Breaking Code Read Protection on the NXP LPC-family Microcontrollers

Chris Gerlinsky
@akacastor
chris@pdrnorth.com
LPC Microcontrollers

IoT MCUs Based on ARM® Technology

LPC 32-bit ARM MCUs offer exceptional ease of use, design flexibility and advanced integration. LPC continues to transform the MCU landscape with its next-generation LPC800 and LPC54000 series focused on addressing today's IoT design challenges.

<table>
<thead>
<tr>
<th>LPC Series MCUs</th>
<th>CPU</th>
<th>Memory</th>
<th>Features</th>
<th>Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPC800</td>
<td>ARM® Cortex-M0+</td>
<td>8-32 KB Flash</td>
<td>SClTime/PWM, FlexCmrm</td>
<td>Sensor gateways, Battery-powered devices</td>
</tr>
<tr>
<td>LPC54000</td>
<td>Cortex-M4*</td>
<td>128-512 KB Flash</td>
<td>USB, CAN, Ethernet, FlexCmrm</td>
<td>Smart home and building automation, Auto-door system, Industrial control and networking, Gaming accessories</td>
</tr>
<tr>
<td>LPC1100</td>
<td>Cortex-M0+</td>
<td>4-32 KB Flash</td>
<td>USB, CAN, EEProm, FlexCmrm</td>
<td>White goods, Industrial control, UPS/power conversion</td>
</tr>
<tr>
<td>LPC1200</td>
<td>Cortex-M0+</td>
<td>32-128 KB Flash</td>
<td>8 LED, IEC 60730 Class B certified</td>
<td>Home automation, Solar inverters</td>
</tr>
<tr>
<td>LPC1300</td>
<td>Cortex-M3</td>
<td>8-64 KB Flash</td>
<td>USB, CAN, Ethernet, LCD, QEI</td>
<td>Consumer peripherals and toys, Home automation, Motor control, Digital power supplies, Solar inverters</td>
</tr>
<tr>
<td>LPC1500</td>
<td>Cortex-M3</td>
<td>64-256 KB Flash</td>
<td>USB, CAN, Ethernet, LCD, QEI</td>
<td>Smart energy, Data collectors, Auto-door system, Industrial control and networking, Medical diagnostics, Medical diagnostics, Medical diagnostics</td>
</tr>
<tr>
<td>LPC1700</td>
<td>Cortex-M4</td>
<td>32-512 KB Flash</td>
<td>USB, CAN, Ethernet, LCD, QEI</td>
<td>Secure industrial gateways, Data collectors, Portable medical equipment, Consumer audio applications</td>
</tr>
<tr>
<td>LPC1800</td>
<td>Cortex-M4</td>
<td>102-64 KB Flash*</td>
<td>Dual HS USB, CAN, Ethernet, LCD, Security</td>
<td>Security industrial gateways, Data collectors, Portable medical equipment, Consumer audio applications</td>
</tr>
<tr>
<td>LPC4000</td>
<td>Cortex-M4</td>
<td>64-512 KB Flash</td>
<td>USB, CAN, Ethernet, LCD, Security</td>
<td>Security industrial gateways, Data collectors, Portable medical equipment, Consumer audio applications</td>
</tr>
<tr>
<td>LPC4300</td>
<td>Cortex-M4*</td>
<td>104-320 KB Flash</td>
<td>Dual HS USB, CAN, Ethernet, LCD, Security</td>
<td>Security industrial gateways, Data collectors, Portable medical equipment, Consumer audio applications</td>
</tr>
<tr>
<td>LPC2000</td>
<td>ARM® Cortex-M4</td>
<td>8-512 KB Flash</td>
<td>USB, CAN, Ethernet, LCD, Security</td>
<td>Legacy MCUs for general embedded applications</td>
</tr>
<tr>
<td>LPC3000</td>
<td>ARM® Cortex-M4</td>
<td>Flashless</td>
<td>USB, CAN, Ethernet, LCD, Security</td>
<td>Legacy MCUs for general embedded applications</td>
</tr>
</tbody>
</table>

Source:
21.13 ISP commands

The following commands are accepted by the ISP command handler. Detailed status
codes are supported for each command. The command handler sends the return code
INVALID_COMMAND when an undefined command is received. Commands and return
codes are in ASCII format.

