BYPASS NXP LPC-FAMILY DEBUG CHECK WITH VOLTAGE FAULT INJECTION

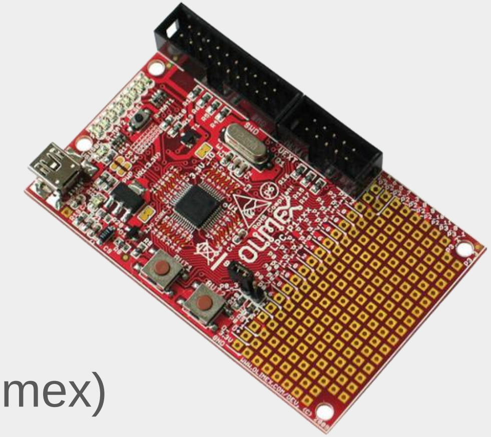
Waleed AlZamil
Embedded devices security researcher
@WaleedAlzamil

Bandar Alharbi
Cybersecurity researcher
@0xB4x

AGENDA

- Introduction
  - LPC
  - Goal
  - Security mechanism
- BootROM analysing & reverse engineering
  - Memory map & Boot process
  - Reverse engineering BootROM
  - Finding the vulnerability
- What is Fault injection (Glitching)?
- Glitching the LPC1343
  - Attack Setup
  - Power analysis
  - Recorded demo
- Conclusion

INTRODUCTION



- LPC1343 is a Cortex-M3 MCU (pic of dev board from Olimex)
- Harvard architecture
- Microcontroller for embedded applications featuring a high level of integration and low power consumption
- Previous work on the LPC1343/… @Chris Gerlinsky

GOAL

1. Study LPC debug locking mechanism

2. Targeting to unlock debug interface

LPC CODE READ PROTECTION

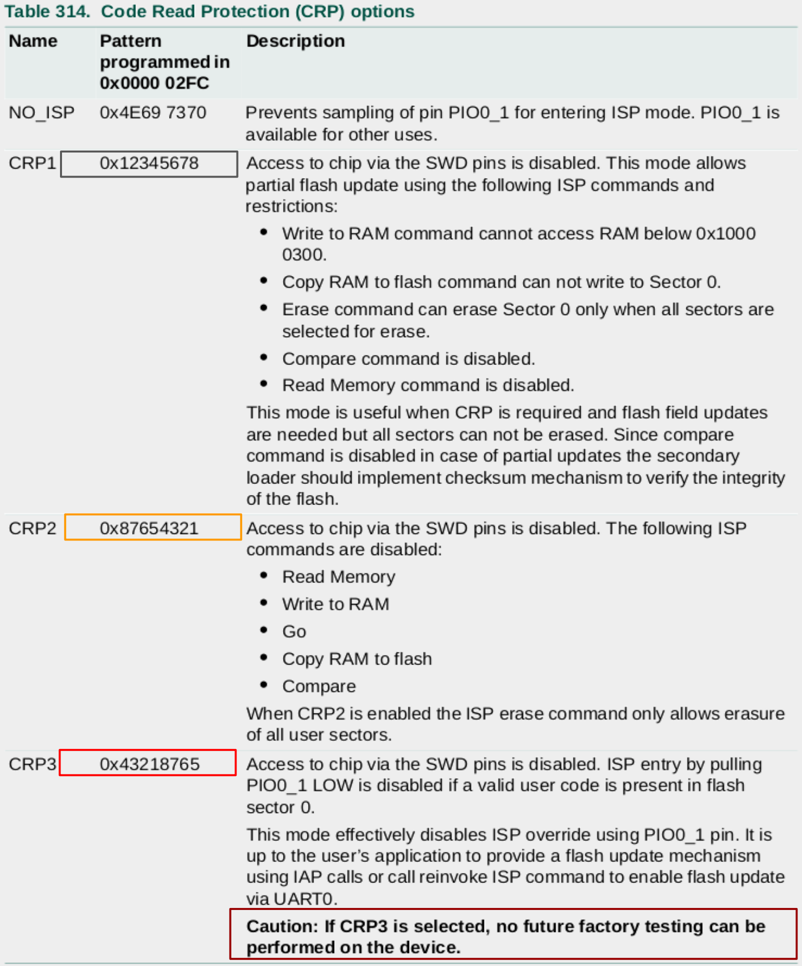


Table 314. Code Read Protection (CRP) options

Name	Pattern programmed in 0x0000 02FC	Description
NO_ISP	0x4E69 7370	Prevents sampling of pin PIO0_1 for entering ISP mode. PIO0_1 is available for other uses.
CRP1	0x12345678	Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions: • Write to RAM command cannot access RAM below 0x1000 0300. • Copy RAM to flash command can not write to Sector 0. • Erase command can erase Sector 0 only when all sectors are selected for erase. • Compare command is disabled. • Read Memory command is disabled. This mode is useful when CRP is required and flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the flash.
CRP2	0x87654321	Access to chip via the SWD pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to flash • Compare When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x43218765	Access to chip via the SWD pins is disabled. ISP entry by pulling PIO0_1 LOW is disabled if a valid user code is present in flash sector 0. This mode effectively disables ISP override using PIO0_1 pin. It is up to the user's application to provide a flash update mechanism using IAP calls or call reinvoke ISP command to enable flash update via UART0. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

LPC CODE READ PROTECTION

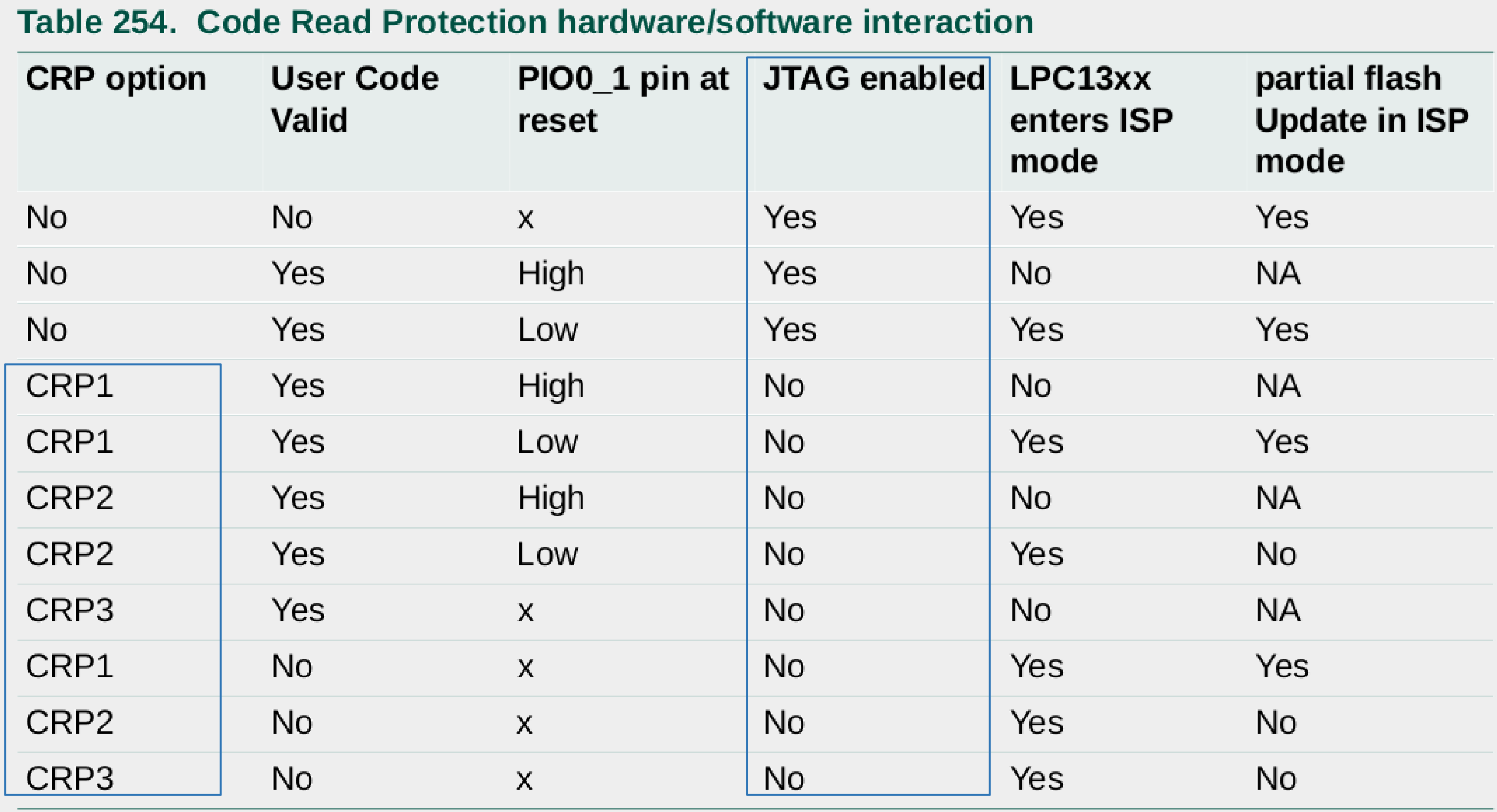
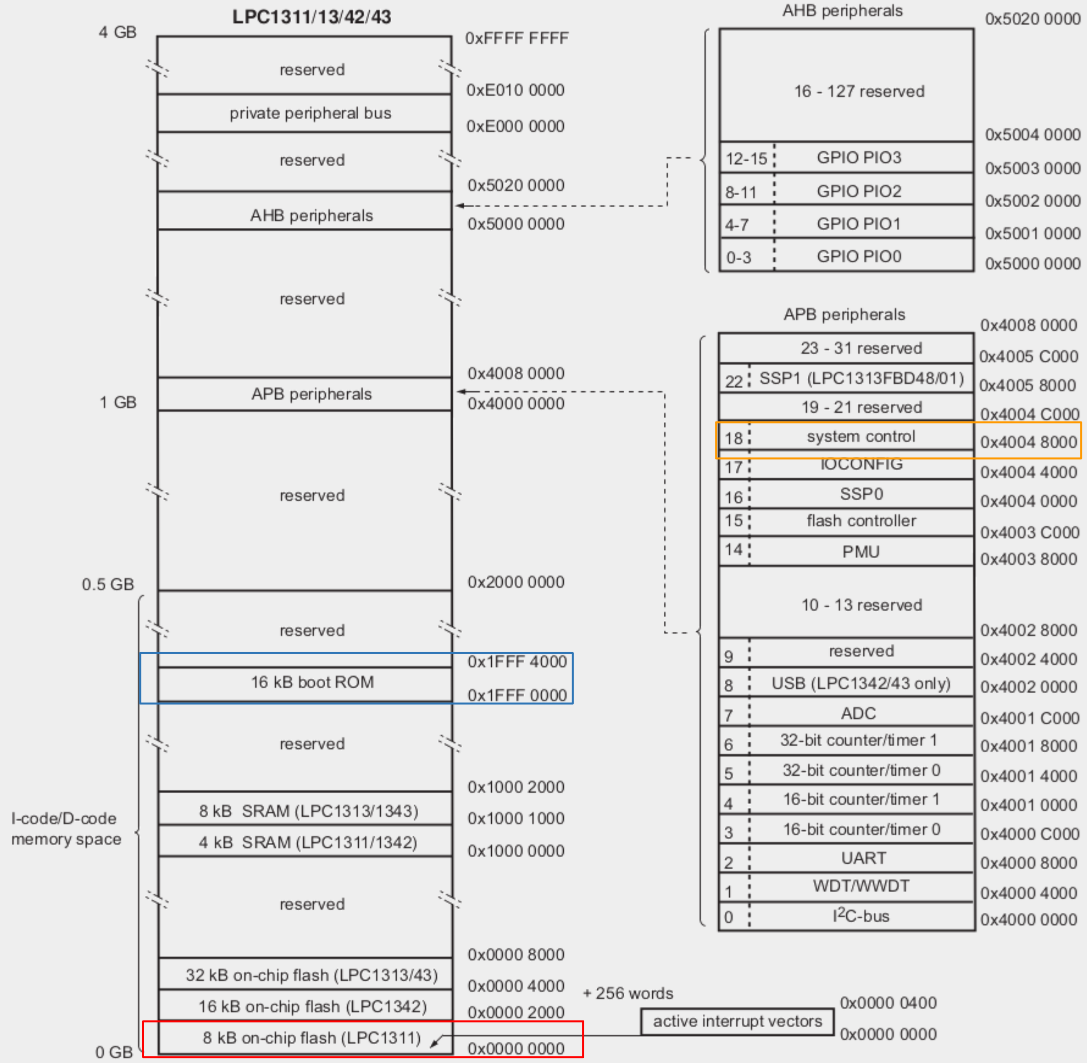


Table 254. Code Read Protection hardware/software interaction

CRP option	User Code Valid	PIO0_1 pin at reset	JTAG enabled	LPC13xx enters ISP mode	partial flash Update in ISP mode
No	No	x	Yes	Yes	Yes
No	Yes	High	Yes	No	NA
No	Yes	Low	Yes	Yes	Yes
CRP1	Yes	High	No	No	NA
CRP1	Yes	Low	No	Yes	Yes
CRP2	Yes	High	No	No	NA
CRP2	Yes	Low	No	Yes	No
CRP3	Yes	x	No	No	NA
CRP1	No	x	No	Yes	Yes
CRP2	No	x	No	Yes	No
CRP3	No	x	No	Yes	No

LPC13xx MEMORY MAP



LPC1311/13/42/43

4 GB
0xFFFF FFFF
reserved
0xE010 0000
private peripheral bus
0xE000 0000
reserved
0x5020 0000
AHB peripherals
0x5000 0000
reserved
0x4008 0000
APB peripherals
1 GB
0x4000 0000
reserved
0.5 GB
0x2000 0000
reserved
0x1FFF 4000
16 kB boot ROM
0x1FFF 0000
reserved
0x1000 2000
8 kB SRAM (LPC1313/1343)
0x1000 1000
4 kB SRAM (LPC1311/1342)
0x1000 0000
reserved
0x0000 8000
32 kB on-chip flash (LPC1313/43)
0x0000 4000
16 kB on-chip flash (LPC1342)
0x0000 2000
8 kB on-chip flash (LPC1311)
0 GB
0x0000 0000

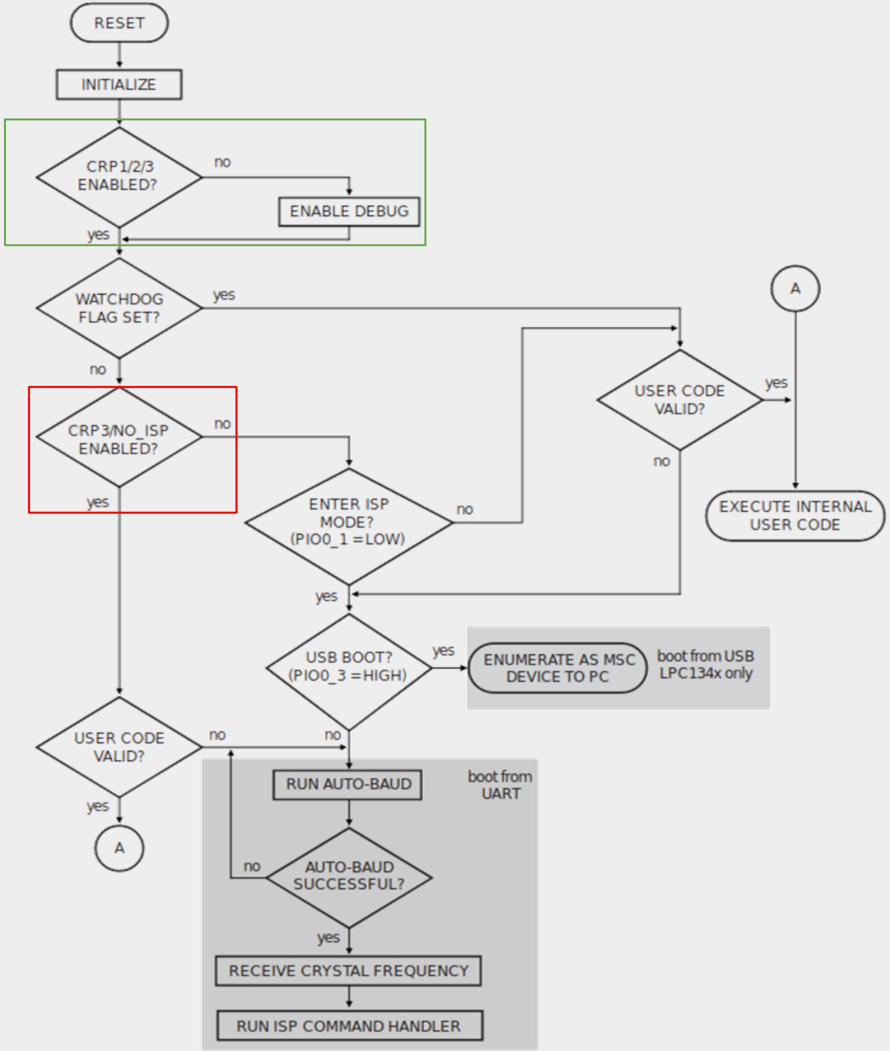
I-code/D-code memory space

+ 256 words
0x0000 0400
active interrupt vectors
0x0000 0000

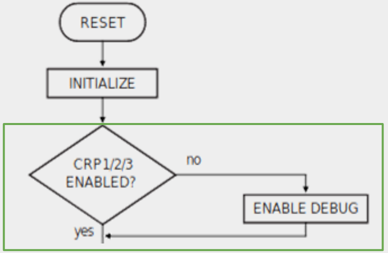
AHB peripherals
0x5020 0000
16 - 127 reserved
0x5004 0000
12-15 GPIO PIO3
0x5003 0000
8-11 GPIO PIO2
0x5002 0000
4-7 GPIO PIO1
0x5001 0000
0-3 GPIO PIO0
0x5000 0000

