

Rpi cheat sheet

Code structure:

Main program (.text):

- ① Initialize Vector Table (IRQ/FIQ)
- ② Init the stack/s for FIQ/IRQ modes
- ③ Init the stack for SVC mode (SVC mode selected)
- ④ Configure GPIOs (I&O)
- ⑤ Configure peripheral interruption: timer/push-buttons)
- ⑥ Local enabling of configured interrupts
- ⑦ Global enabling of interrupts (SVC mode)
- ⑧ Infinite loop (polling of device/s?)

IRQ/FIQ Handler:

- ① Push registers to be used
- ② Source of interruption?
- ③ Perform handler work depending on ②
- ④ Clear event (notify to device IRQ/FIQ has been served)
- ⑤ Pop registers
- ⑥ Return from handler

Pin No.		
3.3V	1	2 5V
GPIO2	3	4 5V
GPIO3	5	6 GND
GPIO4	7	8 GPIO14 Tx
GND	9	10 GPIO15
GPIO17	11	12 GPIO18 Rx
GPIO27	13	14 GND
GPIO22	15	16 GPIO23
3.3V	17	18 GPIO24
GPIO10	19	20 GND
GPIO9	21	22 GPIO25
GPIO11	23	24 GPIO8
GND	25	26 GPIO7

GPIO: configuration as I/O(1), write(2), read(3)

- 000: Input pin
- 001: Output pin
- 010-111: Other modes

GPFSSEL0-5 (3F20 0000)

1. Configure: GPIO9 as Output

X	FSEL9	FSEL8	FSEL7	FSEL6	FSEL5	FSEL4	FSEL3	FSEL2	FSEL1	FSEL0
	0	0	1							

GPSET0-1 (3F20 001C)

2. Writing 1: Set GPIO9 (turn the led on)

SET0		SET1		SET2		SET3		SET4		SET5		SET6		SET7		SET8		SET9	1	SET10		SET11		SET12		SET13		SET14		SET15		SET16		SET17		SET18		SET19		SET20		SET21		SET22		SET23		SET24		SET25		SET26		SET27		SET28		SET29		SET30		SET31	
------	--	------	--	------	--	------	--	------	--	------	--	------	--	------	--	------	--	------	---	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--

GPCLR0-1 (3F20 0028)

2. Writing 0: Clear GPIO9 (turn the led off)

CLR0		CLR1		CLR2		CLR3		CLR4		CLR5		CLR6		CLR7		CLR8		CLR9	1	CLR10		CLR11		CLR12		CLR13		CLR14		CLR15		CLR16		CLR17		CLR18		CLR19		CLR20		CLR21		CLR22		CLR23		CLR24		CLR25		CLR26		CLR27		CLR28		CLR29		CLR30		CLR31	
------	--	------	--	------	--	------	--	------	--	------	--	------	--	------	--	------	--	------	---	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--	-------	--

3. Read: GPIO2

GPLEV0-1 (3F20 0034)

0		1		2		3		4		5		6		7		8		9		10		11		12		13		14		15		16		17		18		19		20		21		22		23		24		25		26		27		28		29		30		31	
---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	---	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--	----	--

