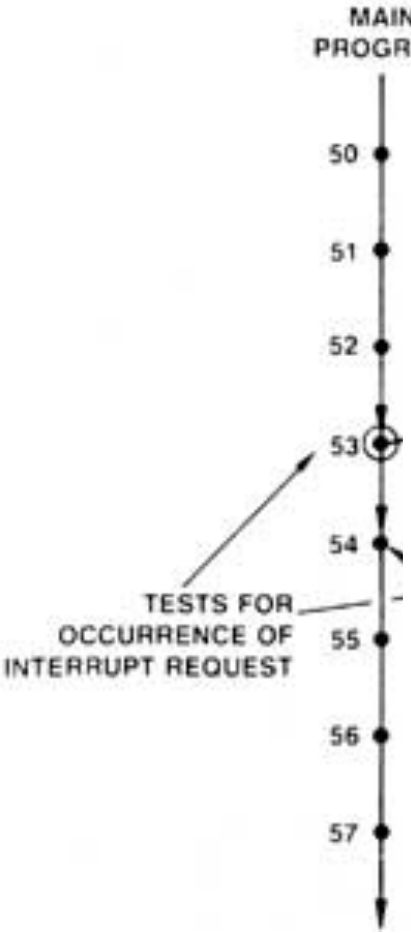
Il seguente materiale e' stato estratto da:

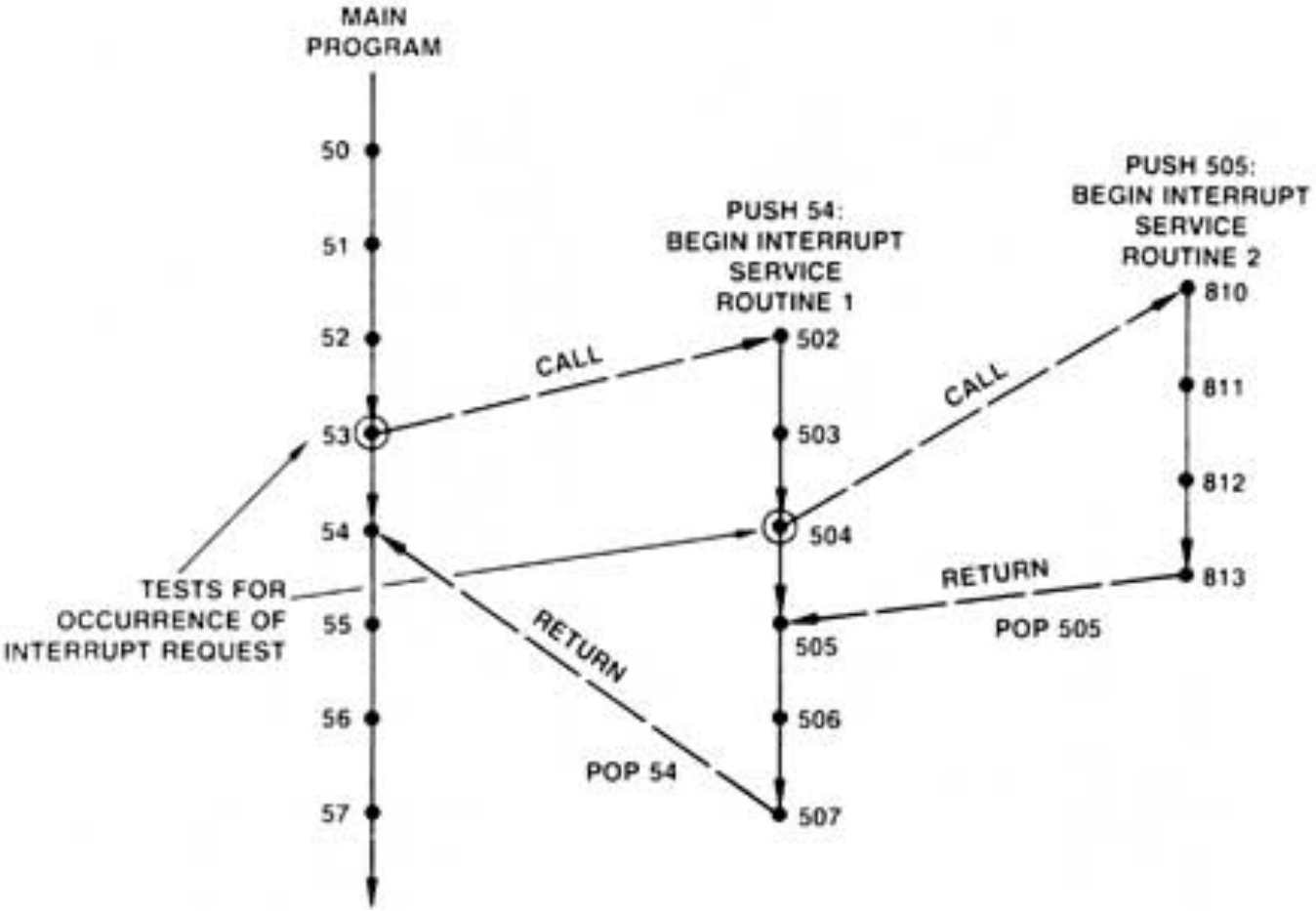
http://processors.wiki.ti.com/index.php/Getting_Started_with_the_TIVA%E2%84%A2_C_Series_TM4C123G_LaunchPad#Workshop_Material