CMD_SUCCESS is sent by ISP command handler only when received ISP command has
been completely executed and the new ISP command can be given by the host.
Exceptions from this rule are "Set Baud Rate", "Write to RAM", "Read Memory", and "Go"
commands.

Table 317. ISP command summary

<table>
<thead>
<tr>
<th>ISP Command</th>
<th>Usage</th>
<th>Described in</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unlock</td>
<td>U &lt;Unlock Code&gt;</td>
<td>Table 318</td>
</tr>
<tr>
<td>Set Baud Rate</td>
<td>B &lt;Baud Rate&gt; &lt;stop bit&gt;</td>
<td>Table 319</td>
</tr>
<tr>
<td>Echo</td>
<td>A &lt;setting&gt;</td>
<td>Table 320</td>
</tr>
<tr>
<td>Write to RAM</td>
<td>W &lt;start address&gt; &lt;number of bytes&gt;</td>
<td>Table 321</td>
</tr>
<tr>
<td>Read Memory</td>
<td>R &lt;address&gt; &lt;number of bytes&gt;</td>
<td>Table 322</td>
</tr>
<tr>
<td>Prepare sector(s) for write operation</td>
<td>P &lt;start sector number&gt; &lt;end sector number&gt;</td>
<td>Table 323</td>
</tr>
<tr>
<td>Copy RAM to flash</td>
<td>C &lt;Flash address&gt; &lt;RAM address&gt; &lt;number of bytes&gt;</td>
<td>Table 324</td>
</tr>
<tr>
<td>Go</td>
<td>G &lt;address&gt; &lt;Mode&gt;</td>
<td>Table 325</td>
</tr>
<tr>
<td>Erase sector(s)</td>
<td>E &lt;start sector number&gt; &lt;end sector number&gt;</td>
<td>Table 326</td>
</tr>
<tr>
<td>Blank check sector(s)</td>
<td>I &lt;start sector number&gt; &lt;end sector number&gt;</td>
<td>Table 327</td>
</tr>
<tr>
<td>Read Part ID</td>
<td>J</td>
<td>Table 328</td>
</tr>
<tr>
<td>Read Boot code version</td>
<td>K</td>
<td>Table 330</td>
</tr>
<tr>
<td>Compare</td>
<td>M &lt;address1&gt; &lt;address2&gt; &lt;number of bytes&gt;</td>
<td>Table 331</td>
</tr>
<tr>
<td>ReadUID</td>
<td>N</td>
<td>Table 332</td>
</tr>
<tr>
<td>Name</td>
<td>Pattern programmed in 0x0000 02FC</td>
<td>Description</td>
</tr>
<tr>
<td>--------</td>
<td>----------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>NO_ISP</td>
<td>0x4E59 7370</td>
<td>Prevents sampling of pin PIO0_1 for entering ISP mode. PIO0_1 is available for other uses.</td>
</tr>
<tr>
<td>CRP1</td>
<td>0x12345678</td>
<td>Access to chip via the SWD pins is disabled. This mode allows partial flash update using the following ISP commands and restrictions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Write to RAM command cannot access RAM below 0x1000 0300.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Copy RAM to flash command can not write to Sector 0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Erase command can erase Sector 0 only when all sectors are selected for erase.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Compare command is disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Read Memory command is disabled.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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</td>
</tr>
<tr>
<td></td>
<td></td>
<td>disabled in case of partial updates the secondary loader should implement checksum mechanism to verify the integrity of the flash.</td>
</tr>
<tr>
<td>CRP2</td>
<td>0x87654321</td>
<td>Access to chip via the SWD pins is disabled. The following ISP commands are disabled:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Read Memory</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Write to RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Go</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Copy RAM to flash</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Compare</td>
</tr>
<tr>
<td></td>
<td></td>
<td>When CRP2 is enabled the ISP erase command only allows erasure of all user sectors.</td>
</tr>
<tr>
<td>CRP3</td>
<td>0x43218765</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This mode effectively disables ISP override using PIO0_1 pin. It is up to the user’s application to provide a flash update mechanism</td>
</tr>
<tr>
<td></td>
<td></td>
<td>using IAP calls or call reinvoke ISP command to enable flash update via UART0.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Caution: If CRP3 is selected, no future factory testing can be performed on the device.</td>
</tr>
</tbody>
</table>
21.10 Boot process flowchart