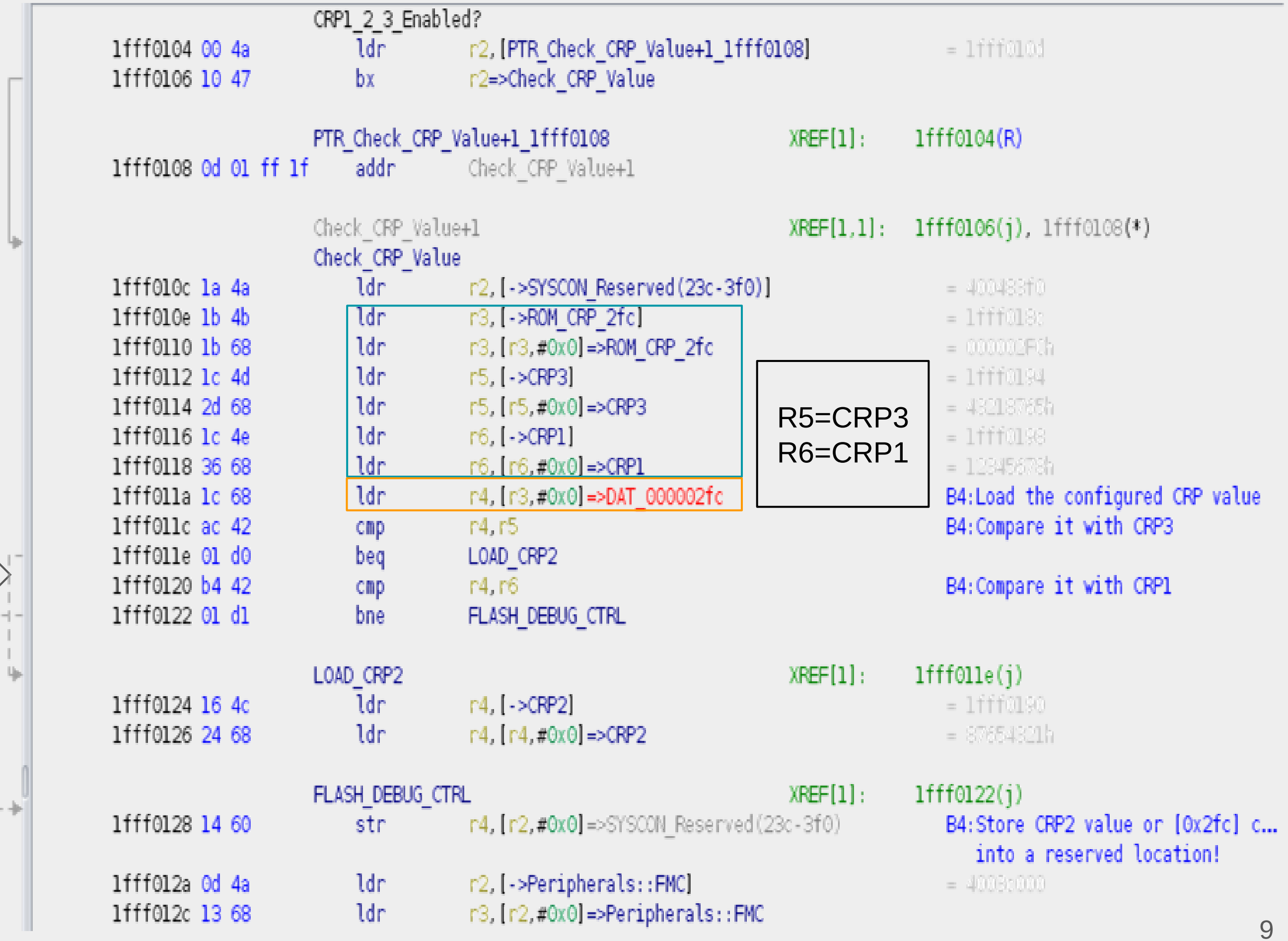
APB peripherals
0x4008 0000
23 - 31 reserved
0x4005 C000
22 SSP1 (LPC1313FBD48/01)
0x4005 8000
19 - 21 reserved
0x4004 C000
18 system control
0x4004 8000
17 IOCONFIG
0x4004 4000
16 SSP0
0x4004 0000
15 flash controller
0x4003 C000
14 PMU
0x4003 8000
10 - 13 reserved
0x4002 8000
9 reserved
0x4002 4000
8 USB (LPC1342/43 only)
0x4002 0000
7 ADC
0x4001 C000
6 32-bit counter/timer 1
0x4001 8000
5 32-bit counter/timer 0
0x4001 4000
4 16-bit counter/timer 1
0x4001 0000
3 16-bit counter/timer 0
0x4000 C000
2 UART
0x4000 8000
1 WDT/WWDT
0x4000 4000
0 I^2C-bus
0x4000 0000

LPC13xx BOOT PROCESS



BOOTLOADER RE





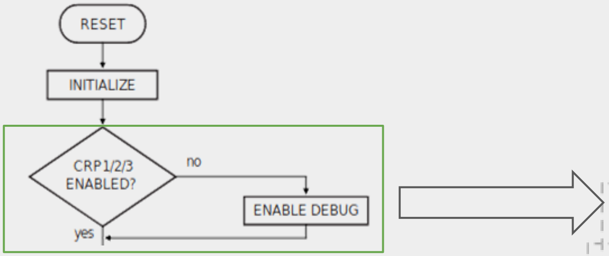
```
                           CRP1_2_3_Enabled?
        1fff0104 00 4a          ldr        r2,[PTR_Check_CRP_Value+1_1fff0108]         = 1fff010d
        1fff0106 10 47          bx         r2=>Check_CRP_Value

                           PTR_Check_CRP_Value+1_1fff0108           XREF[1]:     1fff0104(R)
        1fff0108 0d 01 ff 1f    addr       Check_CRP_Value+1

                           Check_CRP_Value+1                        XREF[1,1]:   1fff0106(j), 1fff0108(*)
                           Check_CRP_Value
        1fff010c 1a 4a          ldr        r2,[->SYSCON_Reserved(23c-3f0)]             = 400483f0
        1fff010e 1b 4b          ldr        r3,[->ROM_CRP_2fc]                          = 1fff018c
        1fff0110 1b 68          ldr        r3,[r3,#0x0]=>ROM_CRP_2fc                   = 000002FCh
        1fff0112 1c 4d          ldr        r5,[->CRP3]                                 = 1fff0194
        1fff0114 2d 68          ldr        r5,[r5,#0x0]=>CRP3                          = 43218765h
        1fff0116 1c 4e          ldr        r6,[->CRP1]                                 = 1fff0198
        1fff0118 36 68          ldr        r6,[r6,#0x0]=>CRP1                          = 12345678h
        1fff011a 1c 68          ldr        r4,[r3,#0x0]=>DAT_000002fc                  B4:Load the configured CRP value
        1fff011c ac 42          cmp        r4,r5                                       B4:Compare it with CRP3
        1fff011e 01 d0          beq        LOAD_CRP2
        1fff0120 b4 42          cmp        r4,r6                                       B4:Compare it with CRP1
        1fff0122 01 d1          bne        FLASH_DEBUG_CTRL

                           LOAD_CRP2                                XREF[1]:     1fff011e(j)
        1fff0124 16 4c          ldr        r4,[->CRP2]                                 = 1fff0190
        1fff0126 24 68          ldr        r4,[r4,#0x0]=>CRP2                          = 87654321h

                           FLASH_DEBUG_CTRL                         XREF[1]:     1fff0122(j)
        1fff0128 14 60          str        r4,[r2,#0x0]=>SYSCON_Reserved(23c-3f0)      B4:Store CRP2 value or [0x2fc] c…
                                                                                         into a reserved location!
        1fff012a 0d 4a          ldr        r2,[->Peripherals::FMC]                     = 4003c000
        1fff012c 13 68          ldr        r3,[r2,#0x0]=>Peripherals::FMC
```

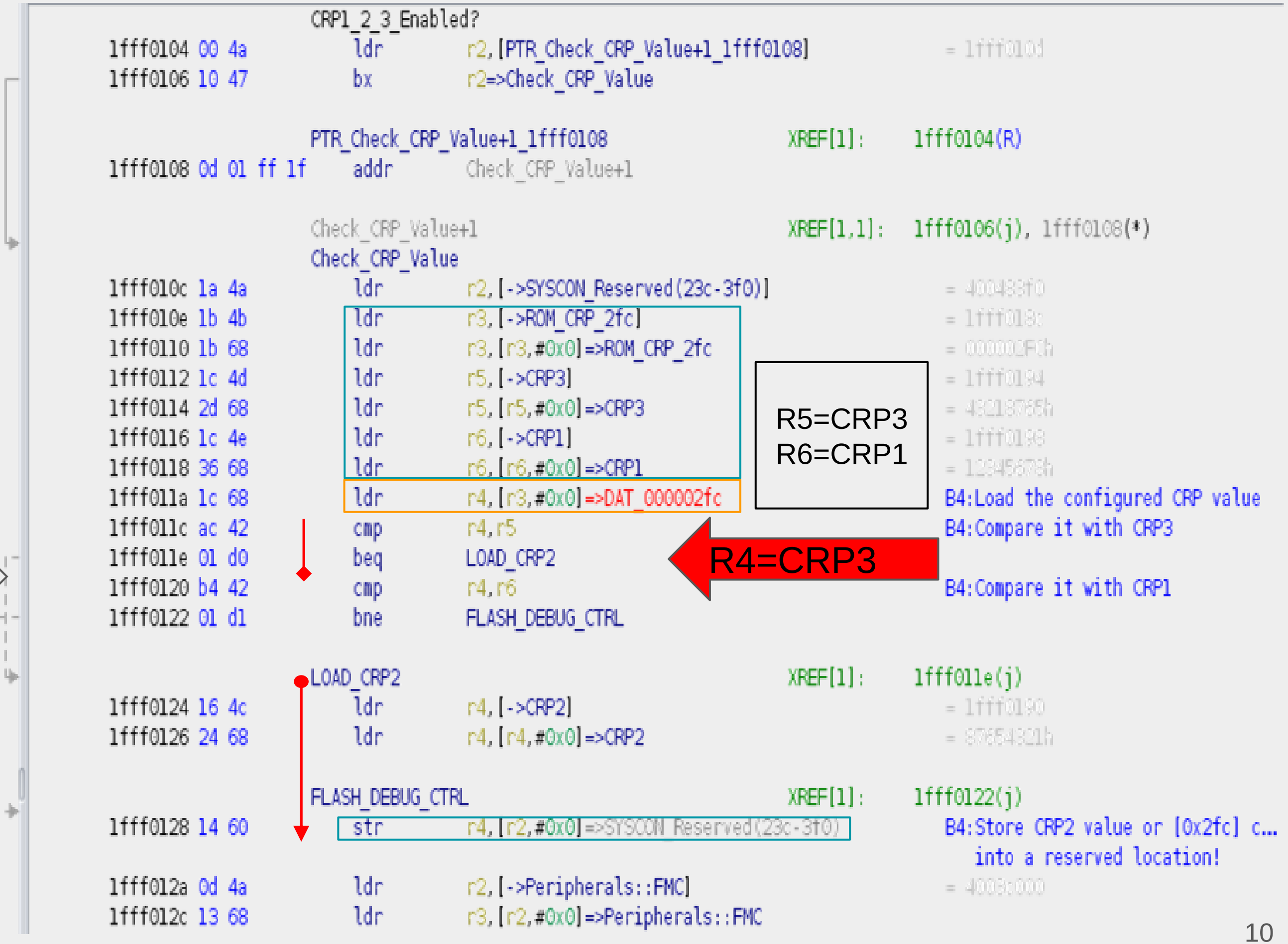
R5=CRP3
R6=CRP1

BOOTLOADER RE

[0x2fc]:

CRP3





```
                    CRP1_2_3_Enabled?
1fff0104 00 4a          ldr         r2,[PTR_Check_CRP_Value+1_1fff0108]                = 1fff010d
1fff0106 10 47          bx          r2=>Check_CRP_Value

                    PTR_Check_CRP_Value+1_1fff0108            XREF[1]:     1fff0104(R)
1fff0108 0d 01 ff 1f    addr        Check_CRP_Value+1

                    Check_CRP_Value+1                         XREF[1,1]:   1fff0106(j), 1fff0108(*)
                    Check_CRP_Value
1fff010c 1a 4a          ldr         r2,[->SYSCON_Reserved(23c-3f0)]                    = 400483f0
1fff010e 1b 4b          ldr         r3,[->ROM_CRP_2fc]                                 = 1fff018c
1fff0110 1b 68          ldr         r3,[r3,#0x0]=>ROM_CRP_2fc                          = 000002FCh
1fff0112 1c 4d          ldr         r5,[->CRP3]                                        = 1fff0194
1fff0114 2d 68          ldr         r5,[r5,#0x0]=>CRP3                                 = 43218765h
1fff0116 1c 4e          ldr         r6,[->CRP1]                                        = 1fff0198
1fff0118 36 68          ldr         r6,[r6,#0x0]=>CRP1                                 = 12345678h
1fff011a 1c 68          ldr         r4,[r3,#0x0]=>DAT_000002fc                         B4:Load the configured CRP value
1fff011c ac 42          cmp         r4,r5                                              B4:Compare it with CRP3
1fff011e 01 d0          beq         LOAD_CRP2
1fff0120 b4 42          cmp         r4,r6                                              B4:Compare it with CRP1
1fff0122 01 d1          bne         FLASH_DEBUG_CTRL

                    LOAD_CRP2                                 XREF[1]:     1fff011e(j)
1fff0124 16 4c          ldr         r4,[->CRP2]                                        = 1fff0190
1fff0126 24 68          ldr         r4,[r4,#0x0]=>CRP2                                 = 87654321h

                    FLASH_DEBUG_CTRL                          XREF[1]:     1fff0122(j)
1fff0128 14 60          str         r4,[r2,#0x0]=>SYSCON_Reserved(23c-3f0)             B4:Store CRP2 value or [0x2fc] c...
                                                                                         into a reserved location!
1fff012a 0d 4a          ldr         r2,[->Peripherals::FMC]                            = 4003c000
1fff012c 13 68          ldr         r3,[r2,#0x0]=>Peripherals::FMC
```