Il meccanismo delle interruzioni e il timer



from:
http://www10.edacafe.com/book/parse_book.php?article=BITSLICE/BIT_CHAP_4/bitCh4C.html

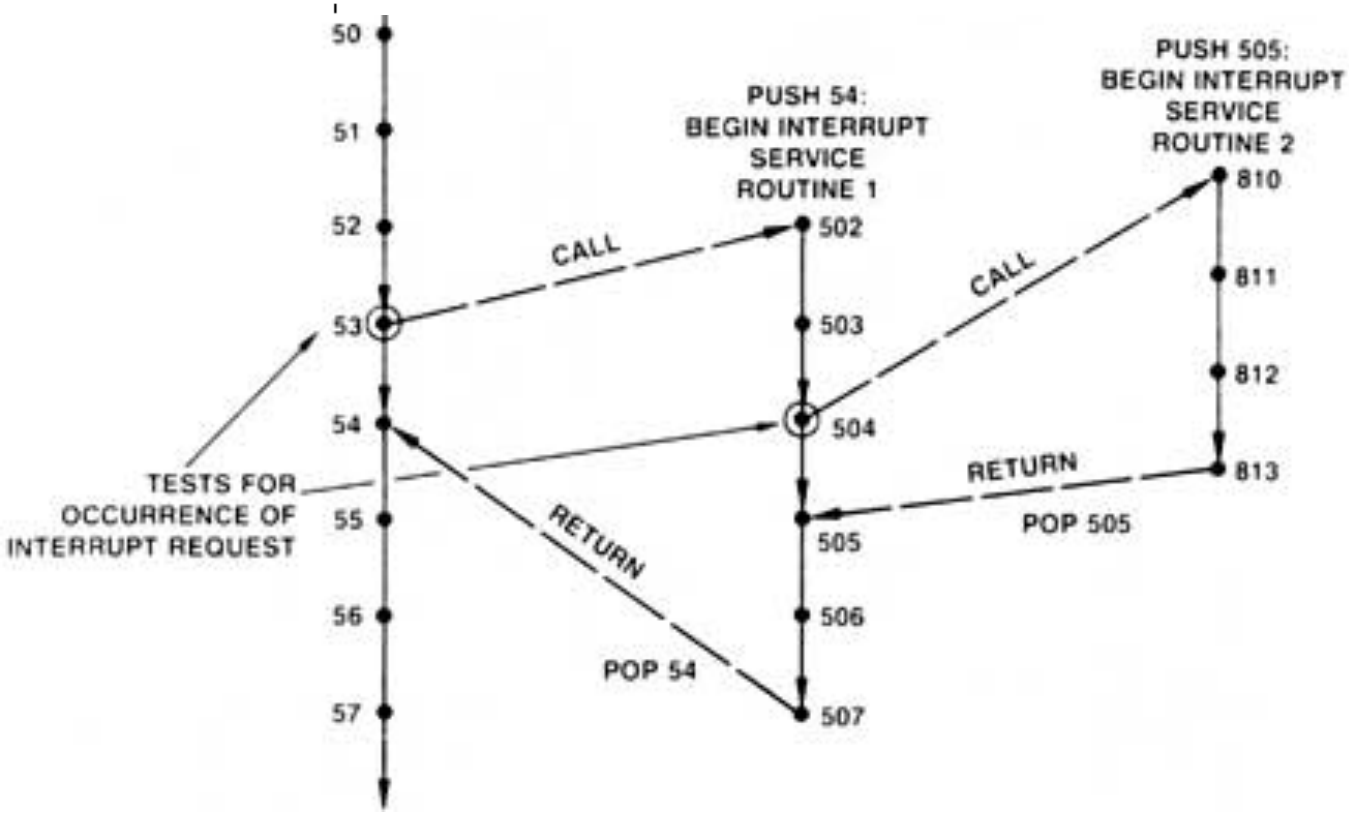
Il meccanismo delle interruzioni e il timer



from:
http://www10.edacafe.com/book/parse_book.php?article=BITSLICE/BIT_CHAP_4/bitCh4C.html

