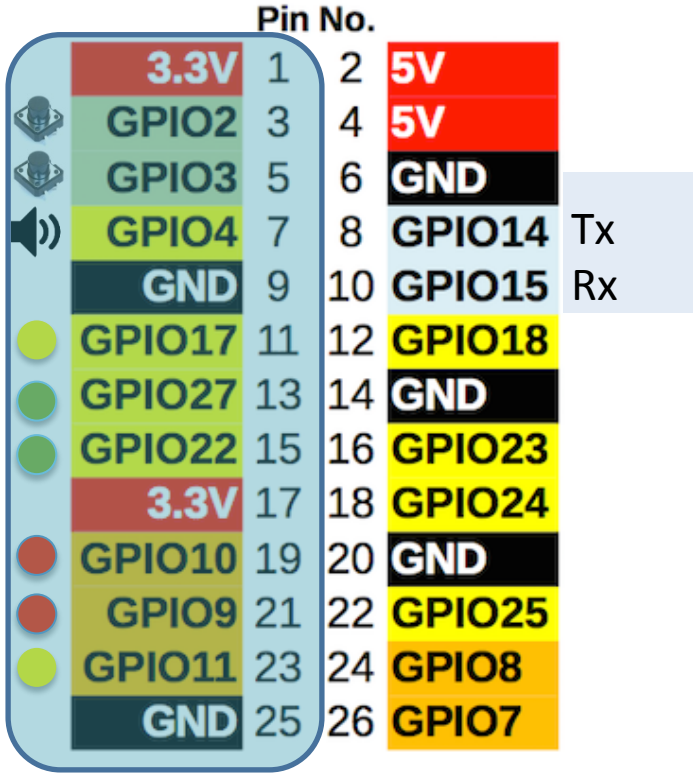
Rpi cheat sheet



Code structure:

Main program (.text):

❶ Initialize Vector Table (IRQ/FIQ)
❷ Init the stack/s for FIQ/IRQ modes
❸ Init the stack for SVC mode (SVC mode selected)
❹ Configure GPIOs (I&O)
❺ Configure peripheral interruption: timer/push-buttons)
❻ Local enabling of configured interrupts
❼ Global enabling of interrupts (SVC mode)
❽ Infinite loop (polling of device/s?)

IRQ/FIQ Handler:

❶ Push registers to be used
❷ Source of interruption?
❸ Perform handler work depending on ❷
❹ Clear event (notify to device IRQ/FIQ has been served)
❺ Pop registers
❻ Return from handler

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

GPFSEL0-5 (3F20 0000)

1. Configure: GPIO9 as Output

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

X	FSEL9	FSEL8	FSEL7	FSEL6	FSEL5	FSEL4	FSEL3	FSEL2	FSEL1	FSEL0
	001									

GPSET0-1 (3F20 001C)

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

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

GPCLR0-1 (3F20 0028)

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

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

3. Read: GPIO2

GPLEV0-1 (3F20 0034)

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

Rpi 3: change HYP mode to SVC mode

Step 0

```
.text
mrs     r0,cpsr
mov     r0, #0b11010011    @ Modo SVC, FIQ&IRQ desact
msr     spsr_cxsf,r0
add     r0,pc,#4
msr     ELR_hyp,r0
eret
```