Source: LPC1343 User Manual
Values at FLASH:02FC that **enable** Code Read Protection:  
0x12345678  CRP1  
0x87654321  CRP2  
0x43218765  CRP3  
0x4E697370  NO_ISP

4 possible 32-bit words

Values at FLASH:02FC that **disable** Code Read Protection:  
0xFFFFFFFF  
0x00000000  
0xABCDEF12  
0x12345679  
0x87654320  
0x53218765  
0x4E687370  
... 

4,294,967,292 possible 32-bit words
Useful tools and links:

lpc21isp – software to program LPC-family microcontrollers using ISP

http://eleceng.dit.ie/frank/arm/BareMetalLPC812/index.html
http://eleceng.dit.ie/frank/arm/BareMetalLPC812/serial.tar.gz

http://eleceng.dit.ie/frank/arm/BareMetalLPC1114/index.html


https://www.olimex.com/Products/ARM/NXP/LPC-P2148/

http://siwawi.bauing.uni-kl.de/avr_projects/arm_projects/lpc2k_bundle_port/
lpc213x_lpc214x_examples_20061205.zip
void dump_bootrom(void)
{
    unsigned char *FlashAddr;

    // loop through boot ROM 1FFF0000 - 1FFFFFFF and send it as ASCII hex bytes
    for( FlashAddr=(unsigned char *)0x1FFF0000; FlashAddr<=(unsigned char *)0x1FFFFFFF; FlashAddr++ )
        printf( "\%02X ", *FlashAddr );
}
```
LDR    R2, =dword_400483F0
LDR    R3, =addr_of_FLASH_CRP
LDR    R5, =CRP3_value
LDR    R3, [R3] ; FLASH_CRP_location
LDR    R5, [R5] ; CRP3
LDR    R4, [R3] ; CRP1
LDR    R6, =CRP1_value
LDR    R4, [R6]
CMP    R4, R5
BEQ    in_mode_CRP1_or_CRP3
BNE    not_CRP1
LDR    R4, =CRP2_value
LDR    R4, [R4]
STR    R4, [R2]
LDR    R2, =FMC_4003C000
LDR    R3, [R2]
movs   R4, #0x40
URS    R3, R4
STR    R3, [R2]
```

PUSH {R3-R7, LR}  ; FMC flash controller base address
LDR R2, =FMC_4003C000
LDR R0, [R2]
LDRA 10 68
LDR R0, [R2]
Mohs R3, #0x40  ; '8'
; clear bit 0x40 of FMC_4003C000 to enable access to user flash area
STM R0, [R2]
LDR R0, =addr_of_FLASH_CRP  ; FLASH:02FC is where CRP value is located in application flash
LDR R1, =dword_1000017C  ; this + 8 = CRP_value_in_RAM
LDR R0, [R0]  ; FLASH_CRP_location; FLASH:02FC is where CRP value is located in application flash
LDR R0, [R0]
STM R0, [R1, #0x1000017C]

LDR R0, [R7, #0x30]  ; CODE XREF: boot_continue_startup+AA1
LDR R0, [R0, #0x30]
LSLS R0, R0, #29
BMI to_run_flash_app_or_bootloader  ; branch if bit 2 of SYSTICK is set
LDR R0, =unk_50003FC0  ; GPIODATAMASK
LDR R0, [R0, #0x3C]  ; R0 = GPIODATA
LSLS R0, R0, #30
BMI to_run_flash_app_or_bootloader  ; branch if bit GPIO0.1 is set (R0.1 not held low for bootloader
LDR R0, =dword_1000017C  ; this + 8 = CRP_value_in_RAM
LDR R0, =CRP3_value
LDR R0, [R0, #0x1000017C]  ; CODE XREF: boot_continue_startup+CB

