IoT Security

STM32 Security

PV285

Václav Oujezský, Karel Slavíček, Tomáš Pitner

Introduction

- Protection of firmware
- Protection of private data in the device
- Guarantee of a service execution

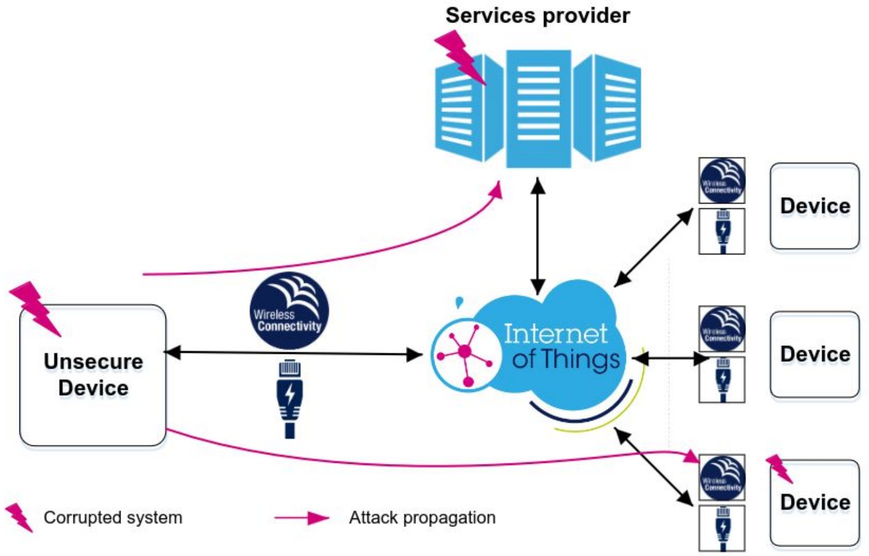
With IoT, the security extends the requirements for **confidentiality** and **authentication** to communication channels, which often require encryption.

Why protection is needed

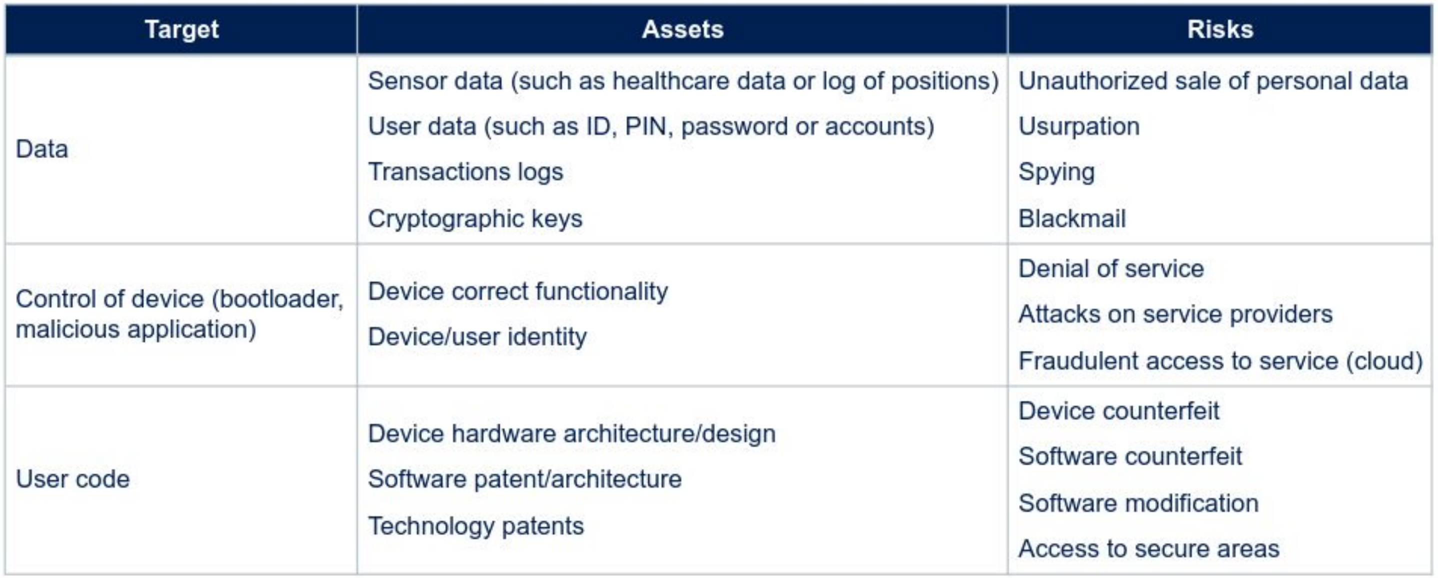
- Reverse-engineering
- Copy custom algorithms
- Flashing or cloning hardware
- Denial of service
- IoT connected devices build around microcontrollers are very attractive for hackers

Why protection is needed

– A single compromised device can corrupts the integrity of an entire network!