R5=CRP3
R6=CRP1

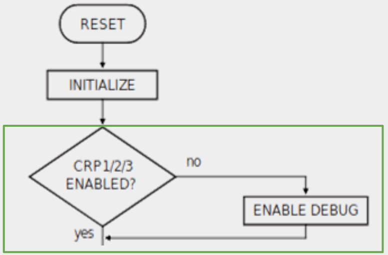
R4=CRP3

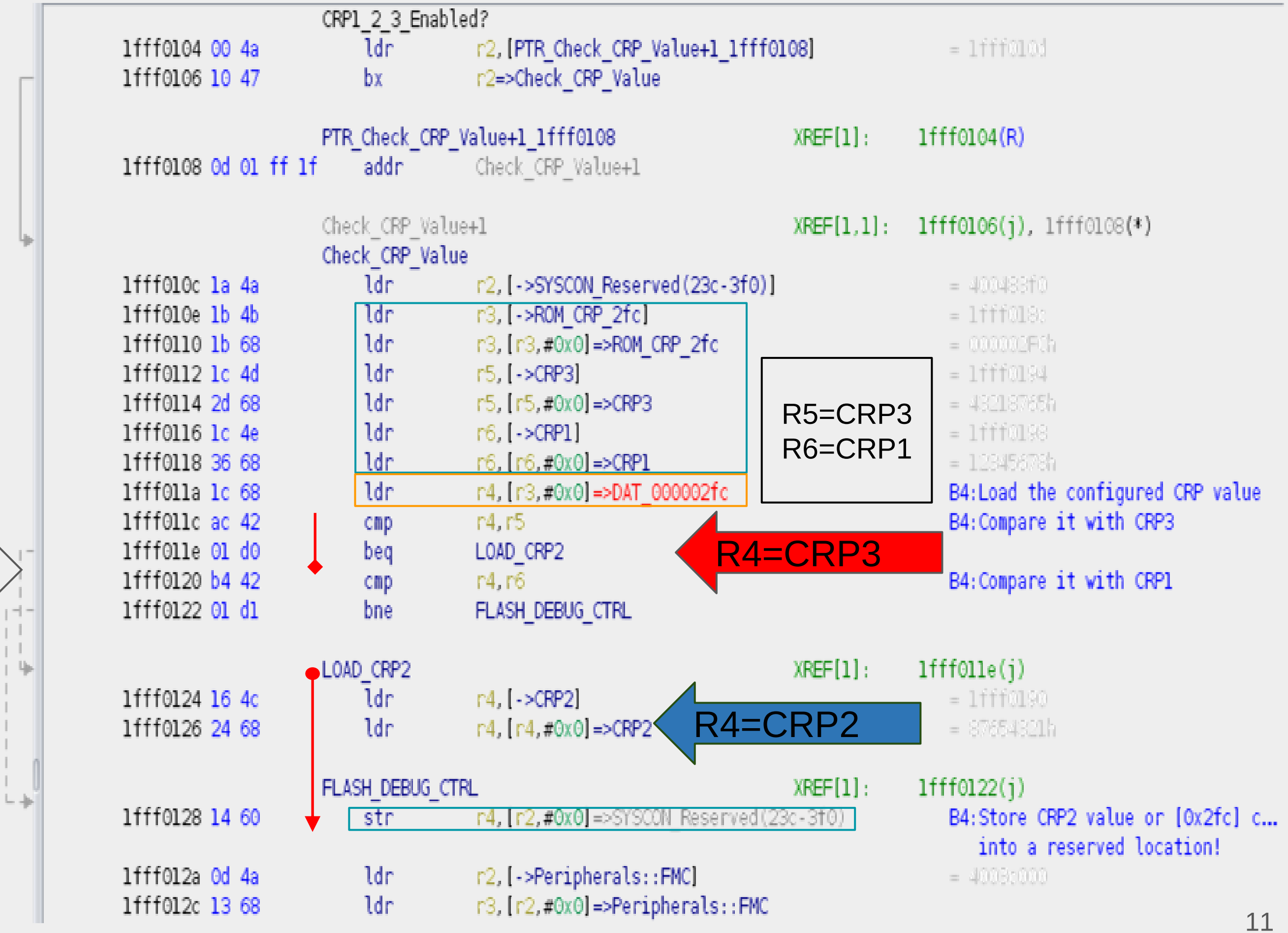
*R4=**0x87654321=CRP2**

BOOTLOADER RE

[0x2fc]:

CRP3





```
                      CRP1_2_3_Enabled?
1fff0104 00 4a          ldr        r2,[PTR_Check_CRP_Value+1_1fff0108]          = 1fff010d
1fff0106 10 47          bx         r2=>Check_CRP_Value

                      PTR_Check_CRP_Value+1_1fff0108              XREF[1]:     1fff0104(R)
1fff0108 0d 01 ff 1f    addr       Check_CRP_Value+1

                      Check_CRP_Value+1                           XREF[1,1]:   1fff0106(j), 1fff0108(*)
                      Check_CRP_Value
1fff010c 1a 4a          ldr        r2,[->SYSCON_Reserved(23c-3f0)]              = 400483f0
1fff010e 1b 4b          ldr        r3,[->ROM_CRP_2fc]                           = 1fff018c
1fff0110 1b 68          ldr        r3,[r3,#0x0]=>ROM_CRP_2fc                    = 000002FCh
1fff0112 1c 4d          ldr        r5,[->CRP3]                                  = 1fff0194
1fff0114 2d 68          ldr        r5,[r5,#0x0]=>CRP3                           = 43218765h
1fff0116 1c 4e          ldr        r6,[->CRP1]                                  = 1fff0198
1fff0118 36 68          ldr        r6,[r6,#0x0]=>CRP1                           = 12345678h
1fff011a 1c 68          ldr        r4,[r3,#0x0]=>DAT_000002fc                   B4:Load the configured CRP value
1fff011c ac 42          cmp        r4,r5                                        B4:Compare it with CRP3
1fff011e 01 d0          beq        LOAD_CRP2
1fff0120 b4 42          cmp        r4,r6                                        B4:Compare it with CRP1
1fff0122 01 d1          bne        FLASH_DEBUG_CTRL

                      LOAD_CRP2                                   XREF[1]:     1fff011e(j)
1fff0124 16 4c          ldr        r4,[->CRP2]                                  = 1fff0190
1fff0126 24 68          ldr        r4,[r4,#0x0]=>CRP2                           = 87654321h

                      FLASH_DEBUG_CTRL                            XREF[1]:     1fff0122(j)
1fff0128 14 60          str        r4,[r2,#0x0]=>SYSCON_Reserved(23c-3f0)       B4:Store CRP2 value or [0x2fc] c…
                                                                                  into a reserved location!
1fff012a 0d 4a          ldr        r2,[->Peripherals::FMC]                      = 4003c000
1fff012c 13 68          ldr        r3,[r2,#0x0]=>Peripherals::FMC
```

R5=CRP3
R6=CRP1

R4=CRP3

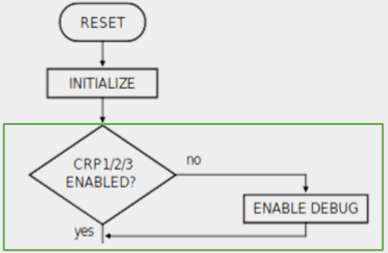
R4=CRP2

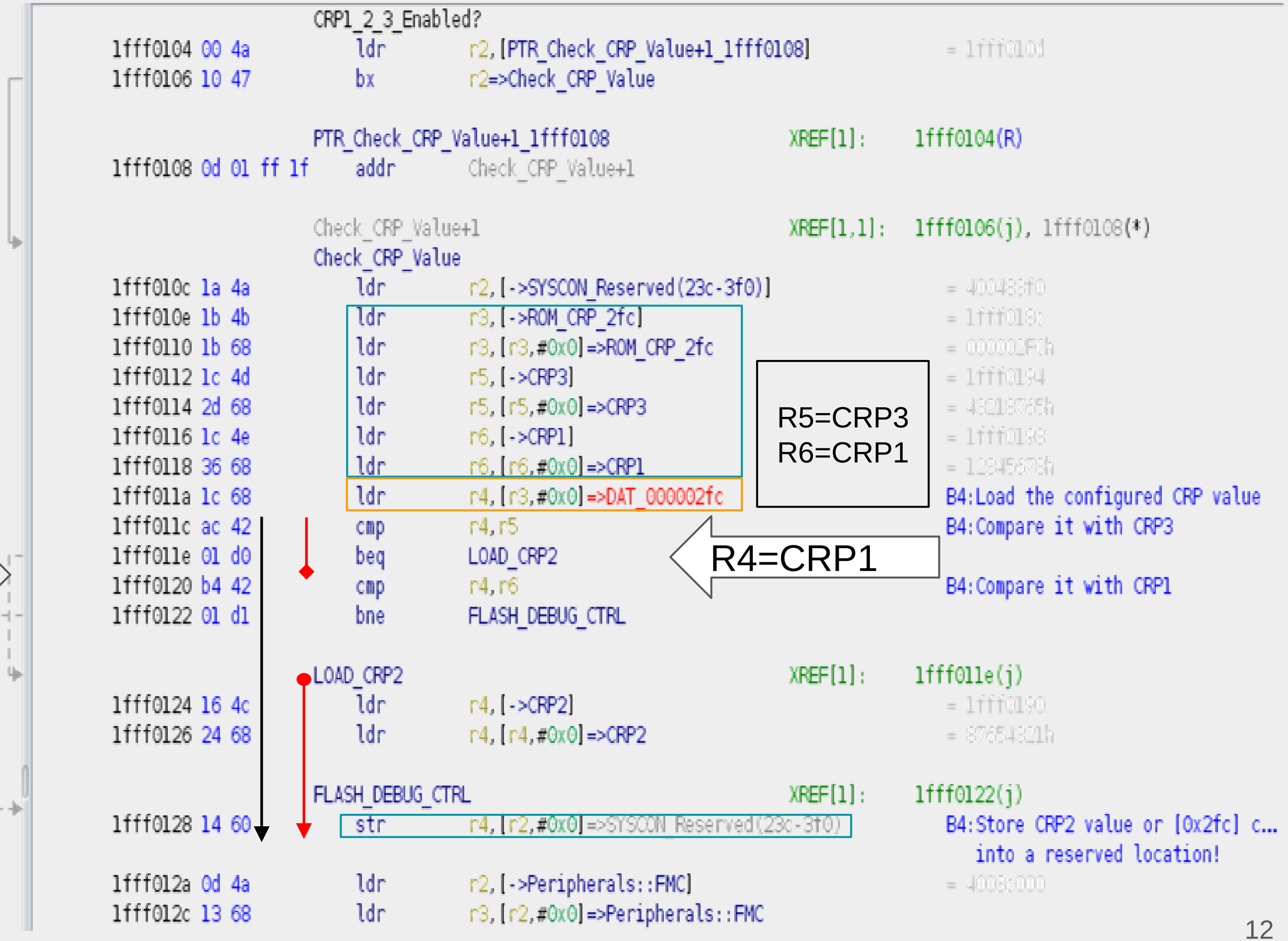
*R4=**0x87654321=CRP2**

BOOTLOADER RE

[0x2fc]:
CRP1

CRP3





```
                       CRP1_2_3_Enabled?
1fff0104 00 4a          ldr         r2,[PTR_Check_CRP_Value+1_1fff0108]         = 1fff010d
1fff0106 10 47          bx          r2=>Check_CRP_Value

                       PTR_Check_CRP_Value+1_1fff0108           XREF[1]:     1fff0104(R)
1fff0108 0d 01 ff 1f    addr        Check_CRP_Value+1

                       Check_CRP_Value+1                        XREF[1,1]:   1fff0106(j), 1fff0108(*)
                       Check_CRP_Value
1fff010c 1a 4a          ldr         r2,[->SYSCON_Reserved(23c-3f0)]             = 400483f0
1fff010e 1b 4b          ldr         r3,[->ROM_CRP_2fc]                          = 1fff018c
1fff0110 1b 68          ldr         r3,[r3,#0x0]=>ROM_CRP_2fc                   = 000002FCh
1fff0112 1c 4d          ldr         r5,[->CRP3]                                 = 1fff0194
1fff0114 2d 68          ldr         r5,[r5,#0x0]=>CRP3                          = 43218765h
1fff0116 1c 4e          ldr         r6,[->CRP1]                                 = 1fff0198
1fff0118 36 68          ldr         r6,[r6,#0x0]=>CRP1                          = 12345678h
1fff011a 1c 68          ldr         r4,[r3,#0x0]=>DAT_000002fc                  B4:Load the configured CRP value
1fff011c ac 42          cmp         r4,r5                                       B4:Compare it with CRP3
1fff011e 01 d0          beq         LOAD_CRP2
1fff0120 b4 42          cmp         r4,r6                                       B4:Compare it with CRP1
1fff0122 01 d1          bne         FLASH_DEBUG_CTRL

                       LOAD_CRP2                                XREF[1]:     1fff011e(j)
1fff0124 16 4c          ldr         r4,[->CRP2]                                 = 1fff0190
1fff0126 24 68          ldr         r4,[r4,#0x0]=>CRP2                          = 87654321h

                       FLASH_DEBUG_CTRL                         XREF[1]:     1fff0122(j)
1fff0128 14 60          str         r4,[r2,#0x0]=>SYSCON_Reserved(23c-3f0)      B4:Store CRP2 value or [0x2fc] c…
                                                                                  into a reserved location!
1fff012a 0d 4a          ldr         r2,[->Peripherals::FMC]                     = 4003c000
1fff012c 13 68          ldr         r3,[r2,#0x0]=>Peripherals::FMC
```