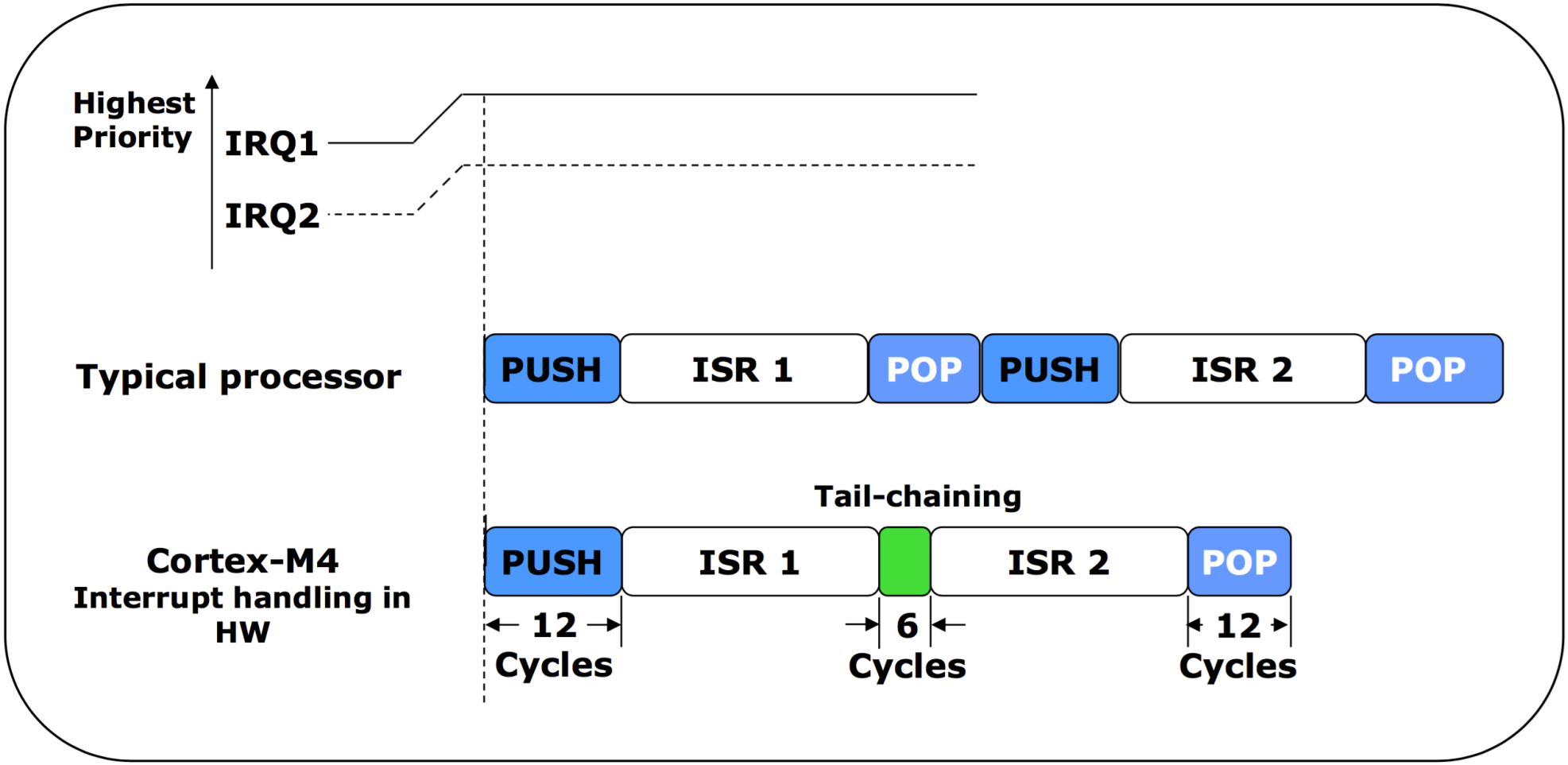
Il meccanismo delle interruzioni e il timer

Typical processor



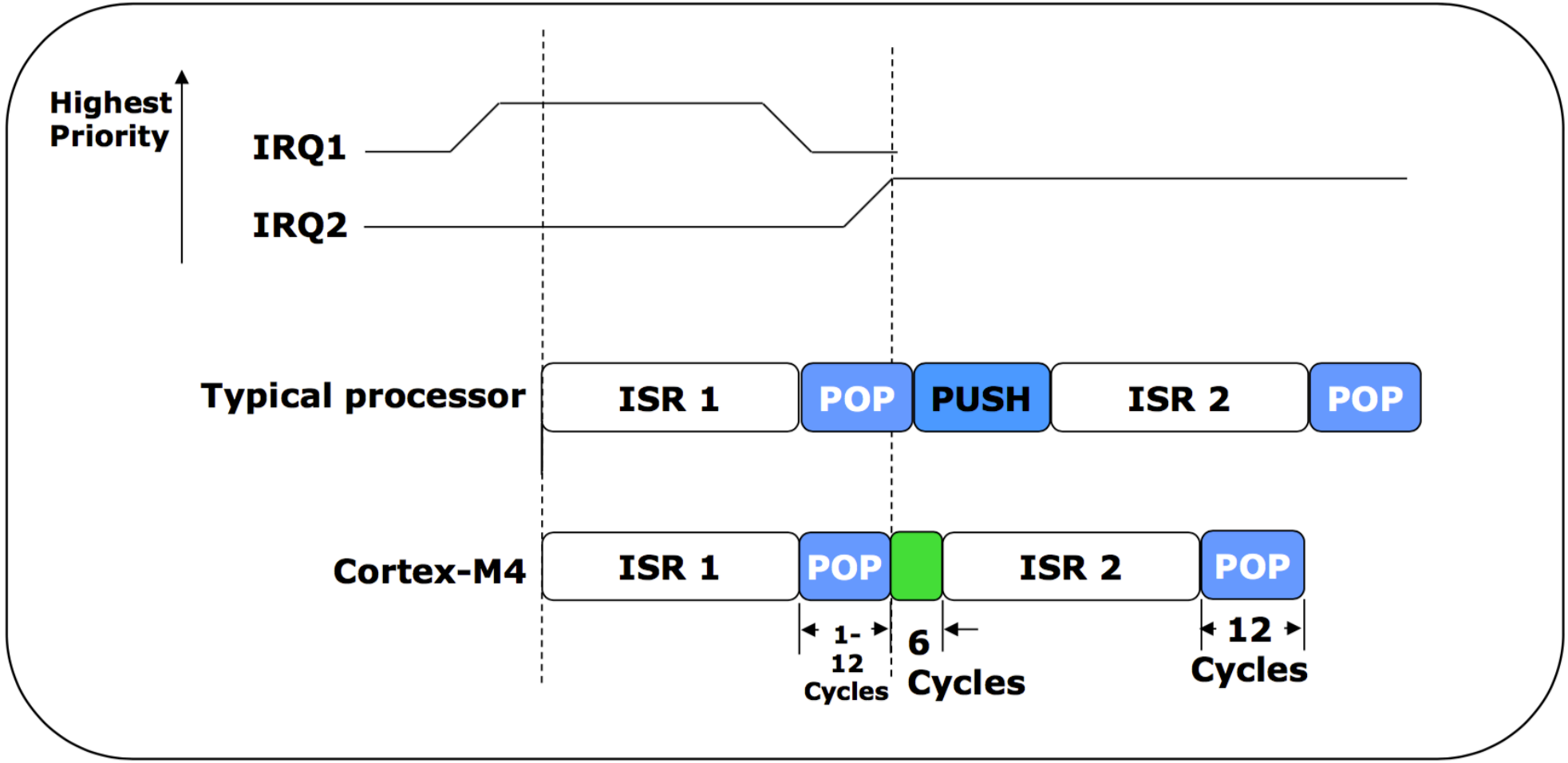
from:
http://www10.edacafe.com/book/parse_book.php?article=BITSLICE/BIT_CHAP_4/bitCh4C.html