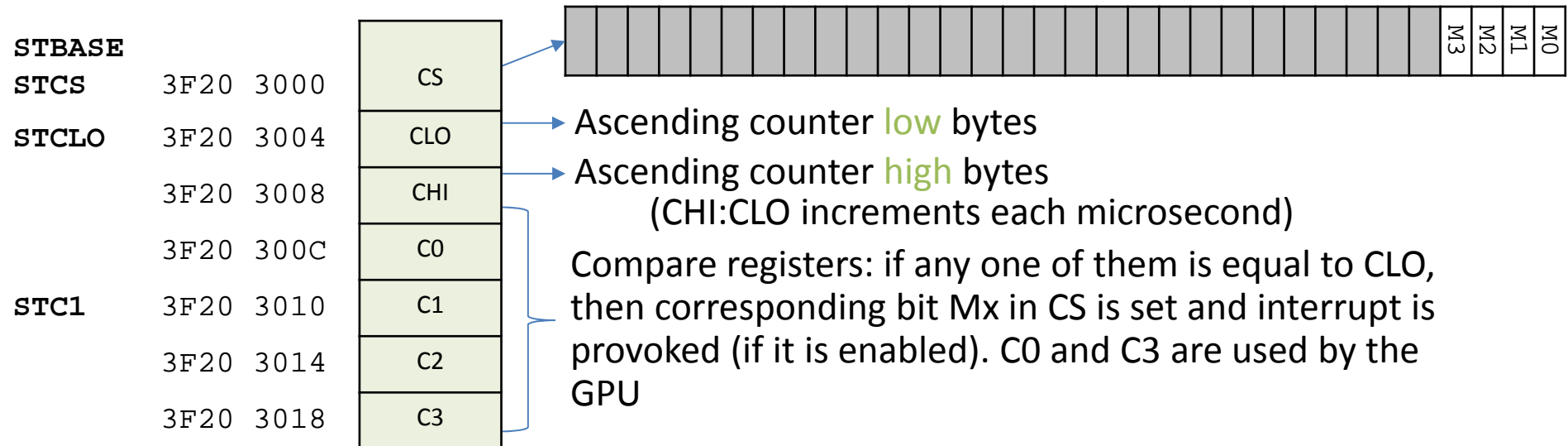
Example: Code for checking push button (GPIO2) and turning led (GPIO9) on

```
.set GPBASE, 0x3F200000
.set GPFSEL0, 0x00
.set GPSET0, 0x1c
.set GPLEV0, 0x34
.text
    ldr r0, =GPBASE
/* guia bits      xx999888777666555444333222111000 */
    mov r1, #0b00001000000000000000000000000000
    str r1, [r0, #GPFSEL0]
/* mask for testing GPIO2 */
    mov r2, #0b00000000000000000000000000000000100
bucle:
    ldr r3, [r0, #GPLEV0]
    tst r3, r2
    bne bucle
/* guia bits      10987654321098765432109876543210
    mov r1, #0b0000000000000000000000001000000000
    str r1, [r0, #GPSET0]
infi: b infi
```

4 Configure GPIOs (I&O)

8 Polling, infinite loop

Timer (polling example)



Delay loop (polling)

```

        ldr r0, =STBASE      @ r0 is an input parameter (ST base address)
        ldr r1, =500000      @ r1 is an input parameter (waiting time in microseconds)

espera:  push {r4, r5}        @ Save r4 and r5 in the stack
        ldr r4, [r0, #STCLO] @ Load CLO timer
        add r4, r1            @ Add waiting time -> this is our ending time
ret1:    ldr r5, [r0, #STCLO] @ Enter waiting loop: load current CLO timer
        cmp r5, r4            @ Compare current time with ending time
        blo ret1              @ If lower, go back to read timer again
        pop {r4, r5}          @ Restore r4 and r5
        bx lr                 @ Return from routine

```

IRQ (timer, push-button). Main program.

1 Initialize Vector Table (IRQ)

```
mov    r0, #0
ADDEXC 0x18, irq_handler
```

2 Stack init for IRQ mode

```
mov    r0, #0b11010010
msr    cpsr_c, r0
mov    sp, #0x8000
```

③ Stack init for SVC mode

```
mov    r0, #0b11010011
msr    cpsr_c, r0
mov    sp, #0x80000000
```

5 Configure timer IRQ

```
ldr    r0, =STBASE
ldr    r1, [r0, #STCLO]
add    r1, #y    @y microseconds
str    r1, [r0, #STC1]
```

6 Enable timer interrupt by comparator C1:

```
ldr      r0,=INTBASE
mov      r1, #0b0010
str      r1,[r0,#INTENIRQ1]
```

7 Enable IRQ (SVC mode):

```
mov    r1, #0b01010011
msr    cpsr_c, r1
```

5 Configure push-button interruption (GPIO 2)

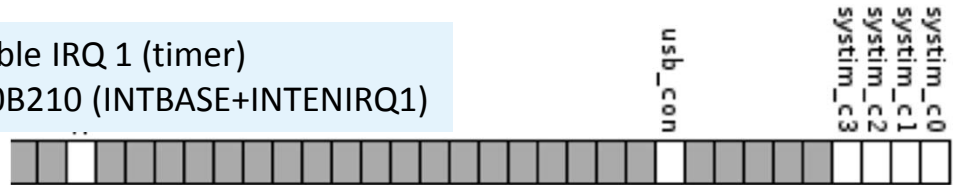
[illegible]