R5=CRP3
R6=CRP1

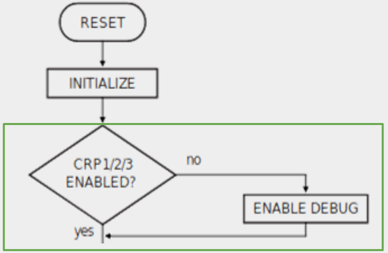
R4=CRP1

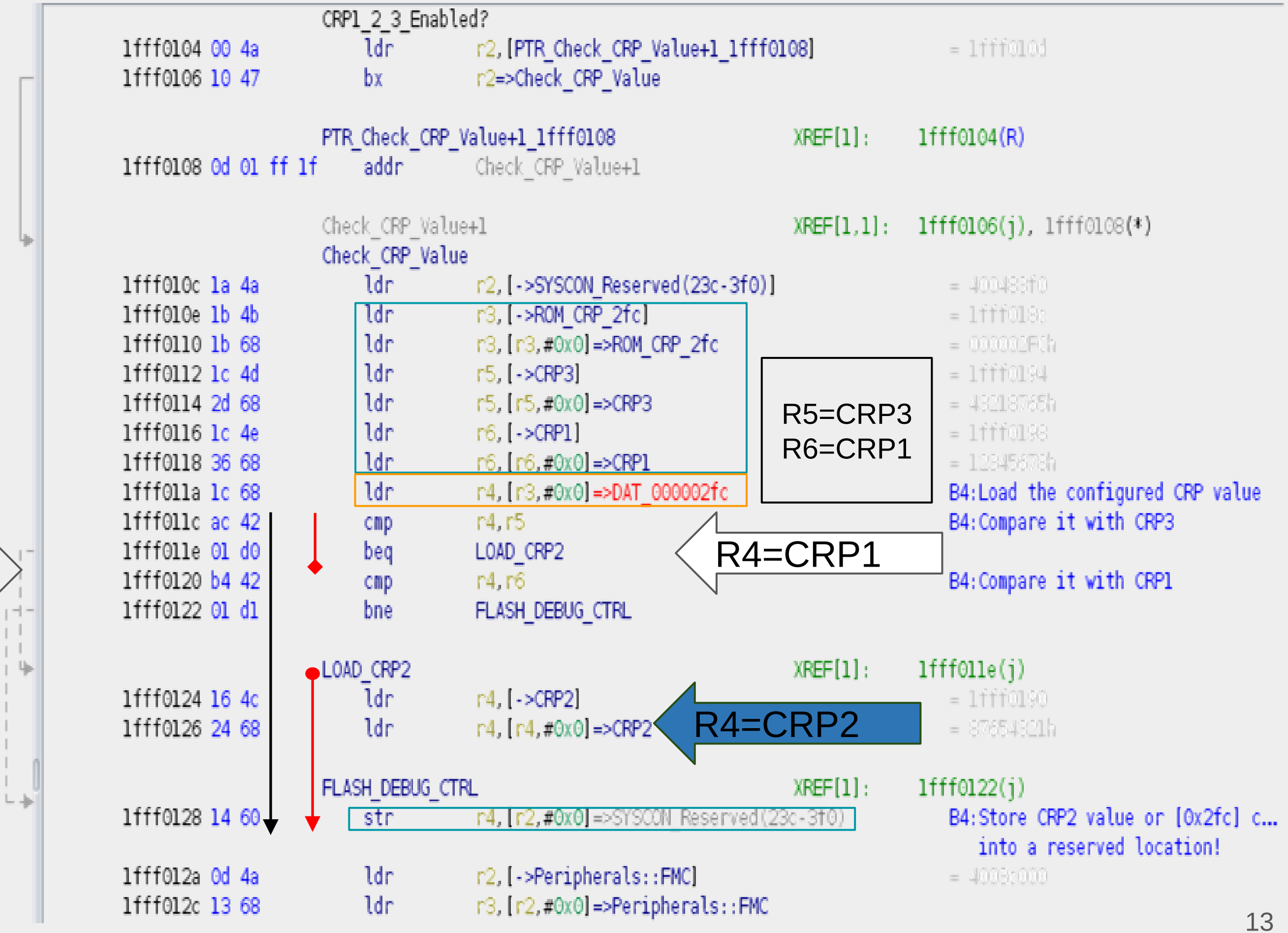
*R4=0x87654321=CRP2

BOOTLOADER RE

[0x2fc]:
CRP1

CRP3





```
                           CRP1_2_3_Enabled?
    1fff0104 00 4a             ldr        r2,[PTR_Check_CRP_Value+1_1fff0108]          = 1fff010d
    1fff0106 10 47             bx         r2=>Check_CRP_Value

                           PTR_Check_CRP_Value+1_1fff0108          XREF[1]:     1fff0104(R)
    1fff0108 0d 01 ff 1f       addr       Check_CRP_Value+1

                           Check_CRP_Value+1                       XREF[1,1]:   1fff0106(j), 1fff0108(*)
                           Check_CRP_Value
    1fff010c 1a 4a             ldr        r2,[->SYSCON_Reserved(23c-3f0)]              = 400483f0
    1fff010e 1b 4b             ldr        r3,[->ROM_CRP_2fc]                           = 1fff018c
    1fff0110 1b 68             ldr        r3,[r3,#0x0]=>ROM_CRP_2fc                    = 000002FCh
    1fff0112 1c 4d             ldr        r5,[->CRP3]                                  = 1fff0194
    1fff0114 2d 68             ldr        r5,[r5,#0x0]=>CRP3                           = 43218765h
    1fff0116 1c 4e             ldr        r6,[->CRP1]                                  = 1fff0198
    1fff0118 36 68             ldr        r6,[r6,#0x0]=>CRP1                           = 12345678h
    1fff011a 1c 68             ldr        r4,[r3,#0x0]=>DAT_000002fc                   B4:Load the configured CRP value
    1fff011c ac 42             cmp        r4,r5                                        B4:Compare it with CRP3
    1fff011e 01 d0             beq        LOAD_CRP2
    1fff0120 b4 42             cmp        r4,r6                                        B4:Compare it with CRP1
    1fff0122 01 d1             bne        FLASH_DEBUG_CTRL

                           LOAD_CRP2                               XREF[1]:     1fff011e(j)
    1fff0124 16 4c             ldr        r4,[->CRP2]                                  = 1fff0190
    1fff0126 24 68             ldr        r4,[r4,#0x0]=>CRP2                           = 87654321h

                           FLASH_DEBUG_CTRL                        XREF[1]:     1fff0122(j)
    1fff0128 14 60             str        r4,[r2,#0x0]=>SYSCON_Reserved(23c-3f0)       B4:Store CRP2 value or [0x2fc] c…
                                                                                         into a reserved location!
    1fff012a 0d 4a             ldr        r2,[->Peripherals::FMC]                      = 4003c000
    1fff012c 13 68             ldr        r3,[r2,#0x0]=>Peripherals::FMC
```

R5=CRP3
R6=CRP1

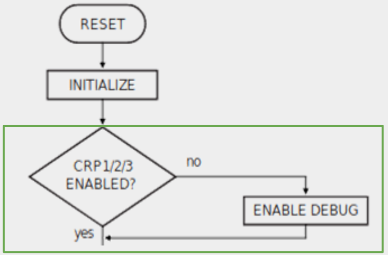
R4=CRP1

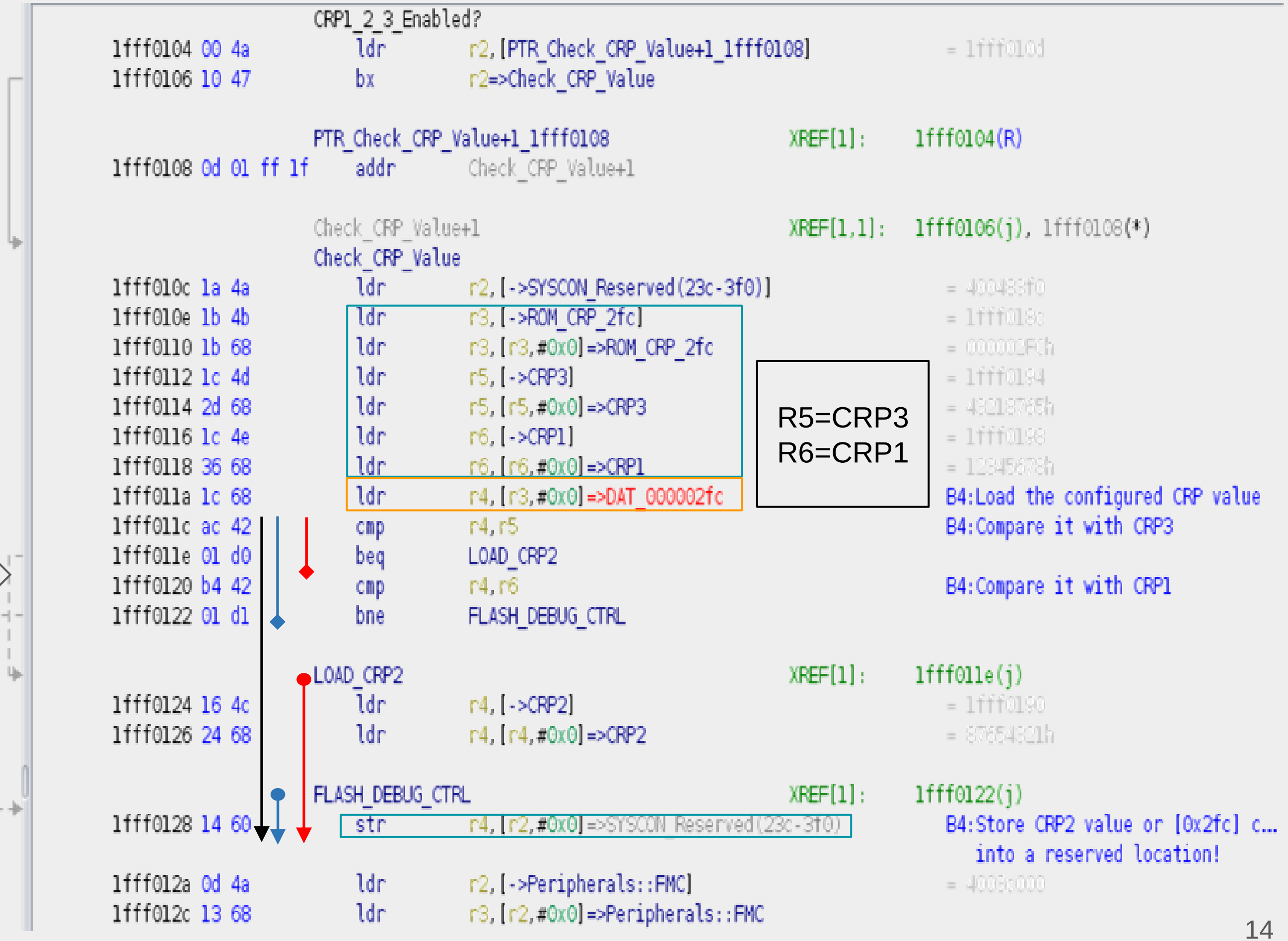
R4=CRP2

*R4=0x87654321=CRP2

BOOTLOADER RE

[0x2fc]:
CRP1
CRP2/NO_ISP/NO_CRP…
CRP3



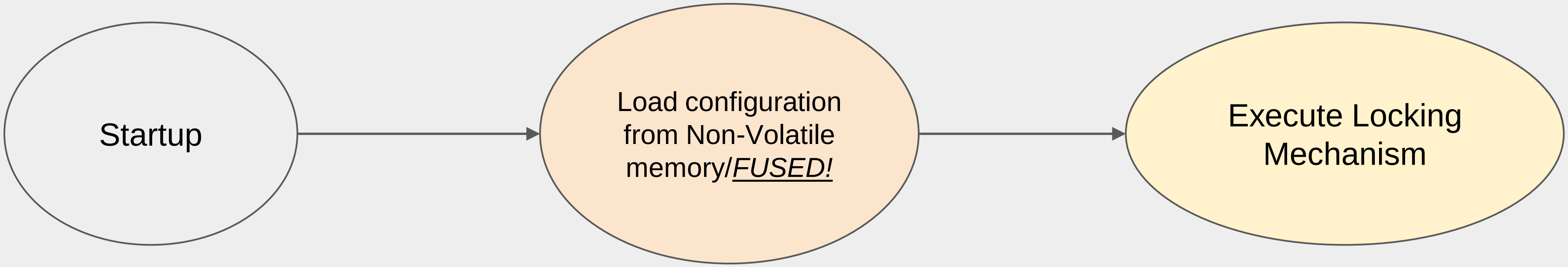


```
                        CRP1_2_3_Enabled?
  1fff0104 00 4a          ldr        r2,[PTR_Check_CRP_Value+1_1fff0108]             = 1fff010d
  1fff0106 10 47          bx         r2=>Check_CRP_Value

                        PTR_Check_CRP_Value+1_1fff0108              XREF[1]:     1fff0104(R)
  1fff0108 0d 01 ff 1f    addr       Check_CRP_Value+1

                        Check_CRP_Value+1                           XREF[1,1]:   1fff0106(j), 1fff0108(*)
                        Check_CRP_Value
  1fff010c 1a 4a          ldr        r2,[->SYSCON_Reserved(23c-3f0)]                 = 400483f0
  1fff010e 1b 4b          ldr        r3,[->ROM_CRP_2fc]                              = 1fff018c
  1fff0110 1b 68          ldr        r3,[r3,#0x0]=>ROM_CRP_2fc                       = 000002FCh
  1fff0112 1c 4d          ldr        r5,[->CRP3]                                     = 1fff0194
  1fff0114 2d 68          ldr        r5,[r5,#0x0]=>CRP3                              = 43218765h
  1fff0116 1c 4e          ldr        r6,[->CRP1]                                     = 1fff0198
  1fff0118 36 68          ldr        r6,[r6,#0x0]=>CRP1                              = 12345678h
  1fff011a 1c 68          ldr        r4,[r3,#0x0]=>DAT_000002fc                      B4:Load the configured CRP value
  1fff011c ac 42          cmp        r4,r5                                           B4:Compare it with CRP3
  1fff011e 01 d0          beq        LOAD_CRP2
  1fff0120 b4 42          cmp        r4,r6                                           B4:Compare it with CRP1
  1fff0122 01 d1          bne        FLASH_DEBUG_CTRL

                        LOAD_CRP2                                   XREF[1]:     1fff011e(j)
  1fff0124 16 4c          ldr        r4,[->CRP2]                                     = 1fff0190
  1fff0126 24 68          ldr        r4,[r4,#0x0]=>CRP2                              = 87654321h

                        FLASH_DEBUG_CTRL                            XREF[1]:     1fff0122(j)
  1fff0128 14 60          str        r4,[r2,#0x0]=>SYSCON_Reserved(23c-3f0)          B4:Store CRP2 value or [0x2fc] c…
                                                                                       into a reserved location!
  1fff012a 0d 4a          ldr        r2,[->Peripherals::FMC]                         = 4003c000
  1fff012c 13 68          ldr        r3,[r2,#0x0]=>Peripherals::FMC
```

R5=CRP3
R6=CRP1

<u>*The value that disables JTAG:0x87654321=CRP2</u>

BOOTLOADER VULNERABILITY



Startup

Load configuration from Non-Volatile memory/*FUSED!*

Execute Locking Mechanism

Corrupting the configuration values
Register corruption
Bypass the load
Bypass/Corrupt compare instruction
…

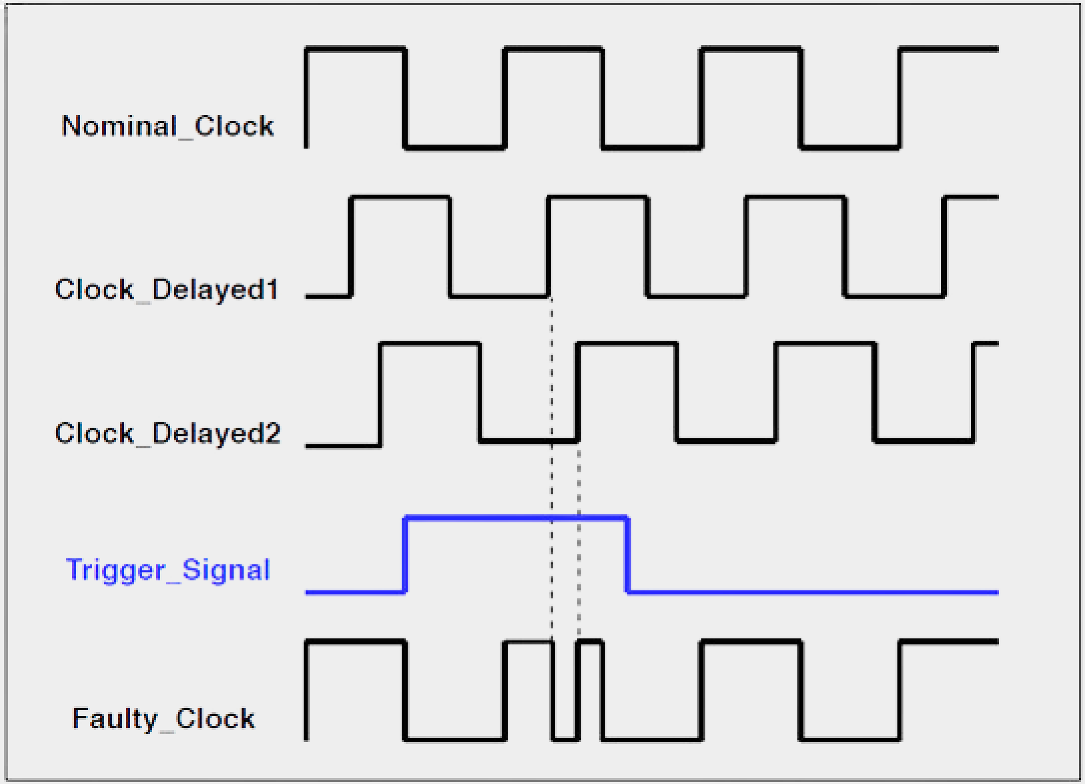
CRP1	0x12345678
CRP2	0x87654321
CRP3	0x43218765
NO_ISP	0x4E697370
NO_CRP	0xFFFFFFFF 0x000000000 0xB4B4B4B4….

WHAT IS FAULT INJECTION (GLITCHING)?