What must be protected



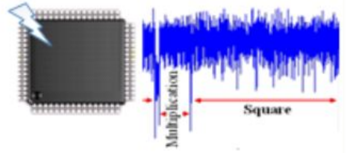
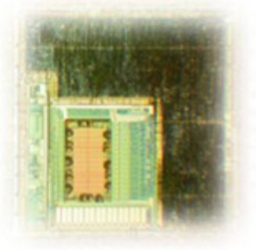
Target	Assets	Risks
Data	Sensor data (such as healthcare data or log of positions) User data (such as ID, PIN, password or accounts) Transactions logs Cryptographic keys	Unauthorized sale of personal data Usurpation Spying Blackmail
Control of device (bootloader, malicious application)	Device correct functionality Device/user identity	Denial of service Attacks on service providers Fraudulent access to service (cloud)
User code	Device hardware architecture/design Software patent/architecture Technology patents	Device counterfeit Software counterfeit Software modification Access to secure areas

Attack types

Attacks on microcontroller are classified in one of the following types

- **Software attack**: exploits software vulnerabilities (such as bug or protocol weaknesses).
- **Hardware non-invasive attack**: focuses on MCU interfaces and environment information.
- **Hardware invasive attack:** destructive attack with direct access to silicon

Attack types

Attacks types	Non-invasive	Semi–invasive	Invasive
-			
Scope	Remote or local	Local board and device level	Local device level
Techniques	Software bugs Protocol weaknesses Trojan horse Eavesdropping	Debug port Power glitches Fault injection Side-channels analysis	Probing Laser FIB Reverse engineering
Cost/ expertise	From very low to high, depending on the security failure targeted	Quite low cost. Need only moderately sophisticated equipment and knowledge to implement.	Very expensive. Need dedicated equipment and very specific skills.
Objectives	Access to confidential assets (code and data). Usurpation Denial of service	Access to secret data or device internal behavior (algorithm).	Reverse engineering of the device (silicon intellectual property) Access to hidden hardware and software secrets (flash memory access)

Software attacks

- Malware, takes control of the device or modifies its functionoality
- Insider thread: firmware
- Must be injected (RAM, flash) and executed by the CPU
- Open doors are debug ports (JTAG, SWD), bootloader, external memory, firmware updates, communication ports.
- Buffer overflow or data attacks
- Low level languages such as C/C++ are unsafe, can lead to memory leaks

Brute forcing

- Authentication based attacks on a human machine interface (HMI)

Hardware attacks

Physical access to the device

- **Invasive attacks**: very expensive, direct access to device silicon (silicon invasive attacks)
- **Non-invasive attacks** (next slides)

Non-invasive attacks

- **Debug port access** or scan chain via JTAG, SWD - deactivation by Readout protection (RDP)
- **Serial port access** (I2C, SPI): software level secure by encryption, isolation of sensitive data, check the data transfer to avoid buffer overflow
- **Fault injection:** clock and power disturbance/glitch attacks, using the device outside the parameters defined in the datasheet to generate malfunctions in the system.

Fault injection - continue

Countermeasures:

Return values checks, strict branching, using non-trivial values as true or false and not 0 or -1, for example mutual Hamming distance

If available, use Clock security system (CSS), internal clock, internal voltage, memory error detection

Non-invasive attacks

Side-channel attacks (SCA)

- Observing the device running characteristics such as power consumption, radiations, temperature, activity time, etc.
- Typical attacks are Simple power analysis (SPA) and Differential power analysis (DPA)
- Protecting by session random keys or cryptographic libraries with behavioral randomization
- Hardware protection is reflected by the certification level (product certification)

Device protections

- **Memory protection:** main security feature, used to protect code and data from internal (software) and external attacks
- **Software isolation:** inter-processes protection to avoid internal attacks
- **Interface protection:** used to protect device entry points like serial or debug ports
- **System monitoring:** detects device external tampering attempts or abnormal behaviors

Configuration protection

Persistent security configuration is stored in a dedicated area of the non-volatile memory, called option bytes (OB).
The configuration is generally frozen by raising the **RDP level**. Additional rules apply case-by-case. Each OB value is preconfigured to a reset value. In case of an error detected in OB loading, the configuration is set to a default value. This default value usually sets the highest possible security level to prevent leaking information from damaged microcontroller.

