupt B wird gerade durch seine Interrupt-Routine ausgeführt.
- Nun steht der Interrupt A, als der Interrupt mit der höheren Priorität zur Behandlung an:
    - **Szenario 1** – Das PRIS-Bit auf der Secure-Seite ist nicht gesetzt:
      Die Interruptbehandlung von Interrupt B wird erwartungsgemäß unterbrochen und nach Beendigung der Interruptbehandlung A weiter ausgeführt.

    - **Szenario 2** – Das PRIS-Bit auf der Secure-Seite ist gesetzt:
      Die Interruptbehandlung von Interrupt B wird NICHT unterbrochen. Die Behandlung des vermeintlich höherprioren Interrupts A hängt sich mittels Tail Chaining an die Behandlung des Interrupts B hinten an.
- Die Ursache für dieses unerwartete Verhalten liegt in der Bitverschiebung bei der Prioritätsbehandlung. Die höhere Priorität 240 (0b11110000) wird durch das Schieben und Einfügen einer führenden 1 zu 248 (0b11111000) und ist damit genauso hoch wie Interrupt B.

Eine Vermeidungsstrategie könnte sein, das letzte implementierte Prioritätsbit einfach nicht zu verwenden und mit der Hälfte der implementierten Prioritäten zu leben. Schon wird das unerwünschte Verhalten vermieden.

Wenn das letzte implementierte Bit beispielsweise immer auf 0 gesetzt wird, dann spielt es auch keine Rolle, ob es bei der Prioritätsauswertung abgeschnitten wird oder nicht, abhängig davon, ob gerade das PRIS-Bit auf der Secure-Seite gesetzt ist oder nicht.

Fazit

In der v8-M Baseline-Architektur des M23 bietet es sich an, das Priority Boosting einzusetzen, da dadurch ein virtuelles drittes Bit hingezaubert wird.

Beim M33, M35 und M55 sollte man sicherstellen, dass das Priority Boosting auf der Secure-Seite nicht verwendet wird. Von der Non-Secure-Seite aus lässt sich das nur weder verhindern noch überprüfen. Also sollte man besser auf der Non-Secure-Seite nur jede zweite Priorität nutzen, dann hat das Priority Boosting keine unerwünschten Nebenwirkungen.

Weiterführende Informationen

MicroConsult Fachwissen zum Thema Mikrocontroller

MicroConsult Training & Coaching zum Thema Arm/Cortex Mikrocontroller

Alle MicroConsult Trainings & Coachings

Autor:

Remo Markgraf ist Senior Management Consultant bei der MicroConsult GmbH. Neben Begeisterung für Innovation und Leidenschaft für Embedded-Systeme verfügt er über langjährige Projekt- und internationale Führungserfahrung in Softwareentwicklung, Systems Engineering, Projekt-, Produkt-, Innovations- und Business Development Management sowie dem technischen Vertrieb.