- Hardware corruption on a normal device
- Causes undefined behavior
- Examples [bit flip, instruction skip, change an instruction, …etc]

TECHNIQUES

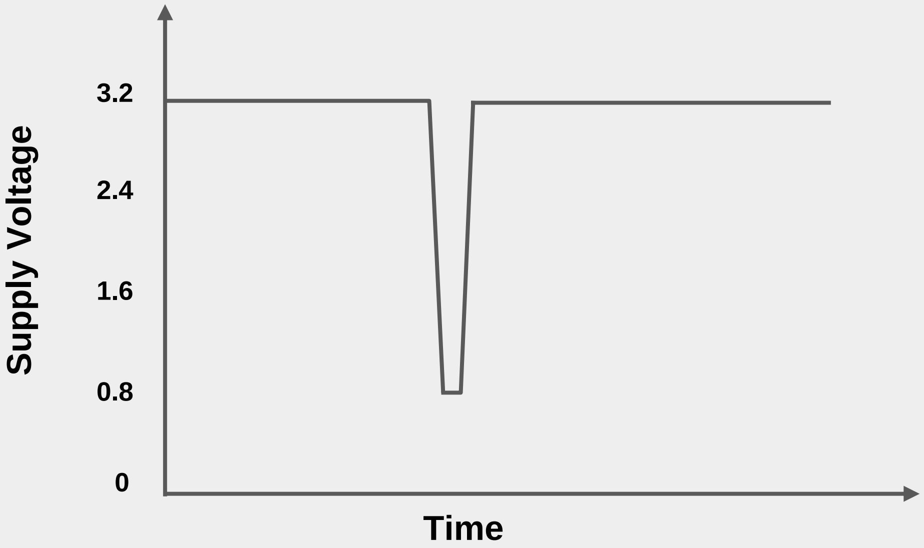
- **Clock**
- Voltage
- Electromagnetic
- Laser
- And more…



https://www.mdpi.com/2079-9292/9/7/1153/htm

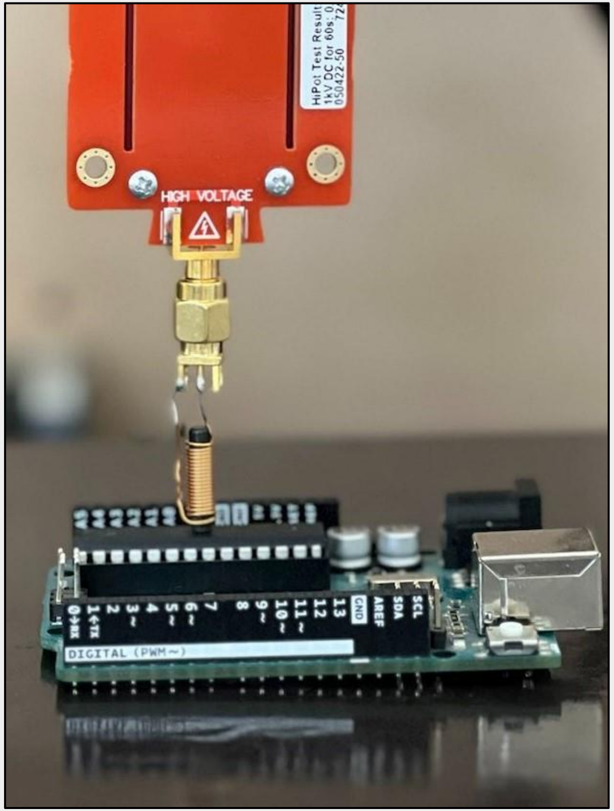
TECHNIQUES

- Clock
- **Voltage**
- Electromagnetic
- Laser
- And more…



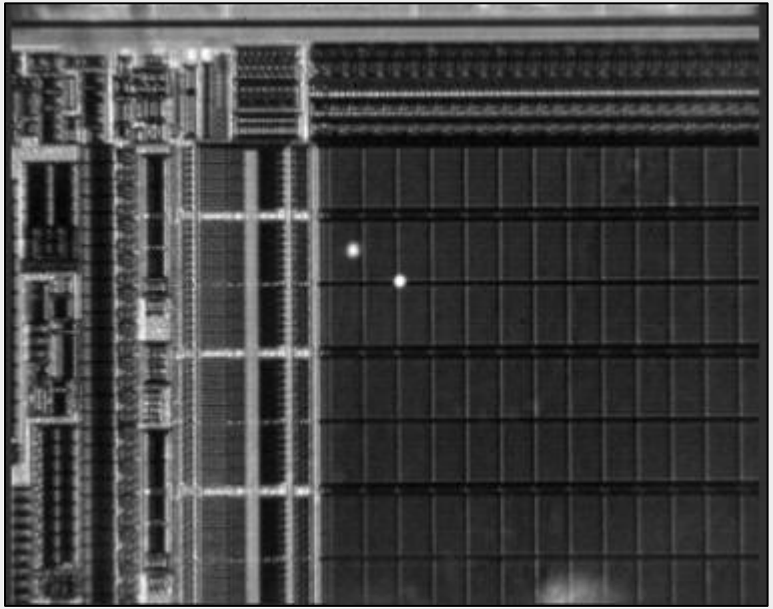
TECHNIQUES

- Clock
- Voltage
- **Electromagnetic**
- Laser
- And more…



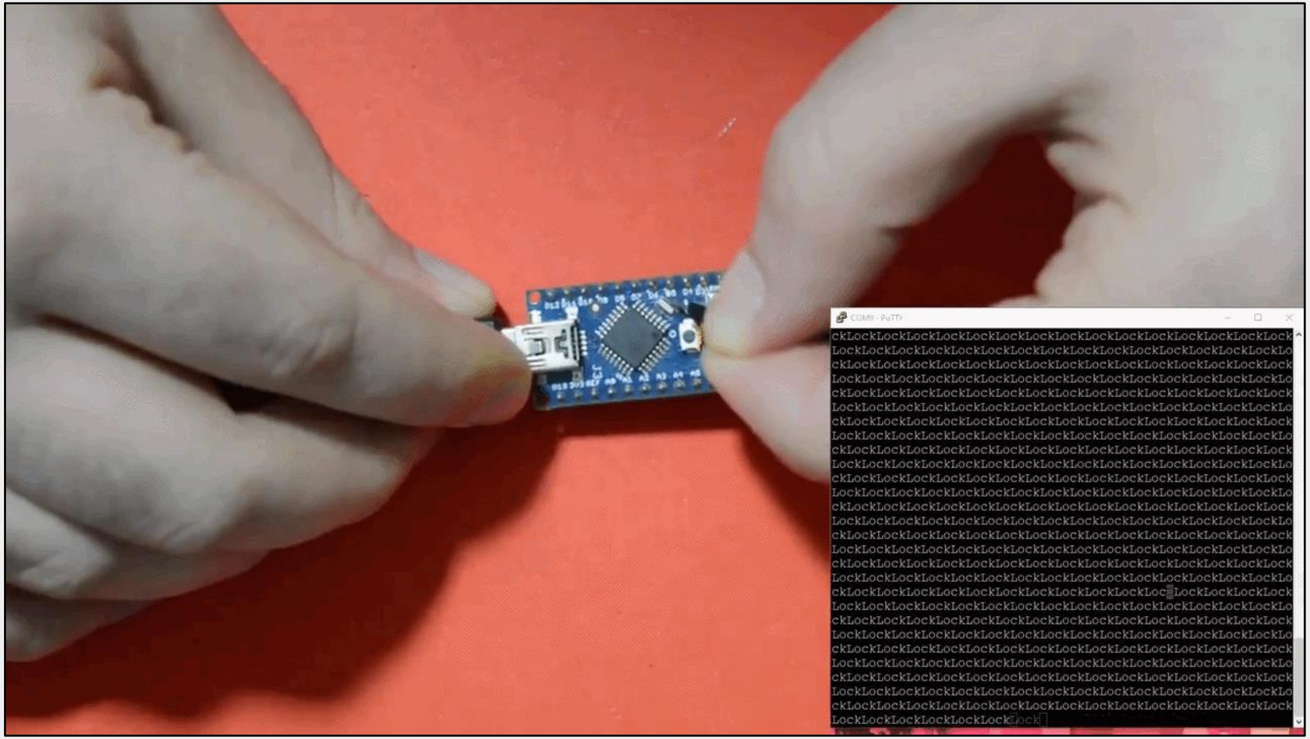
TECHNIQUES

- Clock
- Voltage
- Electromagnetic
- **Laser**
- And more…



https://www.alphanov.com/en/products-services/double-laser-fault-injection

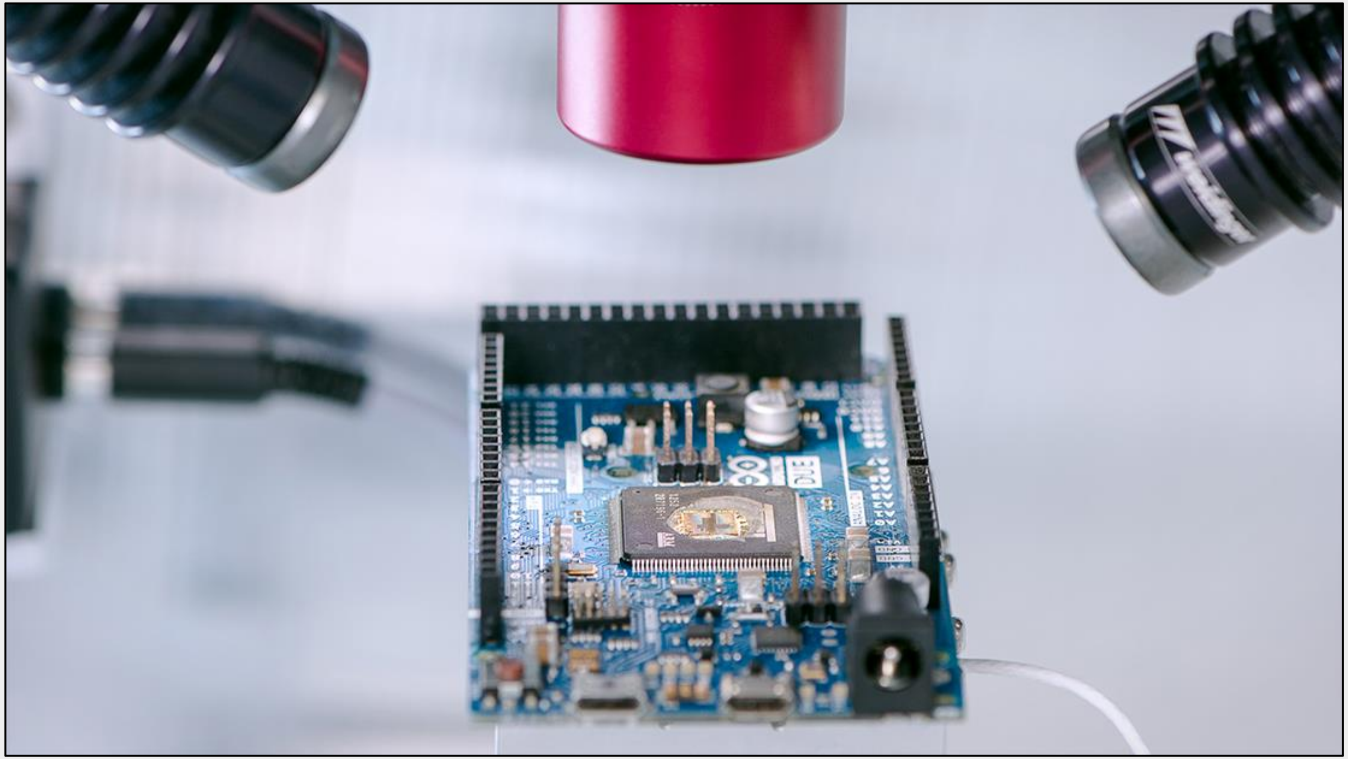
IS IT DIFFICULT?



```
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLoc▒LockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLockLockLockLockLockLockLockLockLockLo
ckLockLockLockLockLockLockLockLockLockLockLockLockLockLockLock
LockLockLockLockLockLockLock
```

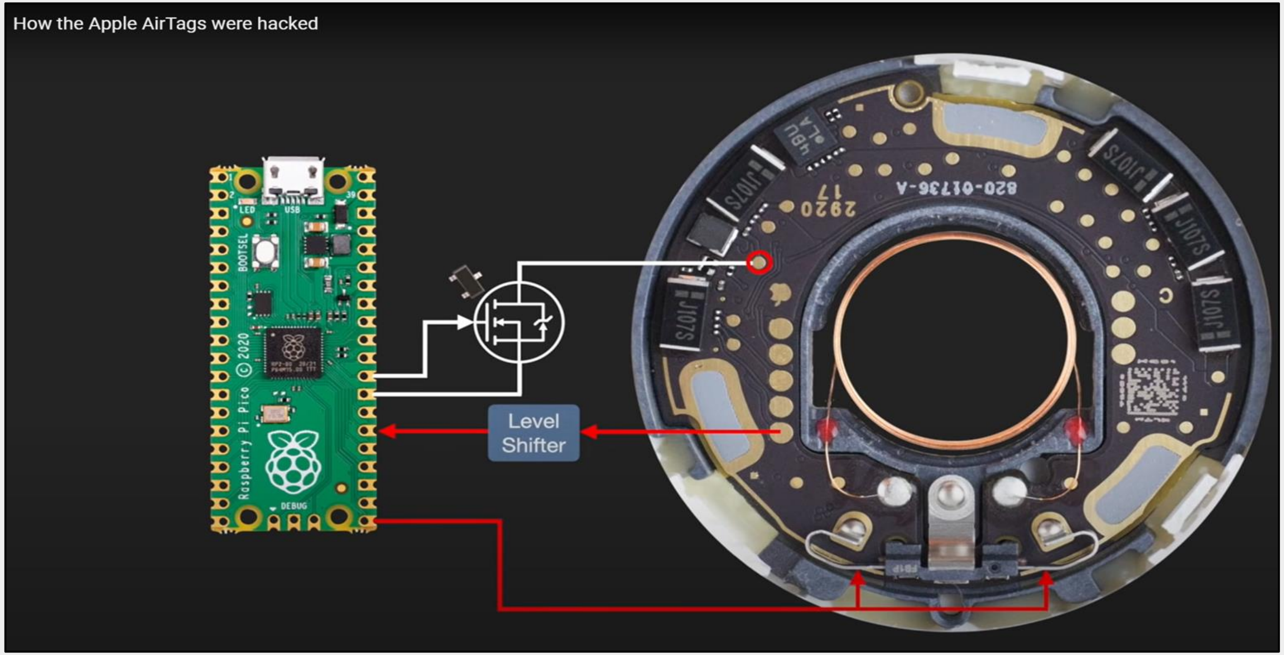
Julio Della Flora @jcldf
https://twitter.com/jcldf/status/1235859271176171521

IS IT DIFFICULT?



https://www.fraunhofer-innovisions.de/cybersicherheit/laser-fault-injection/

EXAMPLES



How the Apple AirTags were hacked

https://www.youtube.com/watch?v=_E0PWQvW-14

EXAMPLES

Bypassing Android MDM Using Electromagnetic Fault Injection
By A Gas Lighter For $1.5 (Arun 24-September-2020)

EXAMPLES

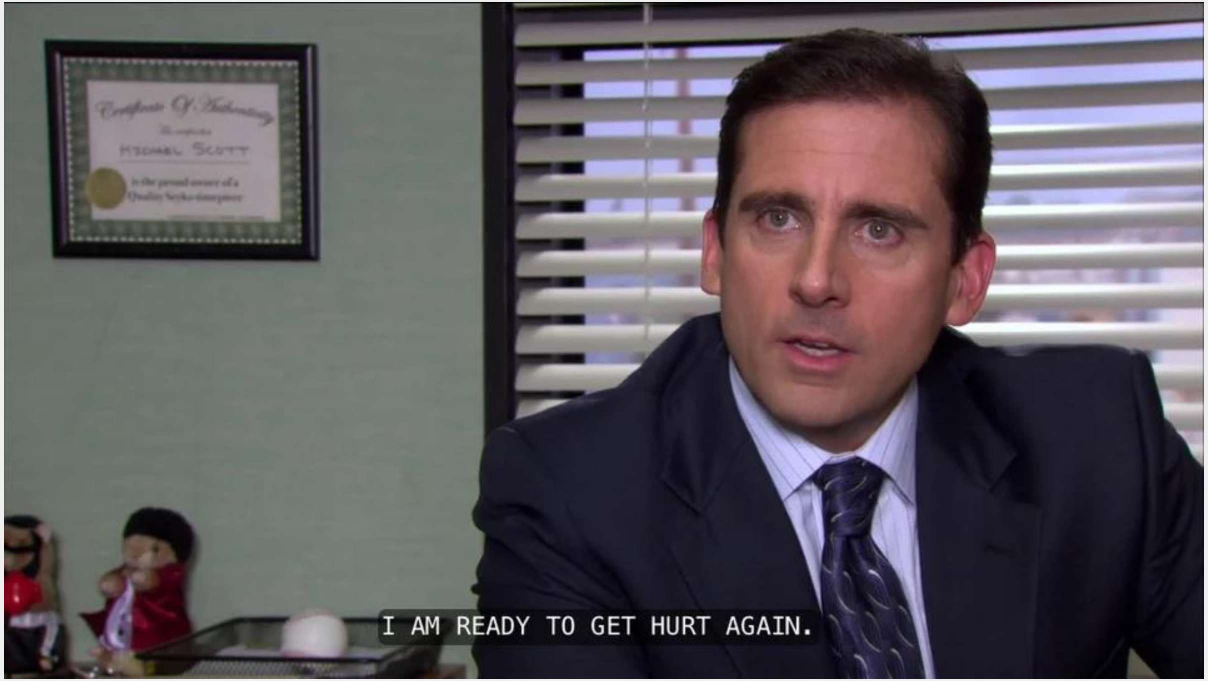
How I hacked a hardware crypto wallet and recovered $2 million (Joe Grand)
Wallet.fail (Thomas, Dmitry, Josh)
Kraken Security Labs (Nick)
Glitching Trezor using EMFI Through The Enclosure (Colin O’flynn)

BEFORE THE ATTACK

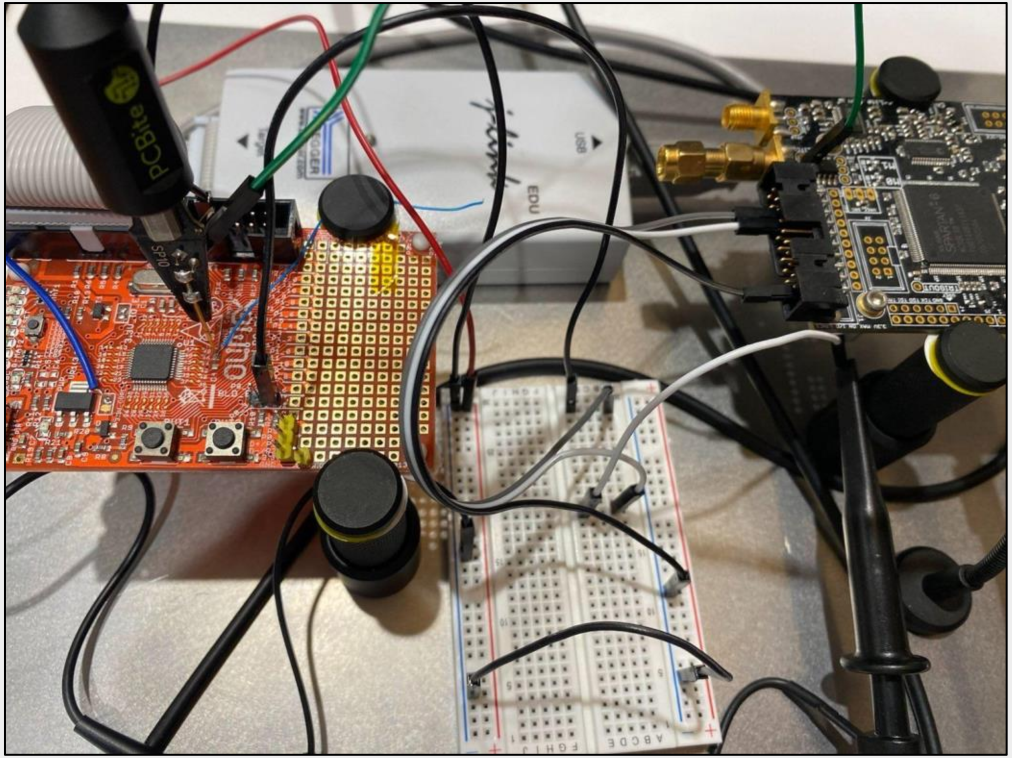
- Inspired by Recon Brussels 2017 talk, by Chris Gerlinsky.