⑥ Enable push-button interruption (IRQ):

```
ldr    r0, =INTBASE
mov    r1, #0b00000000000010000000000000000000
/* guia bits      10987654321098765432109876543210 */
str    r1, [r0, #INTENIRQ2]
```

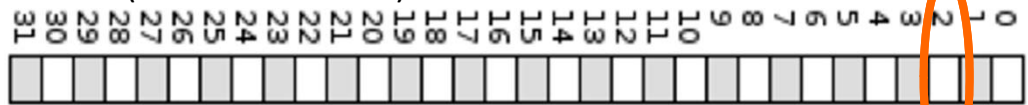
Enable IRQ 1 (timer)

3F00B210 (INTBASE+INTENIRQ1)



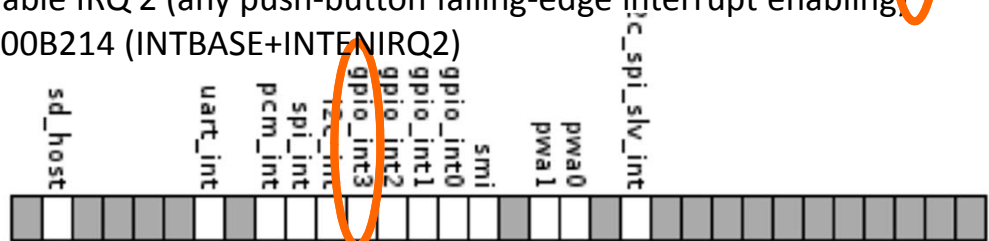
GPFEN0 (push-button falling-edge interrupt configuration, GPIO 2)

3F200058 (GPBASE+GPFEN0)



Enable IRQ 2 (any push-button falling-edge interrupt enabling

3F00B214 (INTBASE+INTENIRQ2)



IRQ (timer, push-button). Handler.

Timer:

① Push registers to be used
push {r0, r1, r2}

Push-button:

2 Source of timer interruption?:

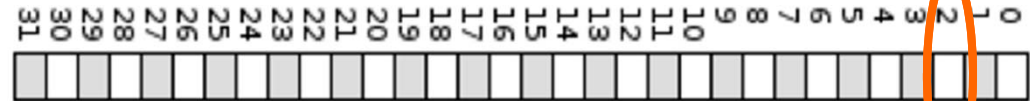
```
ldr    r0, =STBASE
ldr    r2, [r0, #STCS]
ands   r2, #0b0010 @C1?
...
ldr    r2, [r0, #STCS]
ands   r2, #0b1000 @C3?
```

2 Source of interruption?. Check if push-button (2) was pressed

```
ldr    r0, #GPBASE  
ldr    r2, [r0, #GPEDS0]  
ands   r2, #0b00000000000000000000000000000100  
...  

```

GPEDS0 (push-button falling-edge interrupt notification)
20200040 (GPBASE+GPEDS0)



4 Clear event (timer interrupt C1/C3)

(to allow a new interrupt):

```
ldr    r0,=#STBASE
mov    r1,#0b0010 @C1
str    r1,[r0,#STCS]
...
mov    r1,#0b1000 @C3
str    r1,[r0,#STCS]
```

4 Clear event (interrupt by push-button (2))

(to allow a new interrupt):

```
ldr    r0, =GPBASE  
mov    r1, #0b0000000000000000000000000000100  
str    r1, [r0, #GPEDSO]
```

5 Pop registers
pop {r0, r1, r2}

```

6 Return from handler
  subs    pc, lr, #4

```

Using FIQ

Select FIQ Source = 7 bits \rightarrow 128 sources

0-31 represent 32 interruption sources of IRQ 1

32-63 represent 32 interruption sources of IRQ 2

64-95 represent 32 interruption sources of IRQ basic

3F00B20C (INTBASE+INTFIQCON)

[illegible]

- Enable FIQ for C1 of SysTimer
 - Bit 1 of IRQ1 → Code 1
 - Also 1 in FIQ Enable
 - Result: 0b10000001 → 0x81
- Enable FIQ for C3 of SysTimer
 - Bit 3 of IRQ1 → Code 3
 - Also 1 in FIQ Enable
 - Result: 0b10000011 → 0x83
- Enable FIQ for GPIO_int3 (any push button)
 - Bit 20 of IRQ2 → Code 20+32
 - Also 1 in FIQ Enable
 - Result: 0b10110100 → 0xB4

