SWP Telematik

ARM PSA/Trustzone

ARM PSA/Trustzone

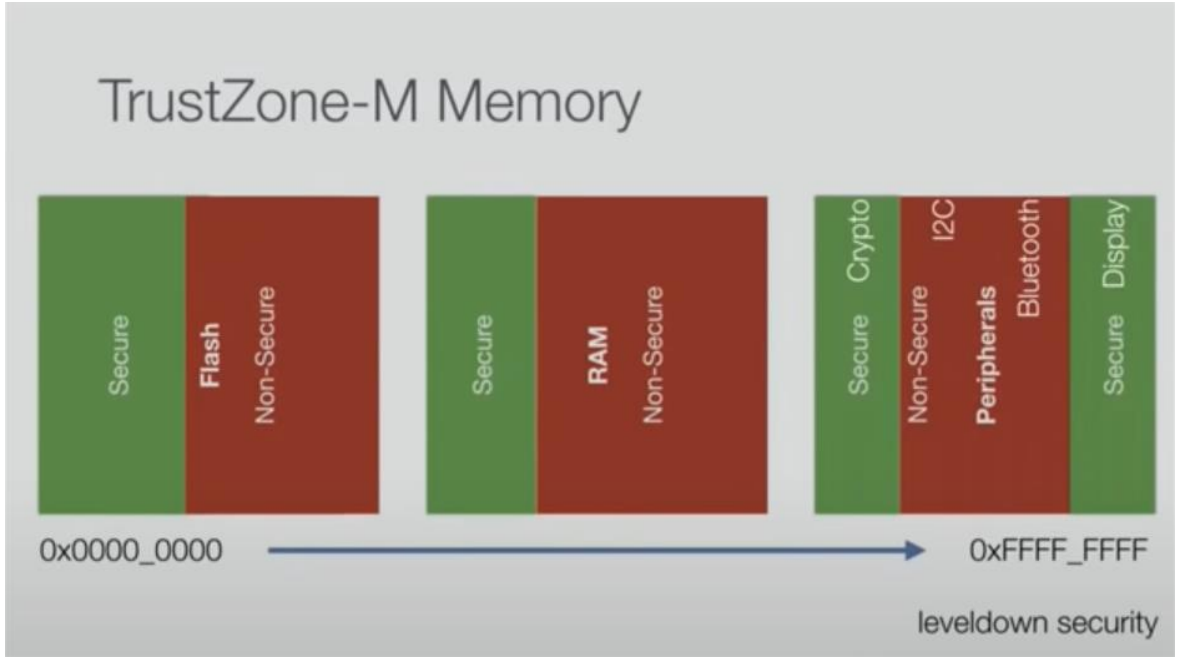
Onur Demir

Nima Thobae

Arsalan Barazesh

Berthold Bußkamp

- Trustzone Cortex M ist eine kleinere Version von Trustzone Cortex A
- Es teilt CPU in einen sicheren und unsicheren Modus auf.
- Es erlaubt uns, Flash oder Ram in sicheren óder unsicheren Teile zu partitionieren.
- Mit Trustzone können wir ein zweites Betriebsystem haben.



- Alle mögliche Angriffe:

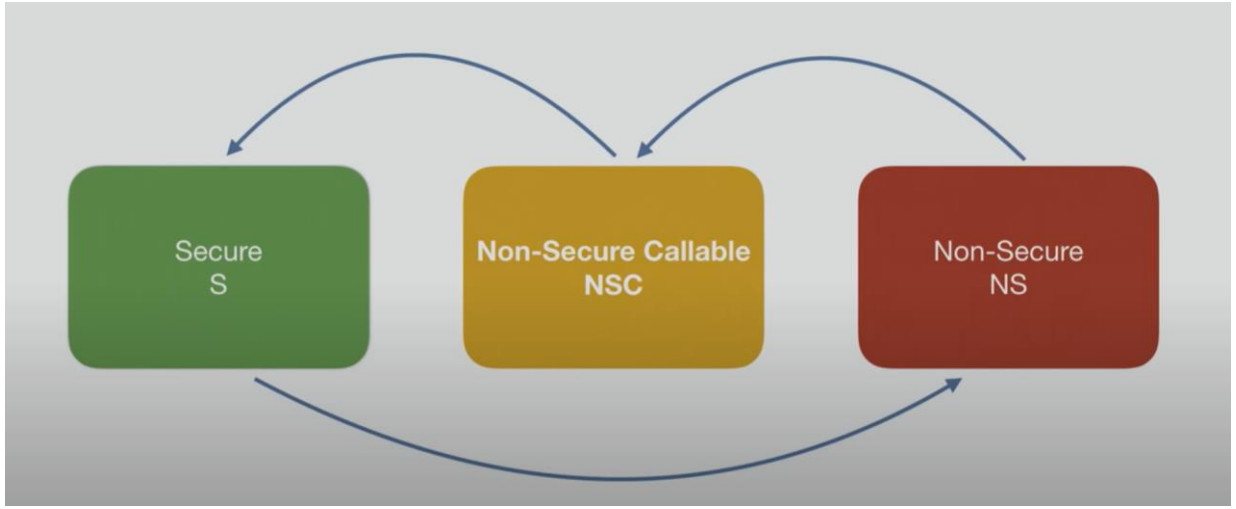
Software:
- Remote Software Attack
- Local Software Attack
- Software Side Channel Attack