CMP R0, R1
BEQ to_run_flash_app_or_bootloader
LDR R1, =0x4e697370  ; CMP NO_ISP value
LDR R0, R1
CMP R0, R1
BEQ to_run_flash_app_or_bootloader
BEQ to_run_flash_app_or_bootloader
LDR R0, R1
to_select_bootloader

run_flash_app_or_bootloader  ; CODE XREF: boot_continue_startup+to_run_flash_app_or_bootloader
PUSH {R4, LR}  ; starts user application in FLASH if is valid
BL run_flash_app_if_valid
BL select_bootloader

End of function run_flash_app_or_bootloader
select_bootloader

; CODE XREF: sub_1FFF01A4+6↑p
; run_flash_app_or_bootloader+6↑p
; boot_continue_startup:to_select_bootloader↓p

PUSH {R4,LR}
LDR R0, =FMC_4003C000
LDR R2, [R0]
MOVS R1, #0x40 ; '0'
ORRS R2, R1
STR R2, [R0]
LDR R2, =dword_430
LDR R4, [R2]
LDR R2, [R0]
LDR R1
STR R2, [R0]
LDR R0, =addr_of_FLASH_CRP ; FLASH:02FC is where CRP value is located in application flash
LDR R1, =dword_1000017C ; this + 8 = CRP_value_in_RAM
LDR R0, [R1, #8, LSL #2] ; FLASH_CRP_location ; FLASH:02FC is where CRP value is located in application flash
LDR R0, [R0]
STR R0, [R1, #8, LSL #2] ; (CRP_value_in_RAM - 0x1000017C)
BL sub_1FFF1226 ; some CLK setup (?)
LSLS R0, R4, #31
BNE to_enter_serial_ISP
LDR R0, =unk_50003FC0
LDR R0, [R0, #0x3C] ; R0 = GPIODATA
LSLS R0, R0, #28 ; check P0.3 / USB_VBUS
BPL to_enter_serial_ISP ; if P0.3 / USB_VBUS is low then enter serial ISP
; (high would indicate USB bootloader should be started)

BL to_enter_serial_ISP

BL sub_1FFF119A
BL sub_1FFF235E

; CODE XREF: select_bootloader+26↑j
; select_bootloader+26↑j

to_enter_serial_ISP

BL enter_serial_ISP
serial_ISP

var_20 = 0x20
var_1C = 0x1C

; CODE XREF: enter_serial_ISP+4A

PUSH R1-R7, LR
LDR R6, =CRP2_value
LDR R7, =CRP3_value
LDR R5, =dword_10000248

; CODE XREF: serial_ISP+16
; serial_ISP+3A
; serial_ISP+9E
; serial_ISP+A4

LDR R0, =dword_10000248
ADD R2, SP, #0x20+var_1C
MOV R1, #0x46 ; 'E'
SUBS R0, #0x94 ; ''
BL uart_gets