1 Initialize Vector Table (FIQ)

```
mov    r0, #0
ADDEXC 0x1C, irq_handler
```

2 Stack init for FIQ mode

```
mov    r0, #0b11010001
msr    cpsr_c, r0
mov    sp, #0x4000
```

⑥ Enable push-button interruption (FIQ):

```
ldr    r0, =INTBASE
mov    r1, #0b10110100
str    r1, [r0, #INTFIQCON]
```

6 Enable C3 interruption (FIQ):

```
ldr    r0, =INTBASE
mov    r1, #0b10000011
str    r1, [r0, #INTFIQCON]
```

7 Enable FIQ and IRQ (SVC mode):

```
mov    r1, #0b00010011
msr    cpsr_c, r1
```


Example: Timer in IRQ and push button in FIQ

```
.include "inter.inc"
.text
```

```
ADDEXC 0x18, irq_handler
ADDEXC 0x1c, fiq_handler
```

```
mov r0, #0b11010001
msr cpsr_c, r0
mov sp, #0x4000
```

Stack init for FIQ mode

```
mov r0, #0b11010010
msr cpsr_c, r0
mov sp, #0x8000
```

Stack init for IRQ mode

```
mov r0, #0b11010011
msr cpsr_c, r0
mov sp, #0x8000000
```

Stack init for SVC mode

```
ldr r0, =GPBASE
mov r1, #0b00001000000000000000000000000000
str r1, [r0, #GPFSEL0]
```

Set GPIO9 as output

```
mov r1, #0b00000000000000000000000000000100
str r1, [r0, #GPFEN0]
```

Enable FE ints through GPIO2

```
ldr r0, =STBASE
ldr r1, [r0, #STCLO]
add r1, #0x400000
str r1, [r0, #STC1]
```

Program timer to interrupt in 4 seconds

```
ldr r0, =INTBASE
mov r1, #0b00000010
str r1, [r0, #INTENIRQ1]
```

Enable C1 interruption

```
mov r1, #0b10110100
str r1, [r0, #INTFIQCON]
```

Enable FIQ for GPIO_int3

```
mov r0, #0b00010011
msr cpsr_c, r0
```

Set SVC mode with FIQ and IRQ enabled

```
bucle: b bucle
```


Example: IRQ and FIQ handlers

```
fiq_handler:
```

push {r0, r1, r2}

```
ldr    r0, =GPBASE
```

```
ldr    r1, =onoff
```

```
ldr    r2, [r1]
```

```
eors    r2, #1
```

```
str    r2, [r1]
```

Update onoff variable
and test if its 0 or 1

```
mov     r1, #0b00000000000000000000000100000000
```

```
streq    r1, [r0, #GPCLR0]
```

```
strne    r1, [r0, #GPSET0]
```

Turn on or off red led

```
mov     r1, #0b00000000000000000000000000000000100
```

```
str    r1, [r0, #GPEDS0]
```

Clear GPIO2 interrupt

```
pop    {r0, r1, r2}
```

```
subs    pc, lr, #4
```

```
irq_handler:
```

push {r0, r1, r2}

```
ldr    r0, =GPBASE
```

```
ldr    r1, =onoff
```

```
ldr    r2, [r1]
```

```
eors    r2, #1
```

```
str    r2, [r1]
```

Update onoff variable
and test if its 0 or 1

```
mov     r1, #0b00000000000000000000000100000000
```

```
strne    r1, [r0, #GPSET0]
```

```
streq    r1, [r0, #GPCLR0]
```

Turn on or off red led

```
ldr    r0, =STBASE
```

```
mov    r1, #0b0010
```

```
str    r1, [r0, #STCS]
```

Clear timer interrupt

```
ldr    r1, [r0, #STCLO]
```

```
add    r1, #0x400000
```

```
str    r1, [r0, #STC1]
```

Program timer to
interrupt in 4 seconds

```
pop    {r0, r1}
```

```
subs    pc, lr, #4
```

A variable

```
onoff: .word 0
```