Hardware:
- Debug Parts
- Side Channel Attack
- Fault Injection

Voltage Glitching: Skip instructions / corrupt flash or memory read&write

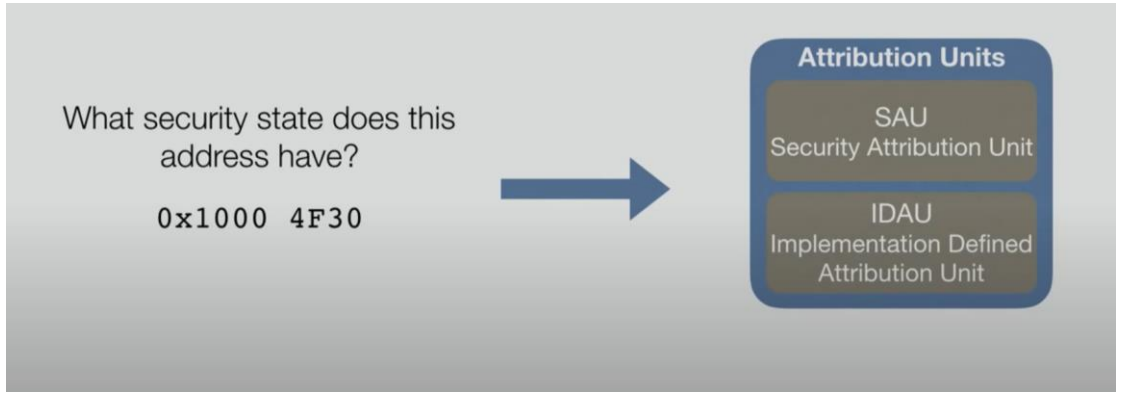
- Flat Memory Map:
  Alle Speichern sind zu einer bestimmten Adresse zugeordnet.

- Secure(S) – Nonsecure(NS) – NonSecureCallable(NSC):
  - Wir sind in der Lage, eine Funktion von einem unsicheren Bereich in einem sicheren Bereich aufzurufen.
  - Aber umgekehrt geht das nicht. Dazu brauchen wir NSC.



Attribution Unit:

Mit AU können wir die Kommunikation zwischen diese Speichern (S-NS-NSC) verwalten.
Wenn wir AU brechen oder deaktivieren können, brechen wir einfach die Sicherheit.



Wenn SAU an ist, sind alle Bereiche sicher.
Wenn SAU aus ist und kein Bereich ist als unsicher markiert, ist alles immernoch sicher.

SAU_CTRL	SAU Control Register
SAU_TYPE	Number of supported regions
SAU_RNR	Region Number Register
SAU_RBAR	Region Base Address
SAU_RLAR	Region Limit Address

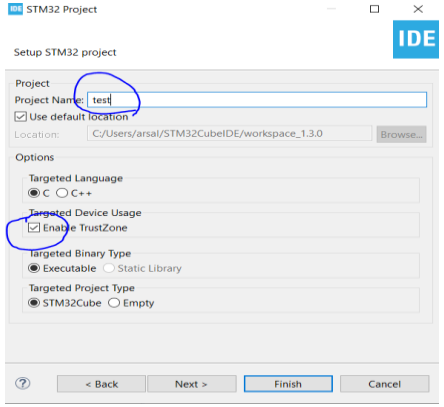
Beispiel für SAU:

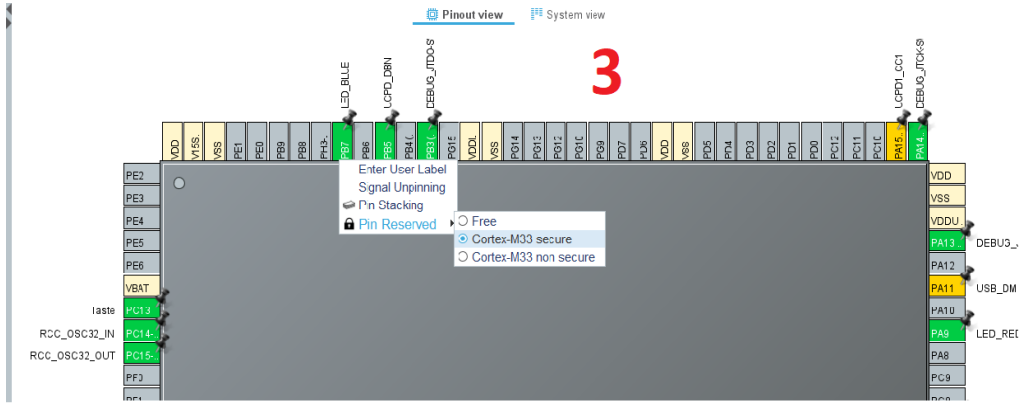
Select region 0	SAU_PNR= 0x0
Base	SAU_PBAR= 0X1000
Set limit Address to 0x1fff	SAU_RLAR= 0x1FE0
Enable SAU	SAU_CTRL= 0x1

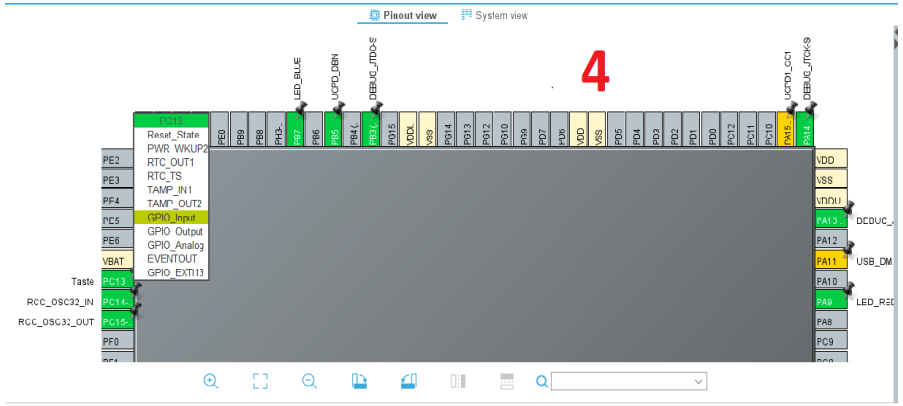
Bei obigen Eingaben wird alle Adressen zwischen 0x1000 und 0x1FFF als NS markiert.
Bei uns ist das in Partition_stm32l552xx.h definiert.

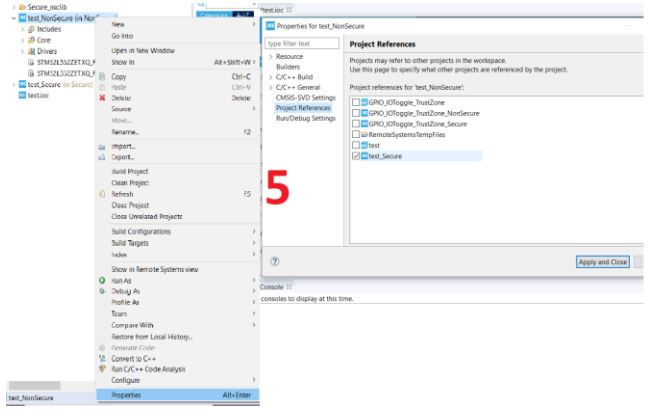
Beispiel Projekt:
Wir haben auf PIN PB7 eine LED und auf Pin PC13 eine Taste.
Die Betätigung der Taste wird in unsecure-mode überprüft.
Und LED-Toggling wird in Secure-Mode aufgeführt.
Dazu brauchen wir eine Definition der Funktion in der Datei secure_nsc.c
Außerdem sollen wir was in Nonsecure-Eigenschaften und Debug-Egenschaften ändern. Damit machen wir Ausführung von Trustzone & NonTrustzone abhängig.