; this does serial RX of ASCII string

BNE isp_command_loop
```assembly
; CODE XREF: serial ISP+42Tj
; serial ISP+461j
; serial ISP+46A

MOVW R2, #1
R2-1 indicates command not allowed if CRP1, 2, or 3 enabled

; local FF1010
MOVW R2, #0
; CODE XREF: serial ISP+45Tj
; serial ISP+45Tj
; serial ISP+45Tj
; serial ISP+45Tj

CMF R1, #0x85 ; 'W'
; 'W'rite command
BEQ command_disabled_if_CRP2_or_3
CMF R1, #0x43 ; 'C'
; command_not_disabled_by_CRP
BNE command_not_disabled_by_CRP

; command_disabled_if_CRP2_or_3
MOVW R4, #1
; CODE XREF: serial ISP+58Tj
; serial ISP+58Tj

; check_current_CRC_level
LDR R1, =word_1000017C ; this = 8 - CRP_value in RAM
LDR R1, [R1, #0x8000017C]
CMP R1, R3
; check if CRP2 is set
BEQ CRP2_2_or_3_is_enabled ; Branch if CRP2 is set

; CRP2_2_or_3_is_enabled
MOV R3, [R7]
CMP R3, [R7]
; check if CRP3 is set
BEQ CRP3_2_or_3_is_enabled ; Branch if CRP3 is set

; CRP3_2_or_3_is_enabled
MOV R3, [R7]
CMP R3, [R7]
; check if CRP3 is set
BEQ CRP3_2_or_3_is_enabled ; Branch if CRP3 is set

; CRP3_2_or_3_is_enabled
LDR R3, [R7]
CMP R3, [R7]
; check if CRP3 is set
BEQ CRP3_2_or_3_is_enabled ; Branch if CRP3 is set

; CRP3_2_or_3_is_enabled
LDR R2, [R6]
CMP R2, R2
; CRP2 / CRP2 / or CRP3
BEQ disallow_command ; Branch if this command requires elevated privileges (ie: not allowed in CRP1/2/3 modes)

; disallow_command
LDR R2, [R6]
CMP R2, R2
; CRP2 / CRP3
BEQ disallow_command ; Branch if this command requires elevated privileges (ie: not allowed in CRP1/2/3 modes)

; CRP2_2_or_3_is_enabled
LDR R2, [R6]
CMP R2, R2
; CRP2 / CRP3
BEQ disallow_command ; Branch if this command requires elevated privileges (ie: not allowed in CRP1/2/3 modes)

; no_CRP_is_set
LDR R2, [R6]
CMP R2, R2
; CRP2 / CRP3
BEQ disallow_command ; Branch if this command requires elevated privileges (ie: not allowed in CRP1/2/3 modes)

; CRP2_or_3_is_enabled
LDR R2, [R6]
CMP R2, R2
; CRP2 / CRP3
BEQ disallow_command ; Branch if this command requires elevated privileges (ie: not allowed in CRP1/2/3 modes)

; disallow_command
MOV R2, #0x7F
MOV R0, #19 ; R0 = response code (19 indicates command disabled by CRP)
```
LPC812

Voltage measured across 10 ohm resistor in series with GND

Reset and start bootloader

Reset and start application
Reset and start bootloader

LPC812

Voltage measured across 10 ohm resistor in series with GND

Reset and start application
MAX4617/MAX4618/MAX4619
High-Speed, Low-Voltage, CMOS Analog Multiplexers/Switches

General Description

The MAX4617/MAX4618/MAX4619 are high-speed, low-voltage, CMOS analog ICs configured as an 8-channel multiplexer (MAX4617), two 4-channel multiplexers (MAX4618), and three single-pole/double-throw (SPDT) switches (MAX4619).

These CMOS devices can operate continuously with a +2V to +5.5V single supply. Each switch can handle rail-to-rail analog signals. The off-leakage current is only 1nA at $T_A = +25^\circ C$ and 10nA at $T_A = +85^\circ C$.

All digital inputs have 0.8V to 2.4V logic thresholds, ensuring TTL/CMOS-logic compatibility when using a single +5V supply.

Features

- Fast Switching Times
  - 15ns $t_{ON}$
  - 10ns $t_{OFF}$
- Pin Compatible with Industry-Standard
  - 74HC4051/74HC4052/74HC4053 and
  - MAX4581/MAX4582/MAX4583
- Guaranteed On-Resistance
  - 10Ω max (+5V Supply)
  - 20Ω max (+3V Supply)
- Guaranteed 1Ω On-Resistance Match Between Channels (single +5V supply)
- Guaranteed Low Off-Leakage Current:
  - 1nA at +25°C
- Guaranteed Low On-Leakage Current:
  - 1nA at +25°C
- +2V to +5.5V Single-Supply Operation
- TTL/CMOS-Logic Compatible

Source: MAX4619 datasheet
U1 74HC4053/MAX4619

- Target glitch VCC from supply
- Target normal VCC from supply
- Output to target chip VCC