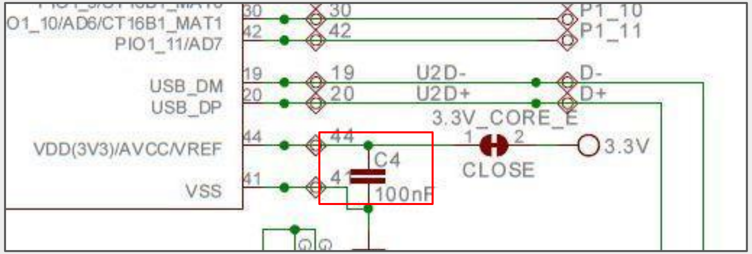
BEFORE THE ATTACK

- Inspired by Recon Brussels 2017 talk, by Chris Gerlinsky.

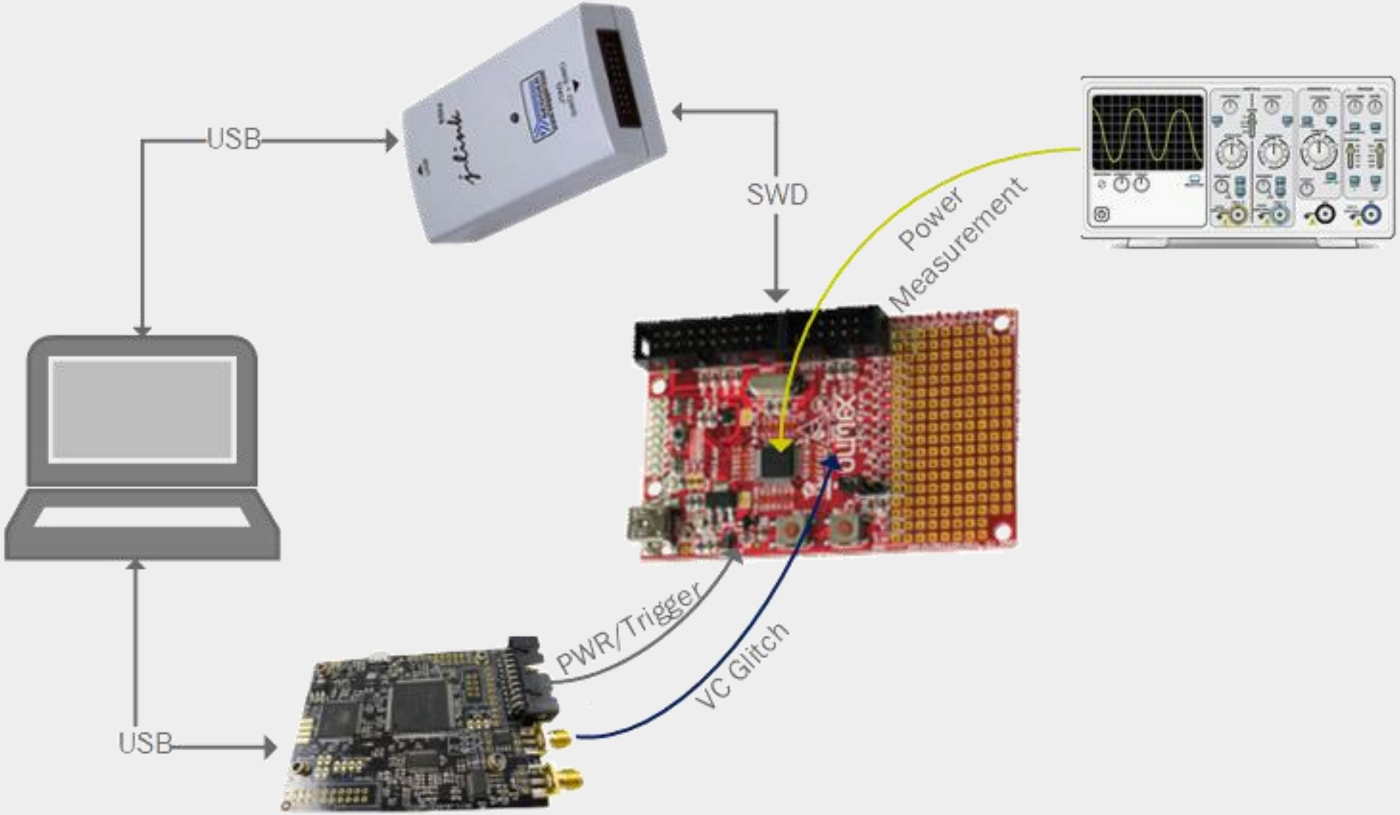


ATTACK SETUP

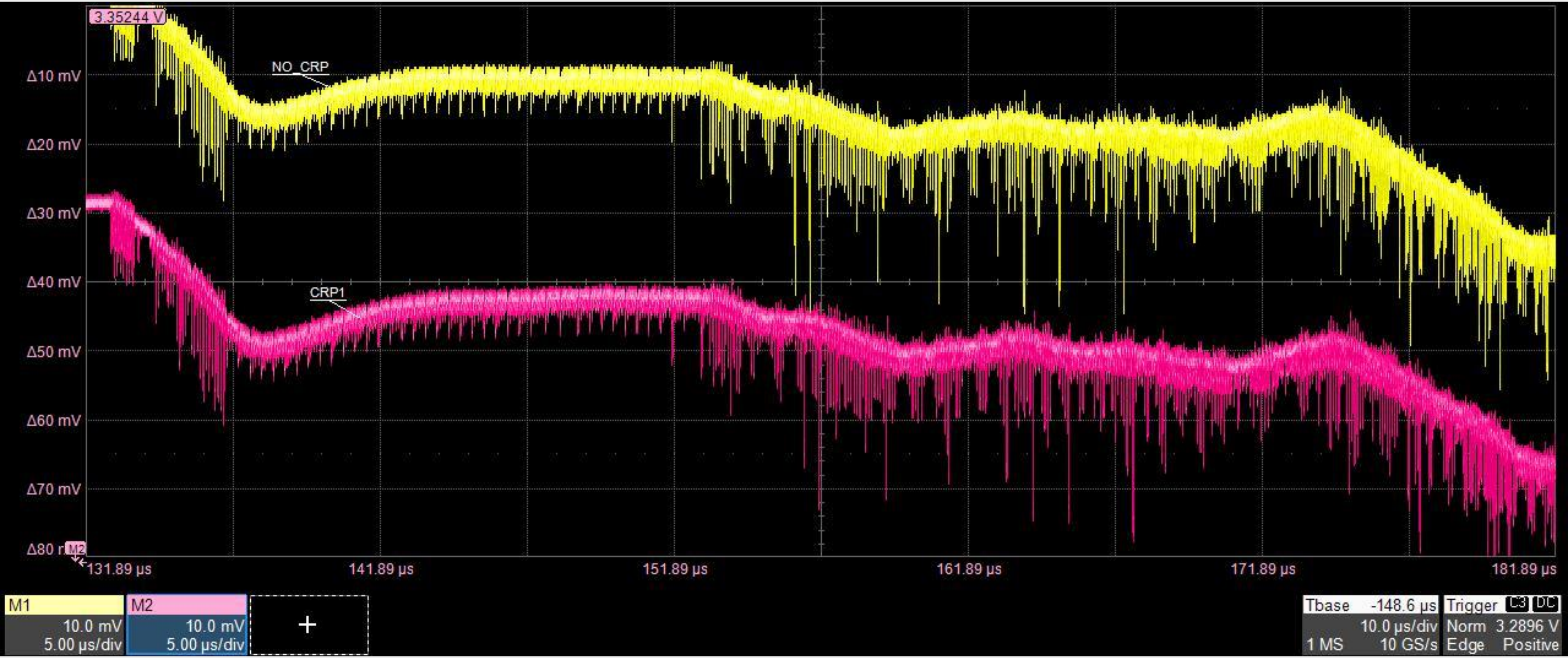




ATTACK SETUP



POWER ANALYSIS



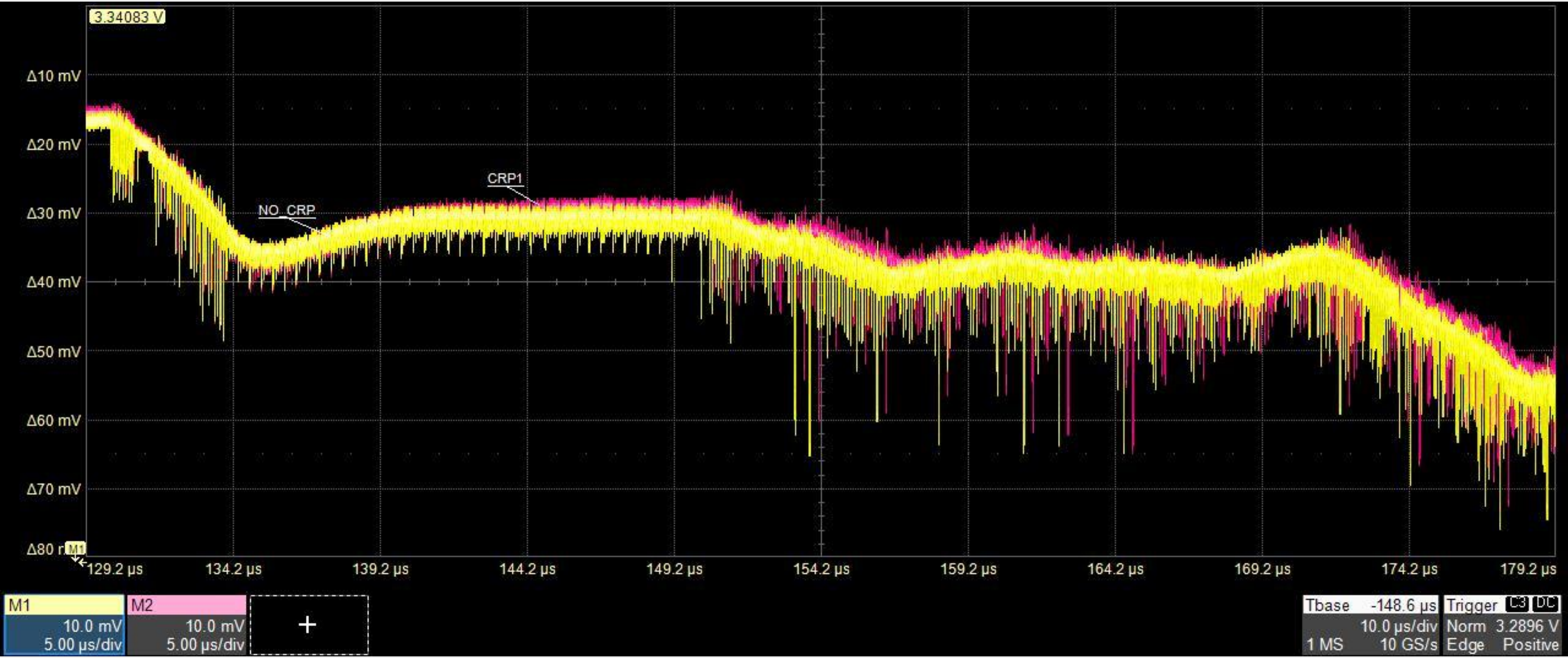
M1 10.0 mV 5.00 µs/div

M2 10.0 mV 5.00 µs/div

Tbase -148.6 µs 10.0 µs/div 1 MS 10 GS/s

Trigger C3 DC Norm 3.2896 V Edge Positive

POWER ANALYSIS



M1 10.0 mV 5.00 µs/div

M2 10.0 mV 5.00 µs/div

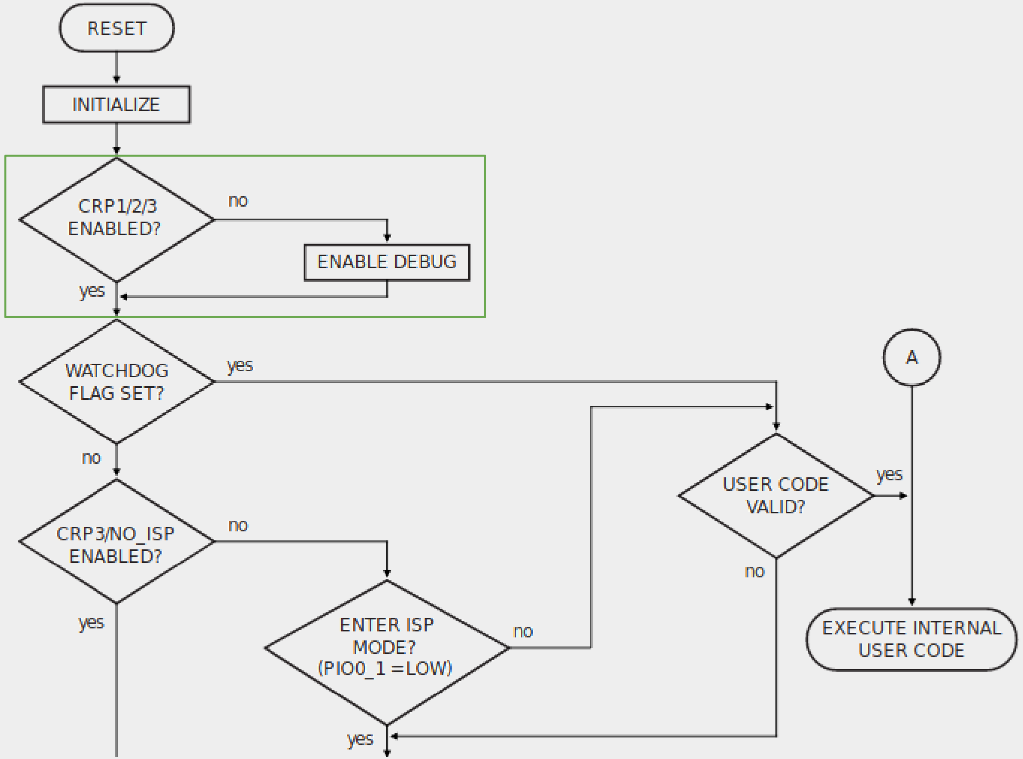
Tbase -148.6 µs 10.0 µs/div 1 MS 10 GS/s