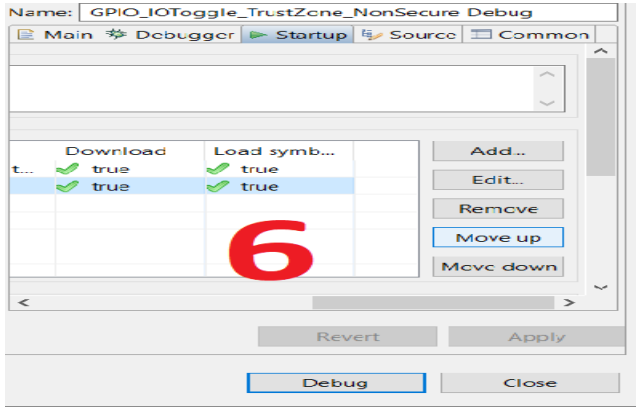
1. File → New → STM32 Project
2. Boardselector → NUCLEO-L552ZE-Q
3. Rightclick on PB7 pin and select Cortex m33 secure
4. Click on PC13 and select GPIO_input. Rightclick on PC13 pin and select Cortex m33 unsecure. Rightclick again change the user label to Taste.
5. Rightclick on Test_nonSecure → Properties → Project references → check Test_secure like picture 5
6. Rightclick on Test_Nonsecure → Debug as → Debug Cofiguration → Select Test_Nonsecure → On startup tap select ADD → select project test secure like picture 7 → moveup nonsecure like picture 6

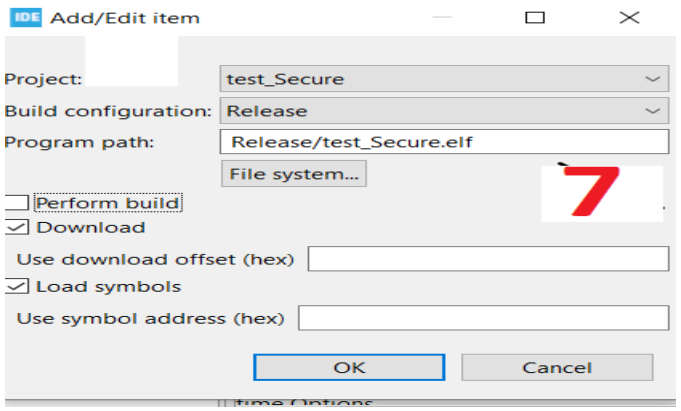












Beim Secure-Modus:

in secure_nsc.h:

definieren wir Toggle-funktion:

```
void my_toggle(void);
```

in secure_nsc.c:

definieren wir Toggle-funktion:

```
CMSE_NS_ENTRY void my_toggle(void){
        HAL_GPIO_TogglePin(LED2_GPIO_Port, LED2_Pin);
}
```

Beim Secure-Modus:

in main.c in der endlose Schleife:

```
while (1){
        /* USER CODE END WHILE */
        if (HAL_GPIO_ReadPin(TasteNS_GPIO_Port, TasteNS_Pin)==GPIO_PIN_SET){
                my_toggle();
        }
        HAL_Delay(1000);
        /* USER CODE BEGIN 3 */
}
```

Helloworld hinzufügen:

```
 while (1)
{
/* USER CODE END WHILE */
        if (HAL_GPIO_ReadPin(TasteNS_GPIO_Port, TasteNS_Pin)==GPIO_PIN_SET){
                arsalantoggle();
                strcpy((char*)buf,"Hello!\r\n");
                HAL_UART_Transmit(&huart1, bud, strlen((char *)buf), HAL_MAX_DELAY);
        }
        HAL_Delay(1000);
/* USER CODE BEGIN 3 */
}
```

Unter windows können wir ein COM66 Port definieren und über Putty (115200)die Ausgabe sehen.
Unter Línux und Mac steht das in /dev/tty*

Quellen:

https://www.youtube.com/watch?v=4u6BAH8mEDw

https://www.st.com/content/st_com/en/about/events/events.html/stm32cubeide-for-stm32l5-microcontroller.html

https://www.youtube.com/watch?v=isOekyygpR8&list=PLEBQazB0HUyRYuzfi4clXsKUSgorErmBv&index=2