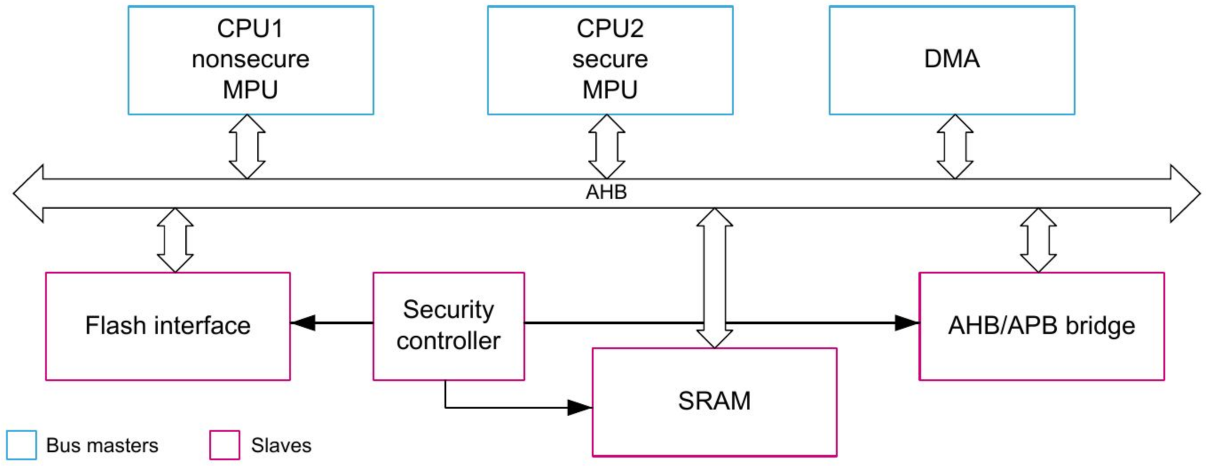
RDP

The STM32F0 and STM32F1 microcontroller series from STMicroelectronics incorporate different security features to protect their firmware, reflecting an evolution in approaches to securing embedded systems.

- **STM32F0**: This series permanently disables the debug interface to prevent unauthorized access through debugging tools. This method provides a straightforward way to secure the chip from firmware extraction or manipulation through standard debug protocols like JTAG or SWD (Serial Wire Debug).
- **STM32F1**: Unlike the F0, the F1 series retains an enabled debug interface, which is often crucial for legitimate development and troubleshooting purposes. To safeguard the firmware, STM32F1 uses flash memory read-out protection (RDP). This protection mechanism is designed to prevent external devices from reading the flash contents. However, this method has vulnerabilities.

Dual-core architecture

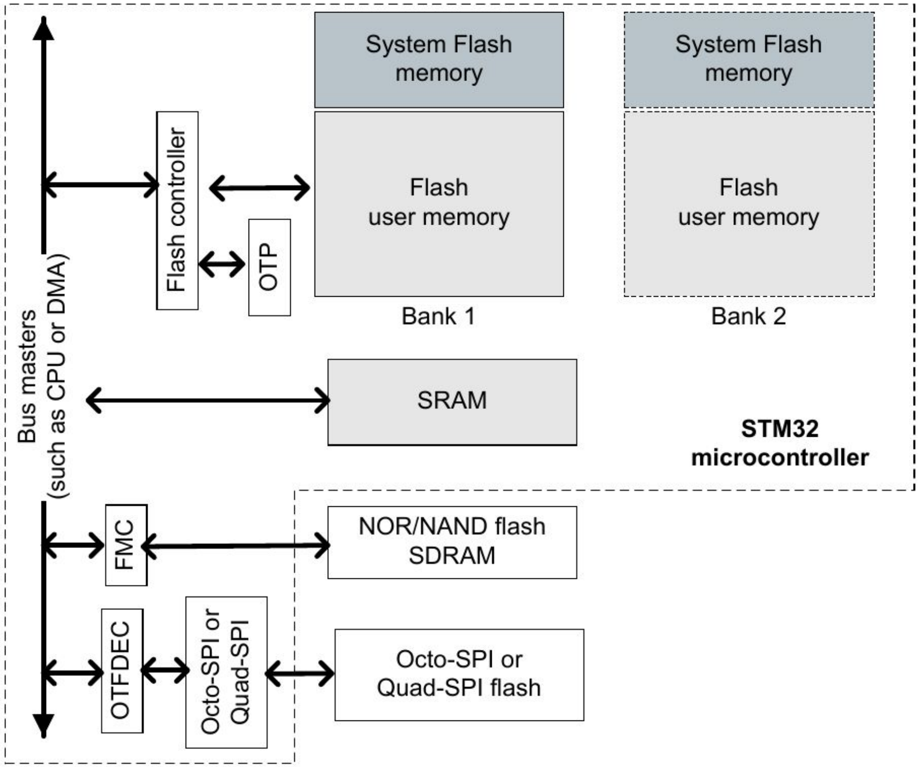
In systems like the STM32WL, one core can operate in a secure mode, while the other operates in a non-secure mode. This separation allows sensitive tasks and data to be handled by the secure core, isolating them from less critical operations that the non-secure core handles.