- Xmega PORTA.3
- Enable VCC output to target chip
- Select between 'normal' and 'glitch' VCC level
Xmega-A1 Xplained

LPC812

MAX4619

10 ohm resistor in series with LPC812 GND
Input from adjustable power supply
red = normal voltage level
green = glitched voltage level
white = ground

Target chip Vdd
Target chip GND
// pin GVCC_EN (port bit 3) must be 0 to enable VCC output to chip
// pin GVCC_SW (port bit 0) is 0 for normal VCC, 1 for glitched VCC
uint8_t glitch_form[8] = {{1<<GVCC_SW), (1<<GVCC_SW), (1<<GVCC_SW), (1<<GVCC_SW), (1<<GVCC_SW), (1<<GVCC_SW), (1<<GVCC_SW), 0, 0};
uint8_t noglitch_form = 0x00;
// GVCC_EN = 0, GVCC_SW = 0

uint16_t glitch_delay = 1; // delay (in 32 MHz cycles) between trigger and glitch

asm volatile(
    "call DelayCycles" "\n"
    "out 0x0011, %0" "\n"
    "out 0x0011, %1" "\n"
    "out 0x0011, %2" "\n"
    "out 0x0011, %3" "\n"
    "out 0x0011, %4" "\n"
    "out 0x0011, %5" "\n"
    "out 0x0011, %6" "\n"
    "out 0x0011, %7" "\n"
    "out 0x0011, %8" "\n"
    : : "r" (glitch_form[0]),
       "r" (glitch_form[1]),
       "r" (glitch_form[2]),
       "r" (glitch_form[3]),
       "r" (glitch_form[4]),
       "r" (glitch_form[5]),
       "r" (glitch_form[6]),
       "r" (glitch_form[7]),
       "r" (noglitch_form),
       "r" (glitch_delay)
); // call DelayCycles() for single-cycle precision delay

// 0x0011 is VPORTE out

// OUT is a single-cycle opcode - executing a series of OUT instructions
// changes the output port each cycle @ the 32 MHz Xmega clock speed

// VCC glitch waveform:
//
// 0 0 0 0 0 0 1
//
// ^___________/
// ^           
// |------------|  
// | 8 clk @32 MHz |
// keep repeating glitch-test for() loop
while(1)
{
  b = NUM_GLITCH_LOOPS; // initial value of 'b' before glitch loop is NUM_GLITCH_LOOPS

  while( GPIO_PORT0 & (1<<13) ) // wait for PIO0_13 to go low (synchronize with glitcher)
    ;

  while( !(GPIO_PORT0 & (1<<13)) ) // wait for PIO0_13 to go high
    ;

  GPIO_CLRPO |= (1<<17); // pull PIO0_17 low while running for() loop

  for( a=0; a<NUM_GLITCH_LOOPS; a++ ) // NUM_GLITCH_LOOPS is #defined as 0x100
    b--;

  GPIO_SETP0 |= (1<<17); // set PIO0_17 high to indicate for() loop has completed

  // expected values:
  // a == NUM_GLITCH_LOOPS (0x100)
  // b == 0
  if( a == NUM_GLITCH_LOOPS && b == 0 ) // no glitch occurred - print '.' to indicate 'a' and 'b' not glitched
    pp_printf(".");
  else // glitch occured - print 'a' and 'b' values
    pp_printf("\n a: %08X  b: %08X\n", a, b);
}
Assembler code generated by GCC

84:main.c  ****
85:main.c  ****
224 .loc 2 85 0
225 00cc 0023
226 00ce 7B60
227 00d0 05E0
228 .L9:
86:main.c  ****
229 00d2 3B68
230 00d4 013B
231 00d6 3B60
85:main.c  ****
232 00d8 7B68
233 00da 0133
234 00dc 7B60
235 .L8:
85:main.c  ****
236 00de 7B68
237 00e0 0F2B
238 00e2 F6DD
239
240
241

for( a=0; a<NUM_GLITCH_LOOPS; a++ )