Interrupt Latency - Tail Chaining

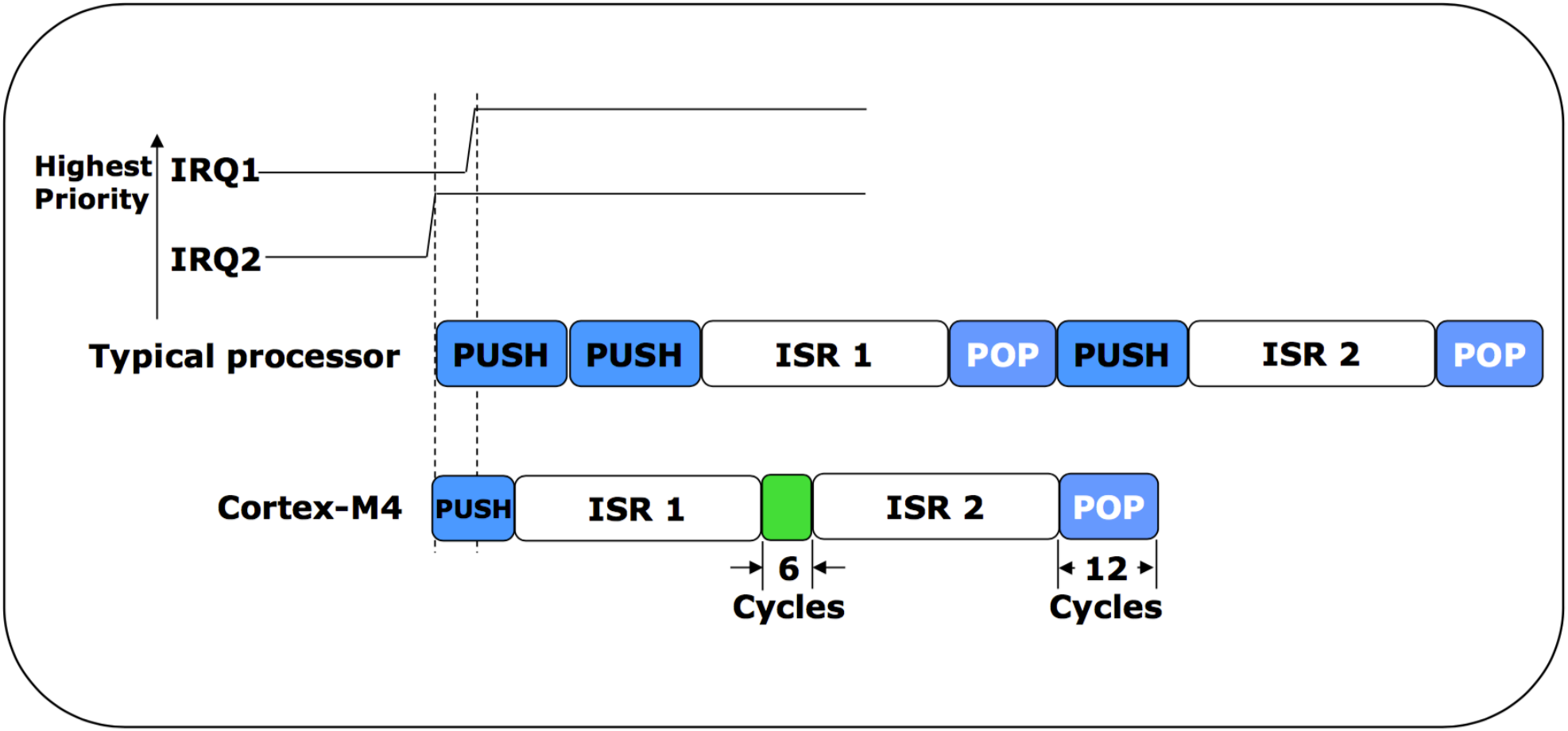


Nested Vectored Interrupt Controller (NVIC)
Interrupt Service Routine (ISR)

Interrupt Latency – Pre-emption



Interrupt Latency – Late Arrival



Cortex-M4® Interrupt Handling

Interrupt handling is automatic. No instruction overhead.