Memory protection

- Firewall
- Flash controller
- PCROP - proprietary code read-out protection feature
- WRP - write protection
- RDP

https://www.st.com/en/embedded-software/x-cube-pcrop.html



Debug port and interface protection

STM32 features:

- read protection (RDP)
- disable of unused ports
- bootloader access forbidden (configured by RDP in STM32 devices)

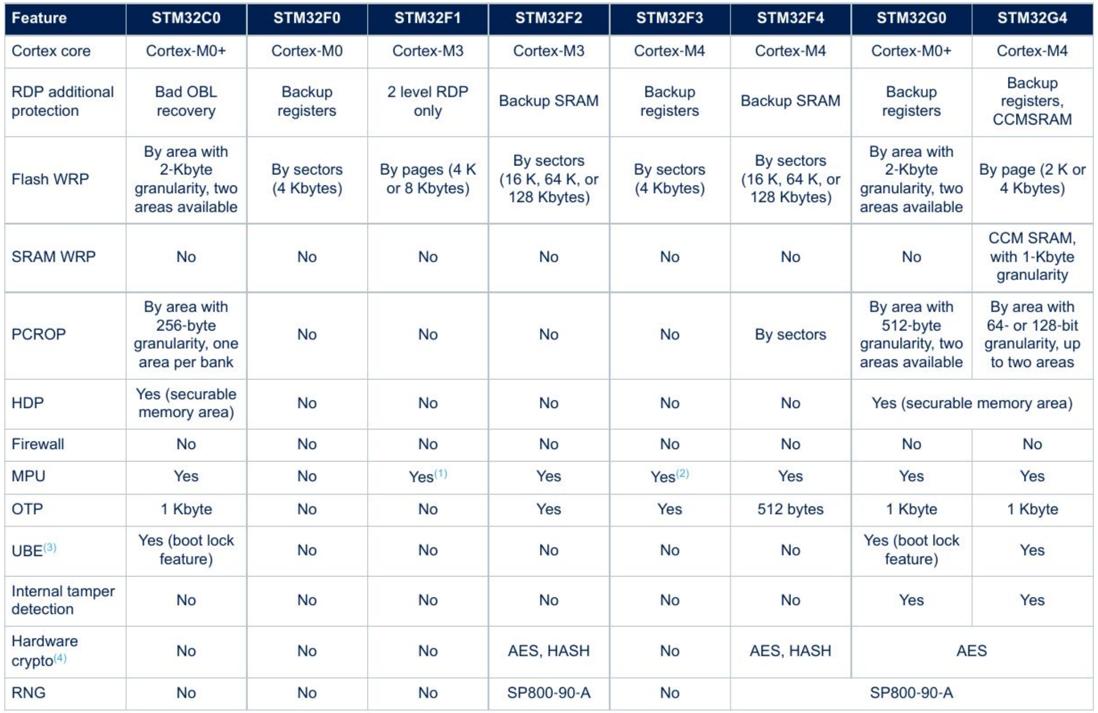
Boot protection

The boot protection relies on a single entry point to a trusted code that can be the user application, or a secure service area if available (RSS).