Trigger C3 DC Norm 3.2896 V Edge Positive

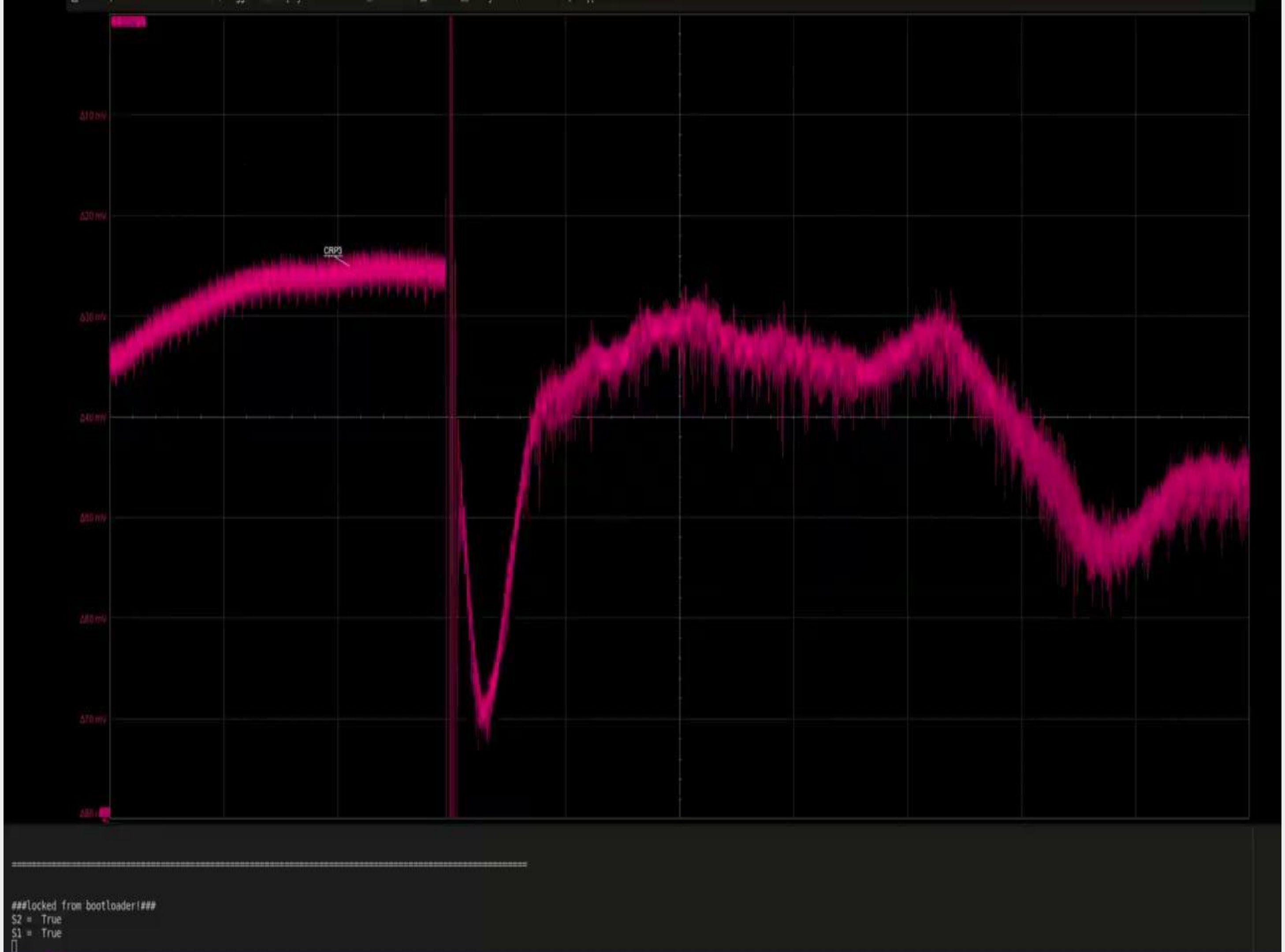
THE ATTACK

- We did the attack first with a **RESET** setup, but we didn’t succeed !!
- After we changed the setup**.** **POWER ON** is the trigger now**.**
- And did the attack we succeeded !!

THE ATTACK

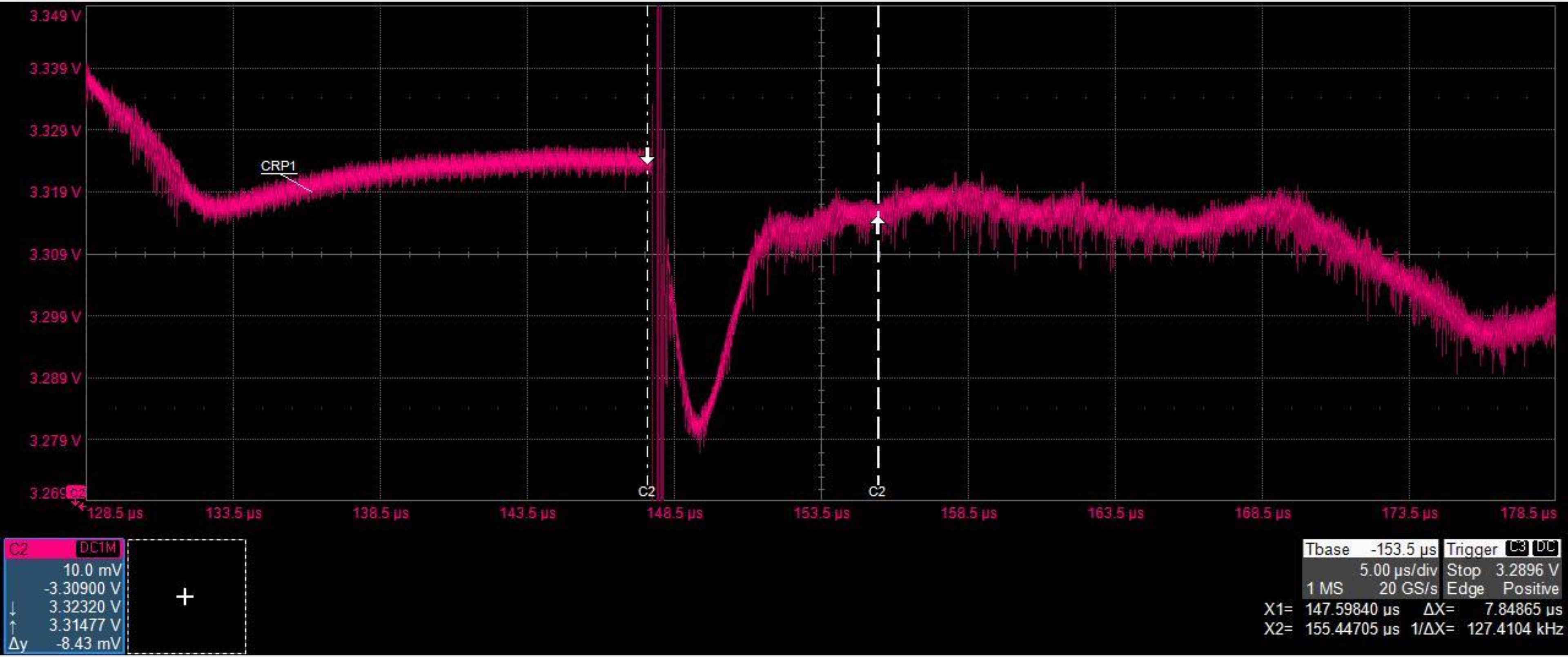

RESET
INITIALIZE
CRP1/2/3 ENABLED?
no
ENABLE DEBUG
yes
WATCHDOG FLAG SET?
yes
no
A
USER CODE VALID?
yes
no
CRP3/NO_ISP ENABLED?
no
yes
ENTER ISP MODE? (PIO0_1 =LOW)
no
yes
EXECUTE INTERNAL USER CODE

Recorded demo



```
================================================================================================

###locked from bootloader!###
S2 =  True
S1 =  True
▯
```

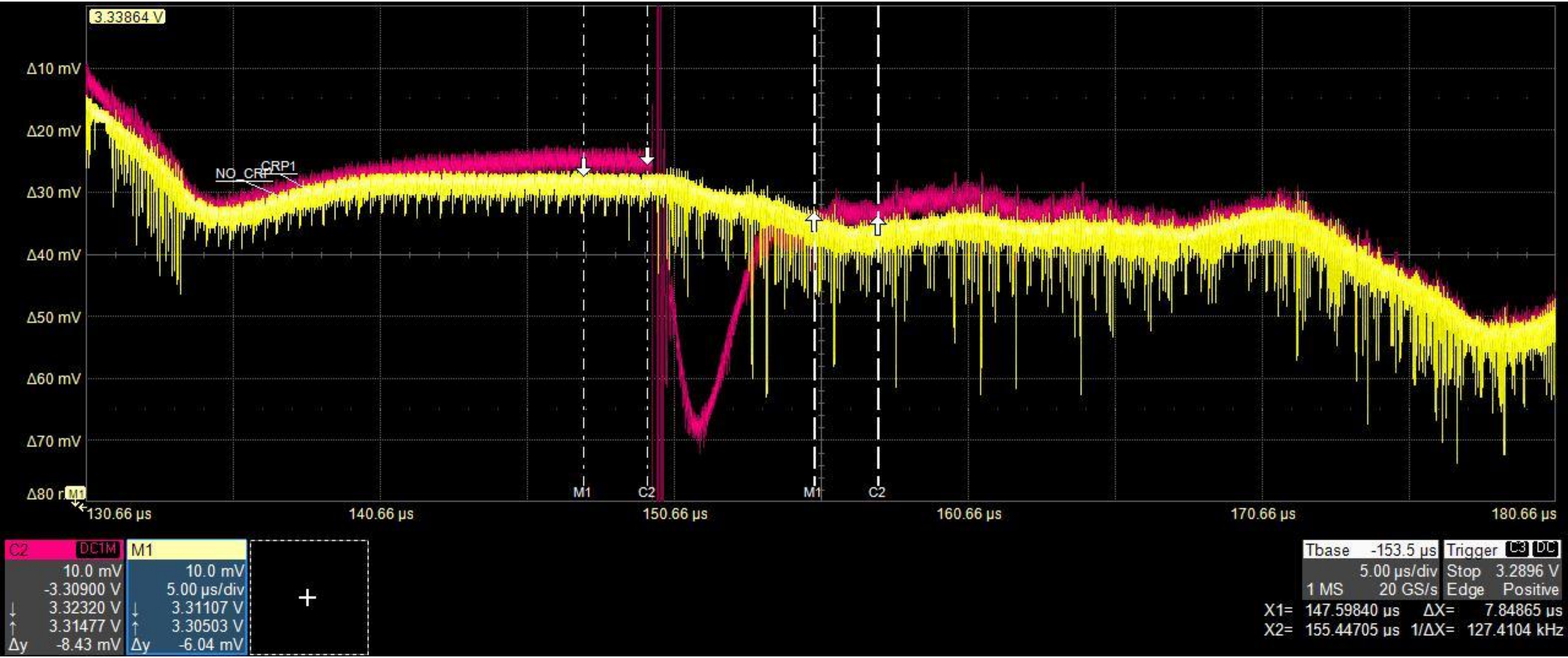
RESULT (CRP1 GLITCH)



```
C2  DC1M
      10.0 mV
  -3.30900 V
↓  3.32320 V
↑  3.31477 V
Δy  -8.43 mV
```

```
Tbase     -153.5 µs   Trigger  C3 DC
          5.00 µs/div Stop   3.2896 V
1 MS       20 GS/s    Edge   Positive
X1=  147.59840 µs    ΔX=     7.84865 µs
X2=  155.44705 µs  1/ΔX=   127.4104 kHz
```

RESULT (CRP1 & NO CRP)

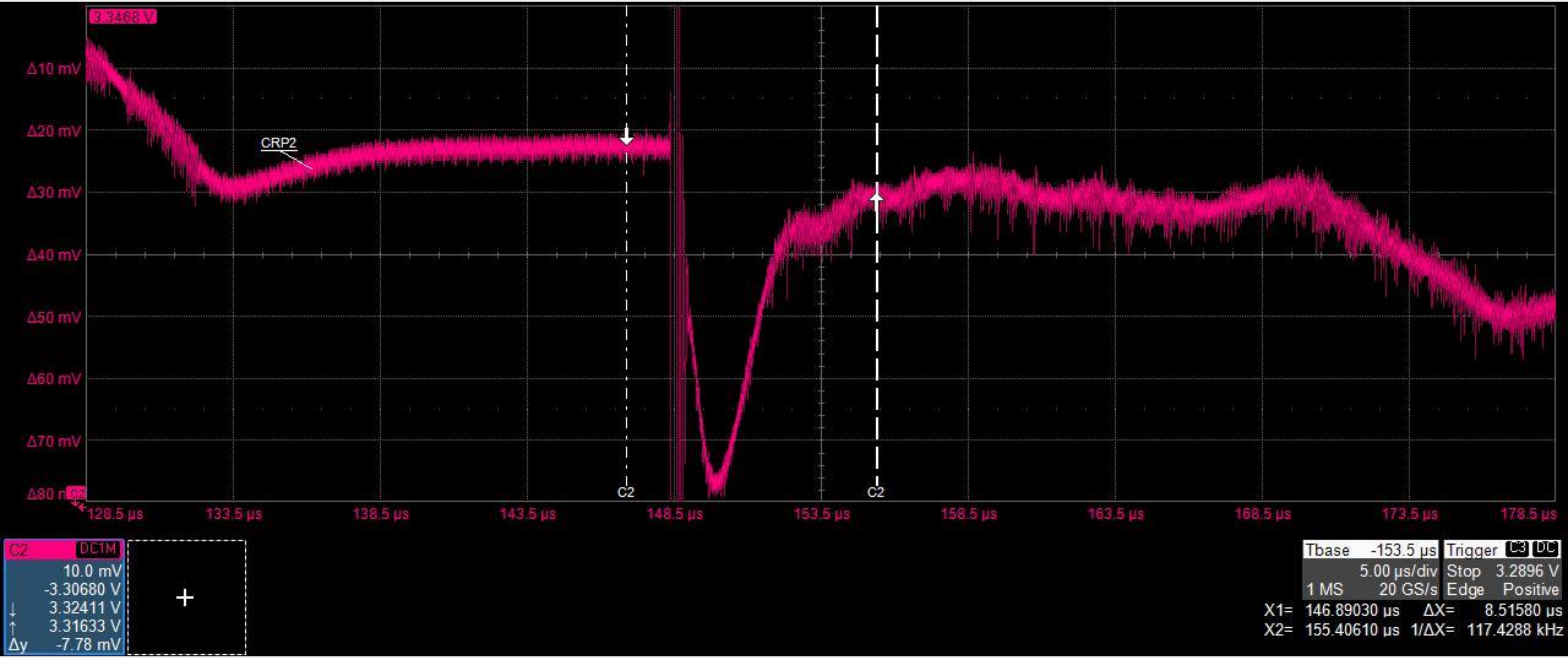


```
C2      DC1M    M1
     10.0 mV         10.0 mV
  -3.30900 V      5.00 µs/div
↓  3.32320 V    ↓  3.31107 V
↑  3.31477 V    ↑  3.30503 V
Δy  -8.43 mV    Δy  -6.04 mV
```

+

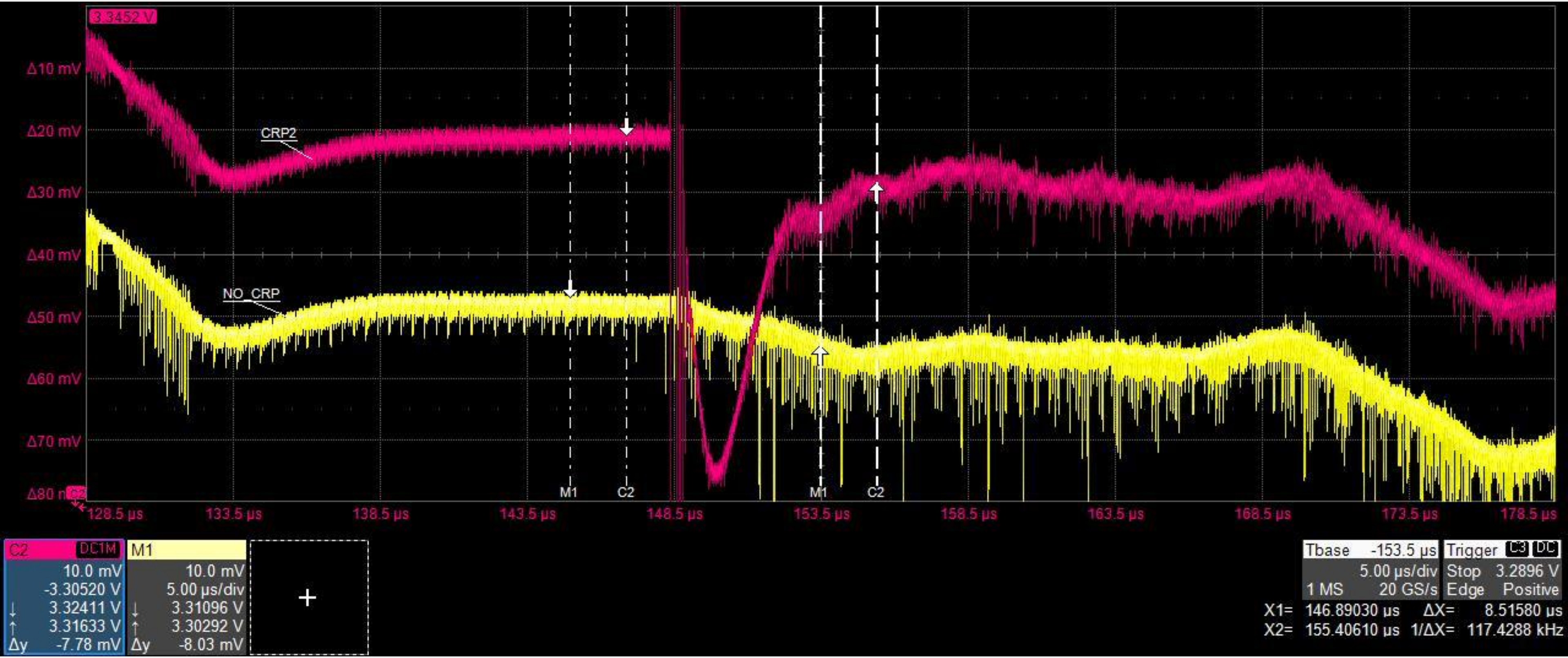
```
Tbase     -153.5 µs    Trigger  C3 DC
          5.00 µs/div  Stop   3.2896 V
1 MS       20 GS/s     Edge   Positive
X1=  147.59840 µs    ΔX=     7.84865 µs
X2=  155.44705 µs  1/ΔX=   127.4104 kHz
```