Entry

- Automatically pushes registers R0–R3, R12, LR, PSR, and PC onto the stack
- In parallel, ISR is pre-fetched on the instruction bus. ISR ready to start executing as soon as stack PUSH complete

Exit

- Processor state is automatically restored from the stack
- In parallel, interrupted instruction is pre-fetched ready for execution upon completion of stack POP

PSR: Program status register - contains the exception type number of the current Interrupt Service Routine (ISR))

LR: The **Link Register (LR**) is register R14, and it stores the return information for subroutines, function
calls, and exceptions.

PC: The **Program Counter (PC)** is register R15, and it contains the current program address

Cortex-M4® Exception Types

Vector Number	Exception Type	Priority	Vector address	Descriptions
1	Reset	-3	0x04	Reset
2	NMI	-2	0x08	Non-Maskable Interrupt
3	Hard Fault	-1	0x0C	Error during exception processing
4	Memory Management Fault	Programmable	0x10	MPU violation
5	Bus Fault	Programmable	0x14	Bus error (Prefetch or data abort)
6	Usage Fault	Programmable	0x18	Exceptions due to program errors
7-10	Reserved	-	0x1C - 0x28	
11	SVCall	Programmable	0x2C	SVC instruction
12	Debug Monitor	Programmable	0x30	Exception for debug
13	Reserved	-	0x34	
14	PendSV	Programmable	0x38	
15	SysTick	Programmable	0x3C	System Tick Timer
16 and above	Interrupts	Programmable	0x40	External interrupts (Peripherals)

General Purpose Timer Module

- Six 16/32-bit and Six 32/64-bit general purpose timers
- Twelve 16/32-bit and Twelve 32/64-bit capture/compare/PWM pins
- Timer modes:
  - One-shot
  - Periodic
  - Input edge count or time capture with 16-bit prescaler
  - PWM generation (separated only)
  - Real-Time Clock (concatenated only)
- Count up or down
- Simple PWM (no deadband generation)
- Support for timer synchronization, daisy-chains, a during debugging
- May trigger ADC samples or DMA transfers

Timer	Up/Down Counter	Even CCP Pin	Odd CCP Pin
16/32-Bit Timer 0	Timer A	T0CCP0	-
	Timer B	-	T0CCP1
16/32-Bit Timer 1	Timer A	T1CCP0	-
	Timer B	-	T1CCP1
16/32-Bit Timer 2	Timer A	T2CCP0	-
	Timer B	-	T2CCP1
16/32-Bit Timer 3	Timer A	T3CCP0	-
	Timer B	-	T3CCP1
16/32-Bit Timer 4	Timer A	T4CCP0	-
	Timer B	-	T4CCP1
16/32-Bit Timer 5	Timer A	T5CCP0	-
	Timer B	-	T5CCP1
32/64-Bit Wide Timer 0	Timer A	WT0CCP0	-
	Timer B	-	WT0CCP1
32/64-Bit Wide Timer 1	Timer A	WT1CCP0	-
	Timer B	-	WT1CCP1
32/64-Bit Wide Timer 2	Timer A	WT2CCP0	-
	Timer B	-	WT2CCP1
32/64-Bit Wide Timer 3	Timer A	WT3CCP0	-
	Timer B	-	WT3CCP1
32/64-Bit Wide Timer 4	Timer A	WT4CCP0	-
	Timer B	-	WT4CCP1
32/64-Bit Wide Timer 5	Timer A	WT5CCP0	-
	Timer B	-	WT5CCP1

the code

Voglio accendere e spegnere il led blu con una frequenza di 10 Hz. All'inizio del ciclo il led si accende, a meta' ciclo si spegne e cosi' via. Voglio realizzare questa cosa tramite il meccanismo delle interruzioni.
In particolare, utilizzo un timer che con frequenza 20 Hz, mi fa partire l'interruzione e accende/spegne alternativamente il led blu.

```
include <stdint.h>
#include <stdbool.h>
#include "inc/tm4c123gh6pm.h"
#include "inc/hw_memmap.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/interrupt.h"
#include "driverlib/gpio.h"
#include "driverlib/timer.h"
```

```
int main(void)
{
  uint32_t ui32Period;

SysCtlClockSet(SYSCTL_SYSDIV_5|SYSCTL_USE_PLL|SYSCTL_XTAL_16MHZ|SYSCTL_OSC_MA
IN);
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
  GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3);
```

abilito il Timer0

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
```

abilito il Timer0

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
```

e lo configuro in maniera che ricominci a contare in maniera periodica una volta che ha finito il conteggio

```
TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
```

abilito il Timer0

```
SysCtlPeripheralEnable(SYSCTL_PERIPH_TIMER0);
```

e lo configuro in maniera che ricominci a contare in maniera periodica una volta che ha finito il conteggio

```
TimerConfigure(TIMER0_BASE, TIMER_CFG_PERIODIC);
```

Voglio che la frequenza sia di 20 Hz, quindi prendo clock rate del processore attraverso il comando SysCtlClockGet e lo divido per (10*2).

```
ui32Period = (SysCtlClockGet() / 10) / 2;
```