asm
decrement 'b'
increment 'a'
check if a < NUM_GLITCH_LOOPS
Video: https://youtu.be/gr_qiN5DDqo
Loop executes normally (no glitch)

\[ \begin{align*}
  a & : 00000010 \\
  b & : fffffff1 \\
\end{align*} \]

\[ \begin{align*}
  a & : 00000010 \\
  b & : 00000000 \\
\end{align*} \]

\[ \begin{align*}
  a & : 50030008 \\
  b & : 0000000e \\
\end{align*} \]

\[ \begin{align*}
  a & : 00000010 \\
  b & : fffffff1 \\
\end{align*} \]

\[ \begin{align*}
  a & : 00000010 \\
  b & : ffffffff2 \\
\end{align*} \]
Loop executes normally (no glitch)

a: 00000010
b: 00000000

a: 00000010
b: ffffffff

a: 50030009
b: 00000009

a: 00000021
b: 100003fc
$ cortex-m3/tools/dump 0 32768 > LPC1343_FLASH.bin
Opening serial port /dev/ttyACM0
Forcing boot loader mode
Syncronizing... done
Identifying... done
part number: 3d00002b
LPC1343, 32kB Flash, 8kB RAM
sbl_config.h:
#define CRP1 0x12345678
#define CRP2 0x87654321
#define CRP3 0x43218765
#define NOCRP 0x11223344

Readout only allowed if CRP word == NOCRP
No CRP
; CODE XREF: R0M:1FFF0106↑
; DATA XREF: R0M:10C_1FFF010409
; R0M:0ff_1FFF0108↑

boot_Startup

ROM:1FFF010C
ROM:1FFF010C
ROM:1FFF010C
ROM:1FFF010C
ROM:1FFF010C 1A 4A LDR R2, =dword_400483F0
ROM:1FFF010C 1B 4B LDR R3, =addr_of_FLASH_CRP ; FLASH:02FC in where CRP value is located in application flash
ROM:1FFF0110 1B 68 LDR R3, [R3] ; FLASH_CRP_location ; FLASH:02FC is where CRP value is located in application flash
ROM:1FFF0112 1C 4D LDR R5, =CRP3_value ; CRP3 0x43218765
ROM:1FFF0114 2D 6B LDR R5, [R5] ; CRP3 0x43218765
ROM:1FFF0116 1C 4E LDR R6, =CRP1_value ; CRP1 0x12345678
ROM:1FFF0118 36 6B LDR R6, [R6] ; CRP1 0x12345678
ROM:1FFF011A 1C 6B LDR R4, [R3] ; R4 contains CRP word from FLASH:02FC
ROM:1FFF011C AC 42 CMP R4, R5
ROM:1FFF011E 01 D0 BEQ in_mode_CRP1_or_CRP3
ROM:1FFF0120 B4 42 CMP R4, R6
ROM:1FFF0122 01 D1 BNE not_CRP1 ; CODE XREF: boot_Startup+12↑)

in_mode_CRP1_or_CRP3

ROM:1FFF0124
ROM:1FFF0124
ROM:1FFF0124
ROM:1FFF0124 16 4C LDR R4, =CRP2_value ; in modes CRP1 or CRP3, load R4 with same value as CRP2 (0x87654321)
ROM:1FFF0126 24 6B LDR R4, [R4] ; CRP2 0x87654321
ROM:1FFF0128

not_CRP1 ; CODE XREF: boot_Startup+16↑)

ROM:1FFF0128
ROM:1FFF0128
ROM:1FFF0128
ROM:1FFF0128 14 60 STR R4, [R2] ; store CRP2 value @ 400483F0 to disable debug mode
ROM:1FFF012A 0D 4A LDR R2, =FMC_4003C000 ; FMC flash controller base address
ROM:1FFF012C 13 68 LDR R3, [R2] ; FMC flash controller base address
ROM:1FFF012E 40 24 MOV R4, #0x40 ; '0'
ROM:1FFF0130 23 43 ORRS R3, R4 ; FMC flash controller base address
ROM:1FFF0132 13 60 STR R3, [R2] ; FMC flash controller base address