STM32 features:

- read protection (RDP)
- unique boot entry
- secure hide protection (HDP)
- TrustZone

Security features



Feature	STM32C0	STM32F0	STM32F1	STM32F2	STM32F3	STM32F4	STM32G0	STM32G4
Cortex core	Cortex-M0+	Cortex-M0	Cortex-M3	Cortex-M3	Cortex-M4	Cortex-M4	Cortex-M0+	Cortex-M4
RDP additional protection	Bad OBL recovery	Backup registers	2 level RDP only	Backup SRAM	Backup registers	Backup SRAM	Backup registers	Backup registers, CCMSRAM
Flash WRP	By area with 2-Kbyte granularity, two areas available	By sectors (4 Kbytes)	By pages (4 K or 8 Kbytes)	By sectors (16 K, 64 K, or 128 Kbytes)	By sectors (4 Kbytes)	By sectors (16 K, 64 K, or 128 Kbytes)	By area with 2-Kbyte granularity, two areas available	By page (2 K or 4 Kbytes)
SRAM WRP	No	No	No	No	No	No	No	CCM SRAM, with 1-Kbyte granularity
PCROP	By area with 256-byte granularity, one area per bank	No	No	No	No	By sectors	By area with 512-byte granularity, two areas available	By area with 64- or 128-bit granularity, up to two areas
HDP	Yes (securable memory area)	No	No	No	No	No	Yes (securable memory area)	
Firewall	No	No	No	No	No	No	No	No
MPU	Yes	No	Yes[1]	Yes	Yes[2]	Yes	Yes	Yes
OTP	1 Kbyte	No	No	Yes	Yes	512 bytes	1 Kbyte	1 Kbyte
UBE[3]	Yes (boot lock feature)	No	No	No	No	No	Yes (boot lock feature)	Yes
Internal tamper detection	No	No	No	No	No	No	Yes	Yes
Hardware crypto[4]	No	No	No	AES, HASH	No	AES, HASH	AES	
RNG	No	No	No	SP800-90-A	No	SP800-90-A		

Readout protection (RDP) in detail

It is a global flash memory protection allowing the embedded firmware code to be protected against copy, reverse engineering, dumping, using debug tools, or code injection in SRAM. The user must set this protection after the binary code is loaded to the embedded flash memory.

The RDP applies to all STM32 devices for:

- the main flash memory
- the option bytes (level 2 only)

Depending on the STM32 device, additional protections are available, including:

- backup registers for real-time clock (RTC)
- backup SRAM
- nonvolatile memories