Setto il numero di conteggio massimo del timer

```
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
```

Setto il numero di conteggio massimo del timer

```
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
```

e abilito le interruzioni. In particolare, abilito il vettore specifico associato al Timer0A

```
IntEnable(INT_TIMER0A);
```

Setto il numero di conteggio massimo del timer

```
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
```

e abilito le interruzioni. In particolare, abilito il vettore specifico associato al Timer0A

```
IntEnable(INT_TIMER0A);
```

faccio partire l'interruzione alla fine del conteggio

```
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
```

Setto il numero di conteggio massimo del timer

```
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
```

e abilito le interruzioni. In particolare, abilito il vettore specifico associato al Timer0A

```
IntEnable(INT_TIMER0A);
```

faccio partire l'interruzione alla fine del conteggio

```
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
```

e infine abilito tutte le interruzioni

```
TimerEnable(TIMER0_BASE, TIMER_A);
```

Setto il numero di conteggio massimo del timer

```
TimerLoadSet(TIMER0_BASE, TIMER_A, ui32Period -1);
```

e abilito le interruzioni. In particolare, abilito il vettore specifico associato al Timer0A

```
IntEnable(INT_TIMER0A);
```

faccio partire l'interruzione alla fine del conteggio

```
TimerIntEnable(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
```

e infine abilito tutte le interruzioni

```
TimerEnable(TIMER0_BASE, TIMER_A);
```

e aspetto.....

```
while(1) {
  }
```

Definisco l'Handler

```
void Timer0IntHandler(void)
{
  // Clear the timer interrupt
  TimerIntClear(TIMER0_BASE, TIMER_TIMA_TIMEOUT);
  // Read the current state of the GPIO pin and
  // write back the opposite state
  if(GPIOPinRead(GPIO_PORTF_BASE, GPIO_PIN_2))
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3, 0);
  else
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 4);
}
```

In tm4c123gh6pg3_startup_ccs.c, sostituisco la linea

```
IntDefaultHandler                    //Timer 0 subtimer A
```

con la linea

```
Timer0IntHandler                     //Timer 0 subtimer A
```

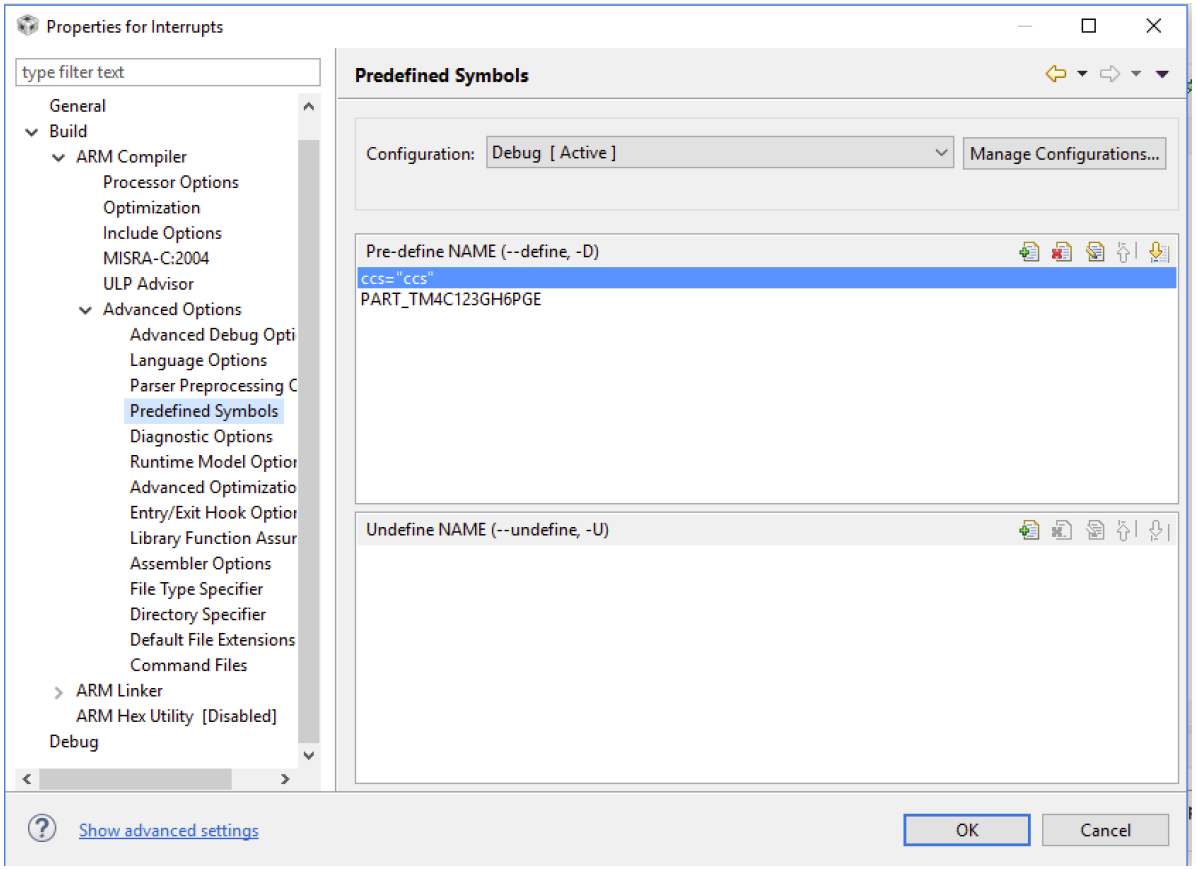
e sotto la linea

```
extern void _c_int00(void);
```