RESULT (CRP2 GLITCH)



```
C2        DC1M
          10.0 mV
     -3.30680 V
↓     3.32411 V
↑     3.31633 V
Δy     -7.78 mV

Tbase        -153.5 µs   Trigger  C3 DC
             5.00 µs/div Stop     3.2896 V
1 MS          20 GS/s    Edge     Positive
X1=  146.89030 µs   ΔX=    8.51580 µs
X2=  155.40610 µs   1/ΔX=  117.4288 kHz
```

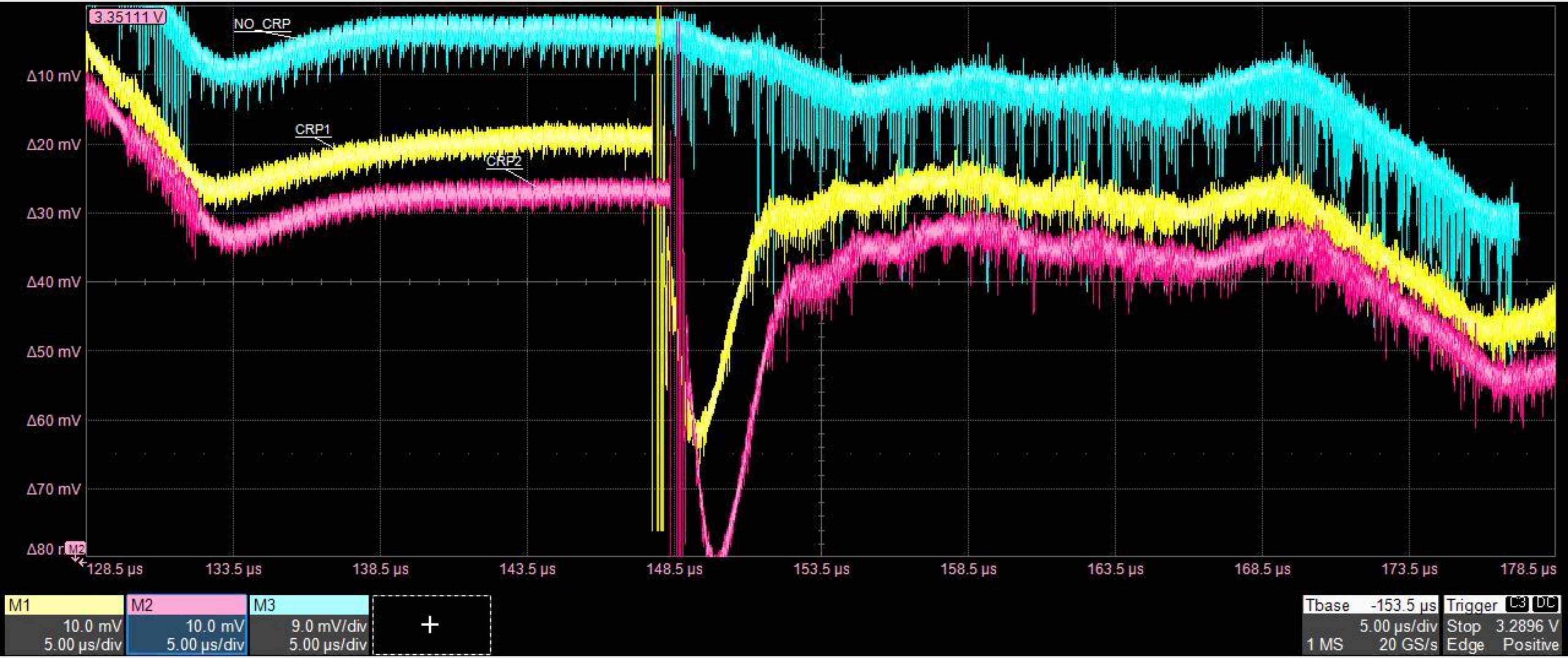
RESLUT (CRP2 & NO_CRP)



```
C2    DC1M     M1
      10.0 mV        10.0 mV
  -3.30520 V     5.00 µs/div
↓  3.32411 V   ↓  3.31096 V
↑  3.31633 V   ↑  3.30292 V
Δy  -7.78 mV   Δy  -8.03 mV
```

```
Tbase     -153.5 µs    Trigger  C3 DC
          5.00 µs/div  Stop     3.2896 V
1 MS      20 GS/s      Edge     Positive
X1=  146.89030 µs    ΔX=      8.51580 µs
X2=  155.40610 µs    1/ΔX=  117.4288 kHz
```

RESLUT (CRP1,2 & NO_CRP)

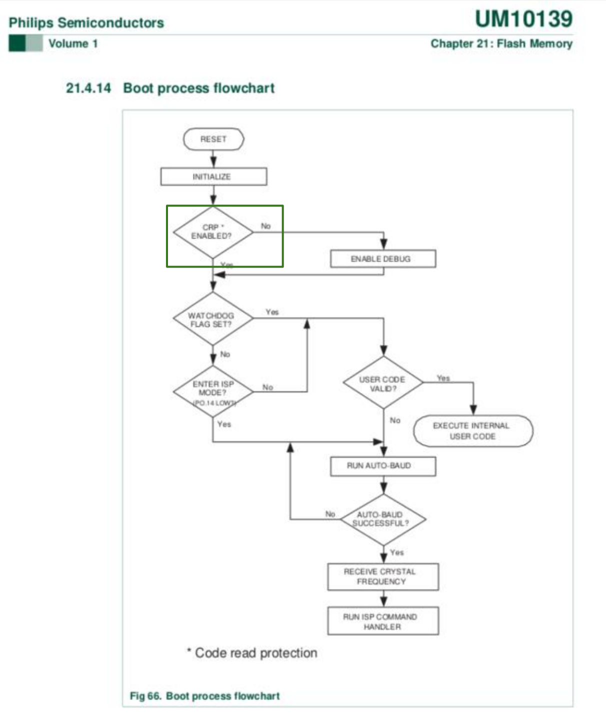


PHILIPS → NXP

UM10139

Volume 1: LPC214x User Manual

Rev. 01 — 15 August 2005 User manual



Philips Semiconductors UM10139

Volume 1 Chapter 21: Flash Memory

21.4.14 Boot process flowchart

* Code read protection

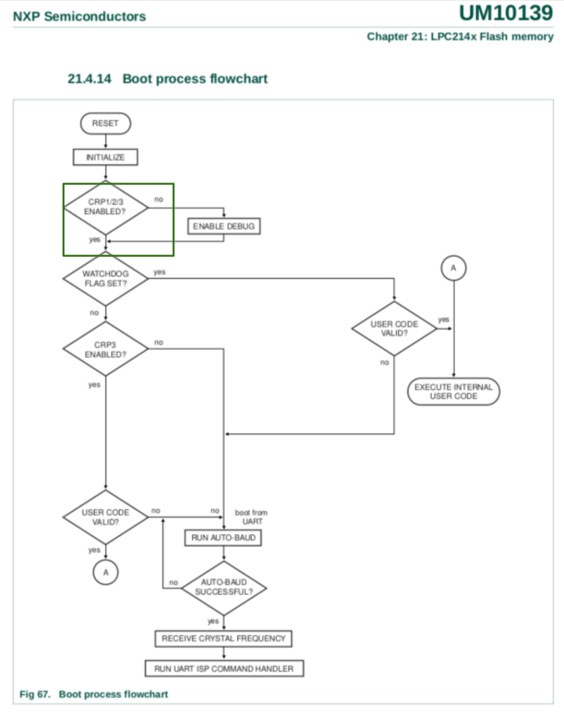
Fig 66. Boot process flowchart

https://www.keil.com/dd/docs/datashts/philips/user_manual_lpc214x.pdf

UM10139

LPC214x User manual

Rev. 4 — 23 April 2012 User manual



NXP Semiconductors UM10139

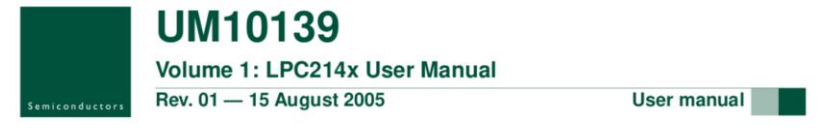
Chapter 21: LPC214x Flash memory

21.4.14 Boot process flowchart

Fig 67. Boot process flowchart

https://www.nxp.com/docs/en/user-guide/UM10139.pdf

PHILIPS → NXP

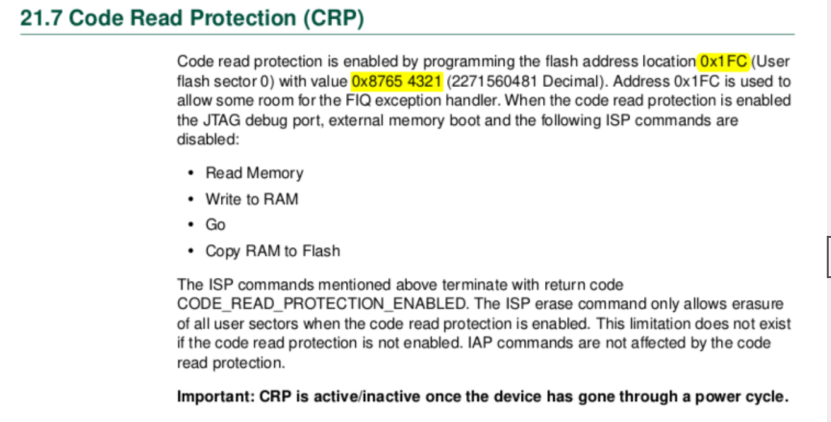


Semiconductors

UM10139

Volume 1: LPC214x User Manual

Rev. 01 — 15 August 2005 **User manual**



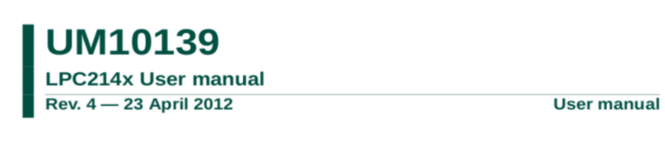
21.7 Code Read Protection (CRP)

Code read protection is enabled by programming the flash address location 0x1FC (User flash sector 0) with value 0x8765 4321 (2271560481 Decimal). Address 0x1FC is used to allow some room for the FIQ exception handler. When the code read protection is enabled the JTAG debug port, external memory boot and the following ISP commands are disabled:

- Read Memory
- Write to RAM
- Go
- Copy RAM to Flash

The ISP commands mentioned above terminate with return code CODE_READ_PROTECTION_ENABLED. The ISP erase command only allows erasure of all user sectors when the code read protection is enabled. This limitation does not exist if the code read protection is not enabled. IAP commands are not affected by the code read protection.

Important: CRP is active/inactive once the device has gone through a power cycle.



UM10139

LPC214x User manual

Rev. 4 — 23 April 2012 **User manual**

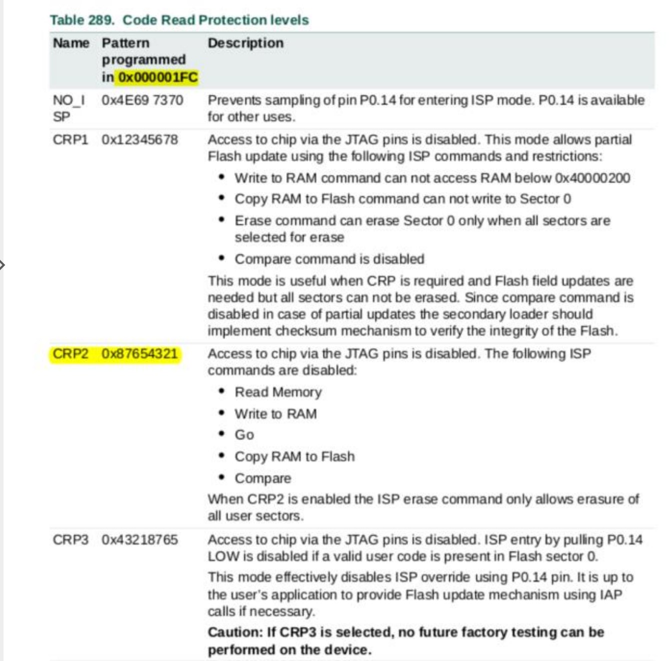


Table 289. Code Read Protection levels

Name	Pattern programmed in 0x000001FC	Description
NO_ISP	0x4E69 7370	Prevents sampling of pin P0.14 for entering ISP mode. P0.14 is available for other uses.
CRP1	0x12345678	Access to chip via the JTAG pins is disabled. This mode allows partial Flash update using the following ISP commands and restrictions: • Write to RAM command can not access RAM below 0x40000200 • Copy RAM to Flash command can not write to Sector 0 • Erase command can erase Sector 0 only when all sectors are selected for erase • Compare command is disabled This mode is useful when CRP is required and Flash field updates are needed but all sectors can not be erased. Since compare command is disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the Flash.
CRP2	0x87654321	Access to chip via the JTAG pins is disabled. The following ISP commands are disabled: • Read Memory • Write to RAM • Go • Copy RAM to Flash • Compare When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.
CRP3	0x43218765	Access to chip via the JTAG pins is disabled. ISP entry by pulling P0.14 LOW is disabled if a valid user code is present in Flash sector 0. This mode effectively disables ISP override using P0.14 pin. It is up to the user's application to provide Flash update mechanism using IAP calls if necessary. **Caution: If CRP3 is selected, no future factory testing can be performed on the device.**

*The value that disables JTAG:**0x87654321=CRP2**

https://www.keil.com/dd/docs/datashts/philips/user_manual_lpc214x.pdf

https://www.nxp.com/docs/en/user-guide/UM10139.pdf

LPC FAMILY

- Sharing same CRP mechanism and boot process (based on user manuals)
  - Cortex-M4
    - LPC5411X
    - LPC5410X
    - LPC43XX
    - LPC43SXX
    - LPC40(8/7)X
  - Cortex-M3
    - LPC18XX
    - LPC17XX
    - LPC15XX
    - LPC13XX
  - Cortex-M0
    - LPC8XX
    - LPC51U68
    - LPC11XX
    - LPC122X
  - ARM7
    - LPC2XXX

- We’ve successfully apply our attack on
  - LPC812 & LPC1114 & LPC1343
- **ECRP** is introduced in **LPC546xx**
- **OTP** banks is used in **LPC540xx…**
- **LPC55xx** (Cortex-M33 ARM trustzone)

CONCLUSION

- We reversed the bootROM and took power consumption traces
- Applied the attack based on **RESET** trigger. (Did Not succeed)
- Changed the setup, our glitch trigger now based on **POWER ON** (Succeeded!)
- CRP locking mechanisms exist since Philips time
- Contacted NXP to disclose (professional and fast)

Q&A

Thank you…

Resources

- LPC1311/13/42/43 User manual
- https://github.com/leveldown-security/SVD-Loader-Ghidra
- https://github.com/CPELyon/lpctools
- Gerlinsky, C. (2017, January 28). Breaking Code Read Protection on the NXP LPC-family Microcontrollers. Recon. https://recon.cx/2017/brussels/talks/breaking_crp_on_nxp.html
- ChipWhisperer-Lite https://www.newae.com/chipwhisperer