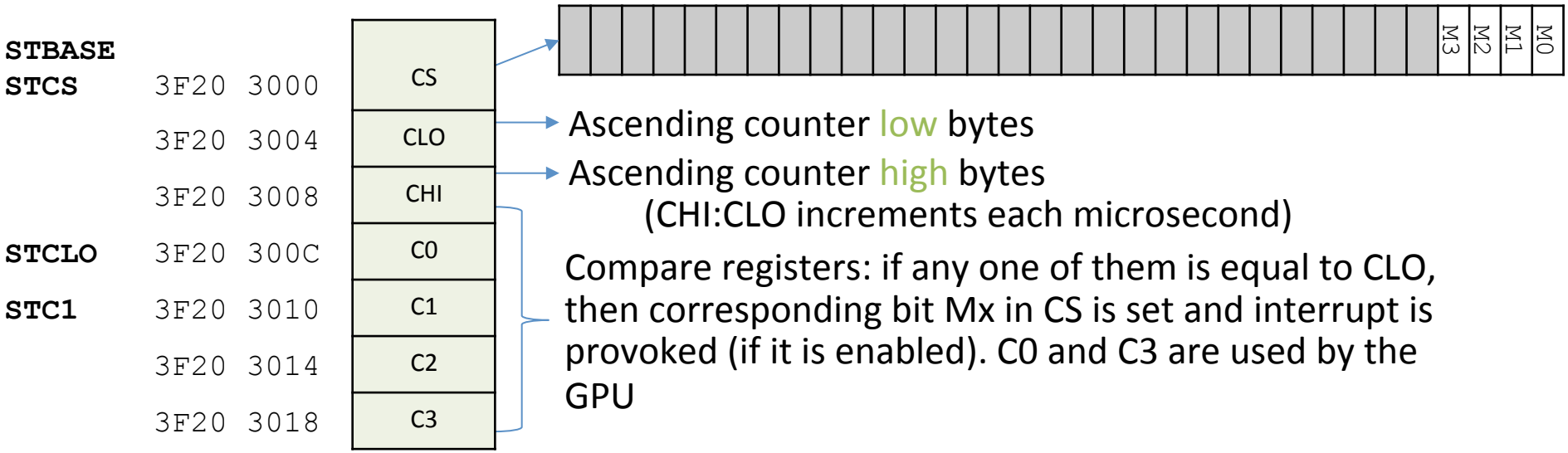
Example: Code for checking push button (GPIO2) and turning led (GPIO9) on

```
        .set GPBASE,  0x3F200000
        .set GPFSEL0, 0x00
        .set GPSET0,  0x1c
        .set GPLEV0,  0x34
.text
        ldr r0, =GPBASE
/* guia bits       xx99988777666555444333222111000 */
        mov r1, #0b00001000000000000000000000000000
        str r1, [r0, #GPFSEL0]
/* mask for testing GPIO2 */
        mov r2, #0b00000000000000000000000000000100
bucle:
        ldr r3, [r0, #GPLEV0]
        tst r3, r2
        bne bucle
/* guia bits      10987654321098765432109876543210
        mov r1, #0b00000000000000000000001000000000
        str r1, [r0, #GPSET0]
infi:  b infi
```

❹ Configure GPIOs (I&O)

❽ 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.

❶ **Initialize Vector Table (IRQ)**

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

❷ **Stack init for IRQ mode**

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

❸ **Stack init for SVC mode**

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

❺ **Configure timer IRQ**

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

❻ **Enable timer interrupt by comparator C1**:

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

❼ **Enable IRQ (SVC mode)**:

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

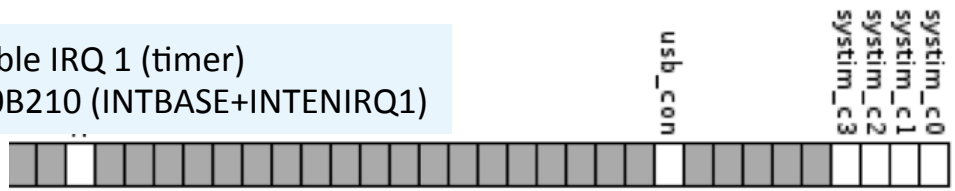
❺ **Configure push-button interruption (GPIO 2)**

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

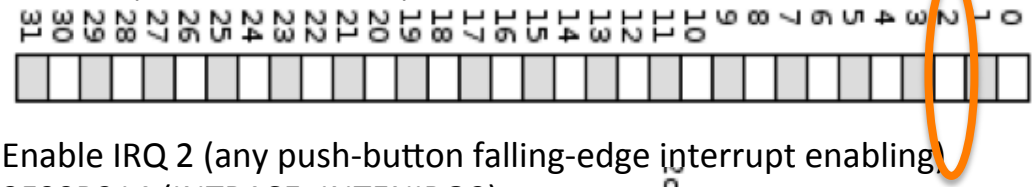
❻ **Enable push-button interruption (IRQ)**:

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

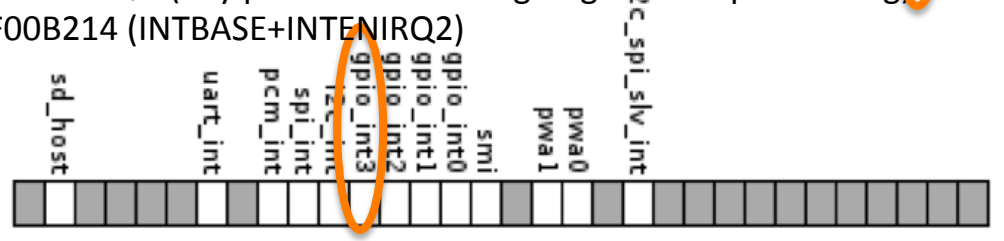
Enable IRQ 1 (timer)
3F00B210 (INTBASE+INTENIRQ1)



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:

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

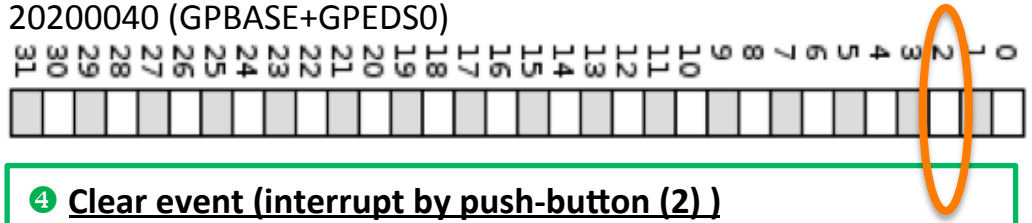
❷ Source of timer interruption?:
```
ldr     r0, =STBASE
ldr     r2, [r0, #STCS]
ands    r2, #0b0010  @C1?
...
ldr     r2, [r0, #STCS]
ands    r2, #0b1000 @C3?
```

❹ **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]
```

Push-button:

❷ 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)



❹ **Clear event (interrupt by push-button (2))**
(to allow a new interrupt):
```
ldr     r0, =GPBASE
mov   r1, #0b00000000000000000000000000000100
str   r1, [r0, #GPEDS0]
```

❺ Pop registers
```
pop    {r0, r1, r2}
```

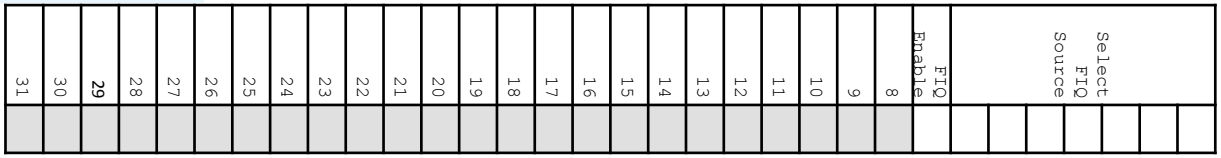
❻ Return from handler
```
subs    pc, lr, #4
```

Using FIQ

Select FIQ Source = 7 bits → 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)

❻ FIQ control



- 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

❶ **Initialize Vector Table (FIQ)**

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

❷ **Stack init for FIQ mode**

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

❻ **Enable push-button interruption (FIQ)**:

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

❻ **Enable C3 interruption (FIQ)**:

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

❼ **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
            mov     r0, #0b11010010
            msr     cpsr_c, r0
            mov     sp, #0x8000
            mov     r0, #0b11010011
            msr     cpsr_c, r0
            mov     sp, #0x8000000
            ldr     r0, =GPBASE
            mov     r1, #0b00001000000000000000000000000000
            str     r1, [r0, #GPFSEL0]
            mov     r1, #0b00000000000000000000000000000100
            str     r1, [r0, #GPFEN0]
            ldr     r0, =STBASE
            ldr     r1, [r0, #STCLO]
            add     r1, #0x400000
            str     r1, [r0, #STC1]

            ldr     r0, =INTBASE
            mov     r1, #0b00000010
            str     r1, [r0, #INTENIRQ1]
            mov     r1, #0b10110100
            str     r1, [r0, #INTFIQCON]
            mov     r0, #0b00010011
            msr     cpsr_c, r0
bucle:      b       bucle
```

Stack init for FIQ mode

Stack init for IRQ mode

Stack init for SVC mode

Set GPIO9 as output

Enable FE ints through GPIO2

Program timer to interrupt in 4 seconds

Enable C1 interruption

Enable FIQ for GPIO_int3

Set SVC mode with FIQ and IRQ enabled

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]
            mov     r1, #0b00000000000000000000001000000000
            streq   r1, [r0, #GPCLR0]
            strne   r1, [r0, #GPSET0]
            mov     r1, #0b00000000000000000000000000000100
            str     r1, [r0, #GPEDS0]
            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]
            mov     r1, #0b00000000000000000000001000000000
            strne   r1, [r0, #GPSET0]
            streq   r1, [r0, #GPCLR0]
            ldr     r0, =STBASE
            mov     r1, #0b0010
            str     r1, [r0, #STCS]
            ldr     r1, [r0, #STCLO]
            add     r1, #0x400000
            str     r1, [r0, #STC1]
            pop     {r0, r1}
            subs    pc, lr, #4
onoff: .word  0
```

Update onoff variable and test if its 0 or 1

Turn on or off red led

Clear GPIO2 interrupt

Update onoff variable and test if its 0 or 1

Turn on or off red led

Clear timer interrupt

Program timer to interrupt in 4 seconds

A variable