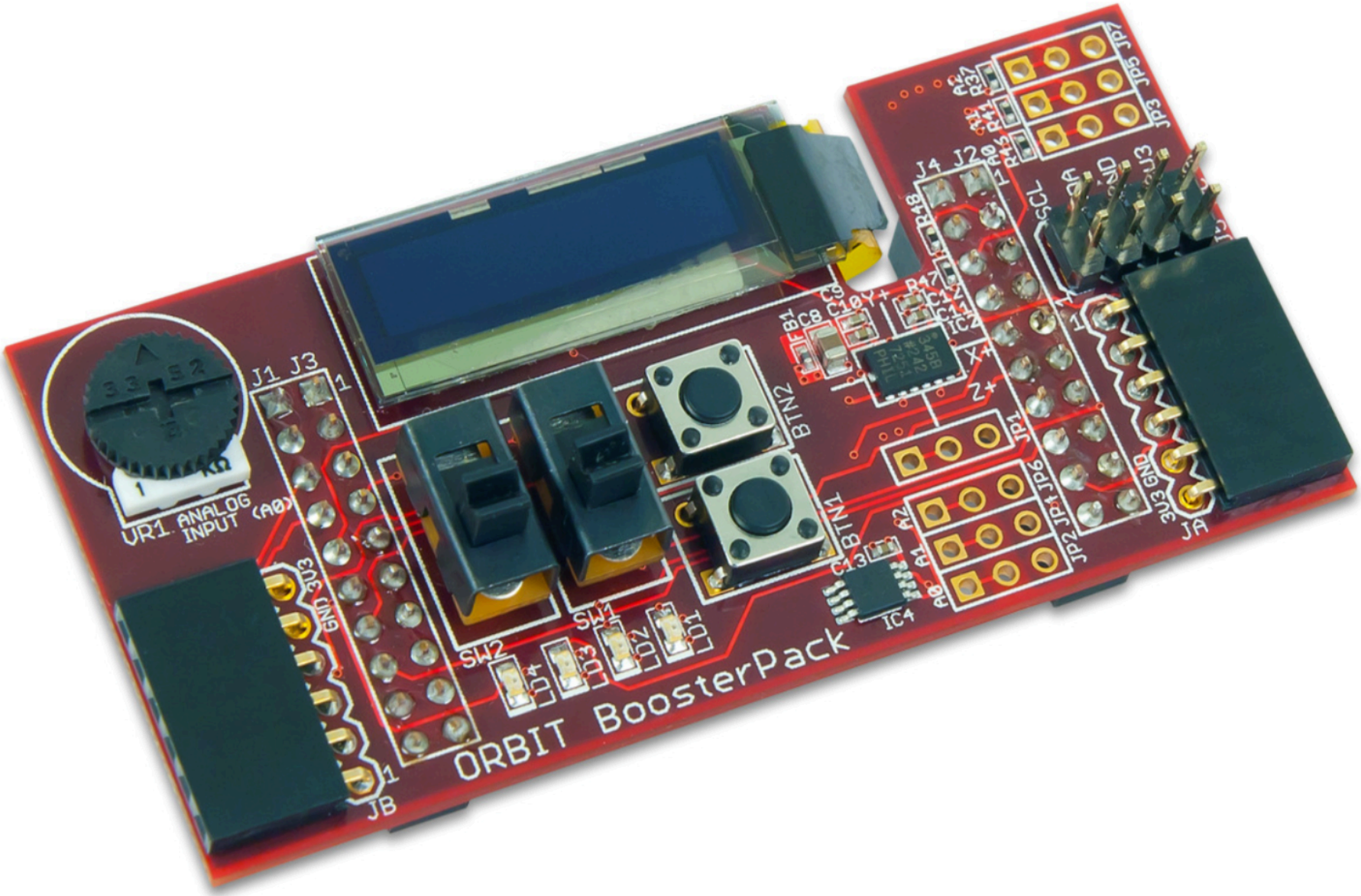
aggiungo

```
extern void Timer0IntHandler(void);
```

Controllare inoltre che nelle proprieta' del progetto, nel menu' ARM compiler, Advanced Options, Predefined Symbols, sia presente la variabile: PART_TM4C123GH6PGE, come sotto