Readout protection (RDP) in detail

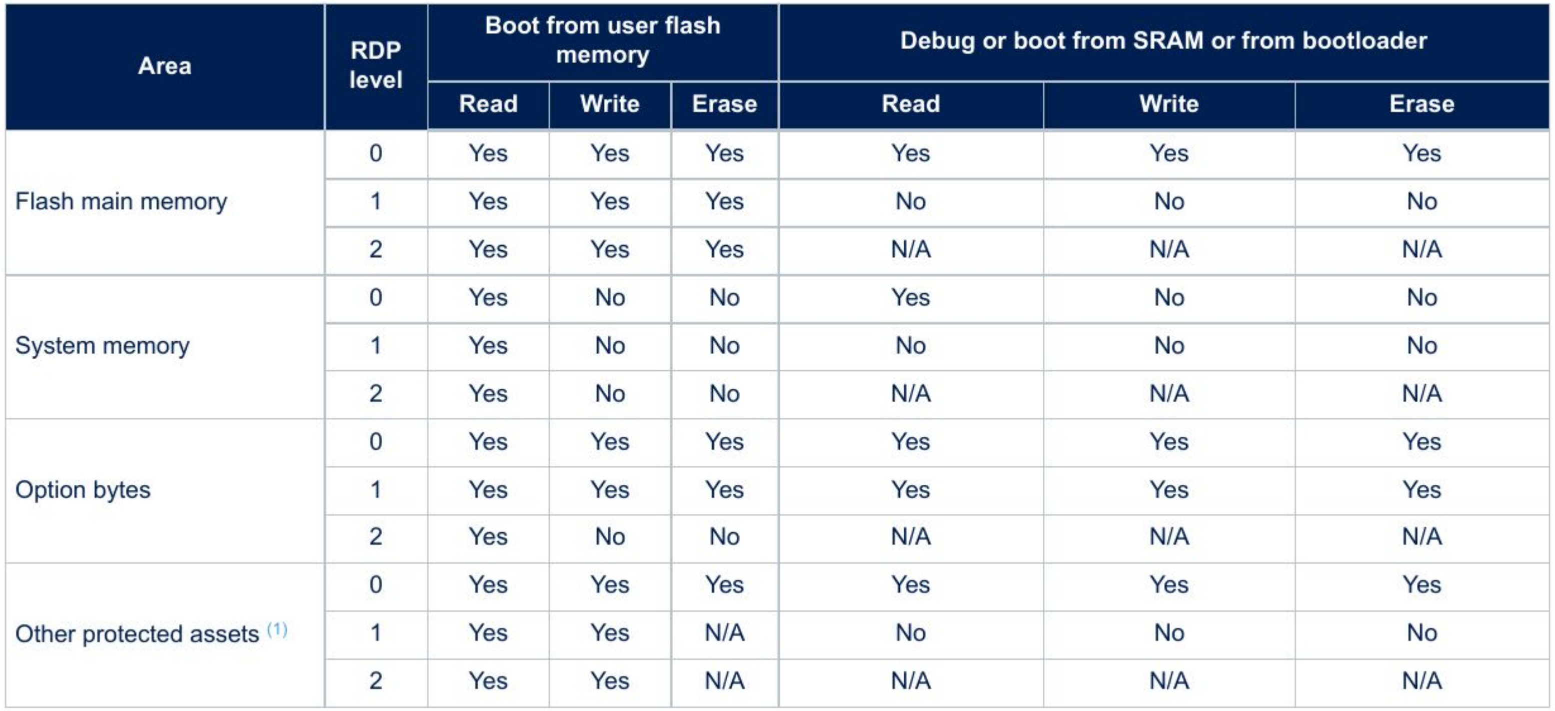
- **Level 0 (default RDP level)**: The flash memory is fully open, and all memory operations are possible in all boot configurations (debug features, boot from RAM, boot from system memory bootloader, boot from flash memory). There is no protection in this configuration mode that is appropriate only for development and debug.
- **Level 1:** Flash memory accesses (read, erase, program), or SRAM2 accesses via debug features (such as serial-wire or JTAG) are forbidden, even while booting from SRAM or system memory bootloader. In these cases, any read request to the protected region generates a bus error. However, when booting from flash memory, accesses to both flash memory and to SRAM2 (from user code) are allowed.
- **Level 2:** All protections provided in Level 1 are active, and the MCU is fully protected. The RDP option byte and all other option bytes are frozen, and can no longer be modified. The JTAG, SWV (single-wire viewer), ETM, and boundary scan are all disabled.

RDP level regression

A level regression is possible with the following consequences:

- Regression from RDP level 1 to RDP level 0 leads to a flash memory mass erase, and the erase of SRAM2 and backup registers.
- Regression from RDP level 1 to RDP level 0.5 leads to a partial flash memory erase: only the nonsecure part is erased.
- Regression from RDP level 0.5 to RDP level 0 leads to a flash memory mass erase, and the erase of SRAM2 and backup registers.
- In RDP level 2, with exception of preconfigured OEM keys in STM32Ux series, no regression is possible.

RDP protection



Area	RDP level	Boot from user flash memory			Debug or boot from SRAM or from bootloader		
		Read	Write	Erase	Read	Write	Erase
Flash main memory	0	Yes	Yes	Yes	Yes	Yes	Yes
	1	Yes	Yes	Yes	No	No	No
	2	Yes	Yes	Yes	N/A	N/A	N/A
System memory	0	Yes	No	No	Yes	No	No
	1	Yes	No	No	No	No	No
	2	Yes	No	No	N/A	N/A	N/A
Option bytes	0	Yes	Yes	Yes	Yes	Yes	Yes
	1	Yes	Yes	Yes	Yes	Yes	Yes
	2	Yes	No	No	N/A	N/A	N/A
Other protected assets [1]	0	Yes	Yes	Yes	Yes	Yes	Yes
	1	Yes	Yes	N/A	No	No	No
	2	Yes	Yes	N/A	N/A	N/A	N/A

LAB RDP

LAB: RDP https://stm32world.com/wiki/STM32_Readout_Protection_(RDP)

- Review STM32 RDP levels and their impact on access to memory.
- Use STM32CubeProgrammer to apply different RDP levels to an STM32 microcontroller and observe the outcomes.
- Discuss the process and implications of changing from RDP Level 1 to Level 0, emphasizing the data loss that occurs