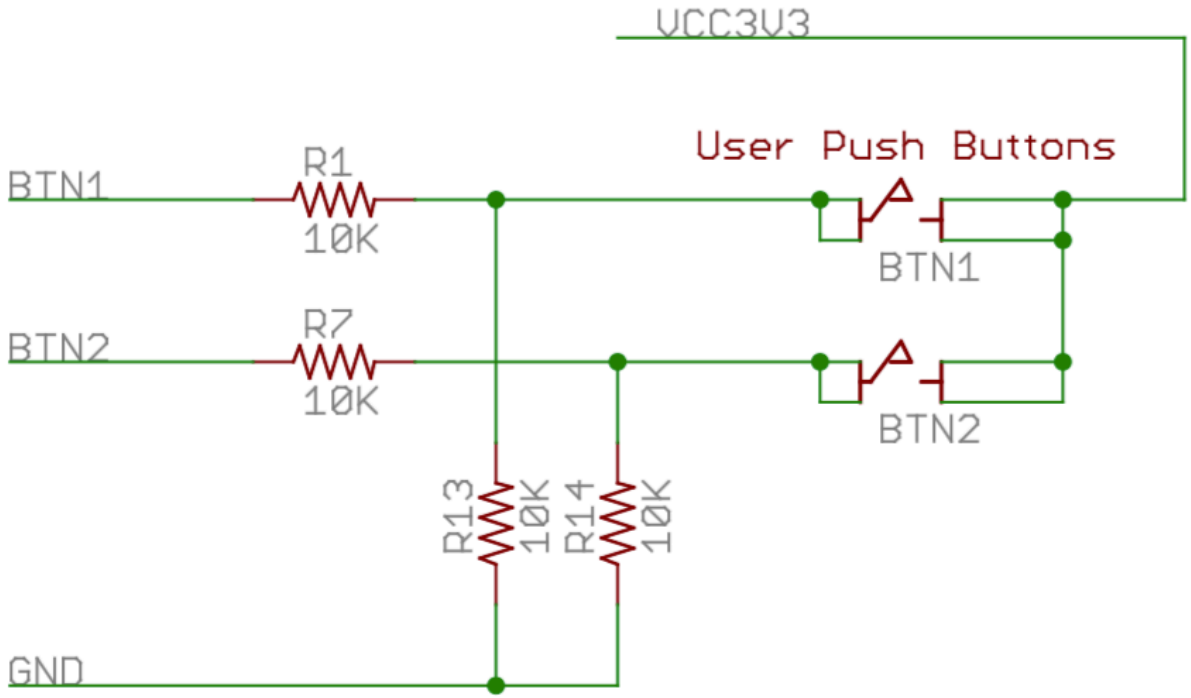
Scheda Orbit


ORBIT BoosterPack
VR1 ANALOG INPUT (A0)
J1 J3
JB
JA
BTN1
BTN2
SW1
SW2
LD1
LD2
LD3
LD4
IC4
IC2
JP1
JP2
JP3
JP4
JP5
JP6
J2
J4
SCL
SDA
GND
3U3

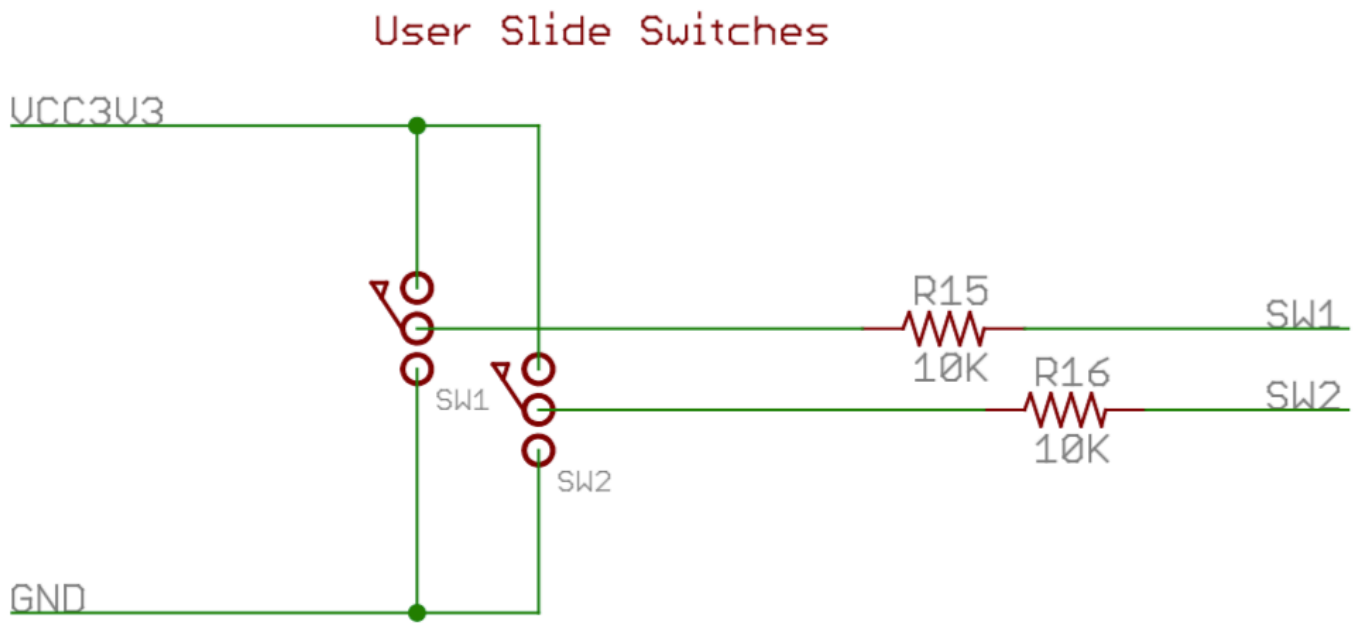
Features Include:

- two 1x6 Digilent Pmod™ connectors
- 3-axis accelerometer
- 256 Kbit I^2C EEPROM
- I^2C temperature sensor
- 128x32 pixel OLED display
- analog potentiometer

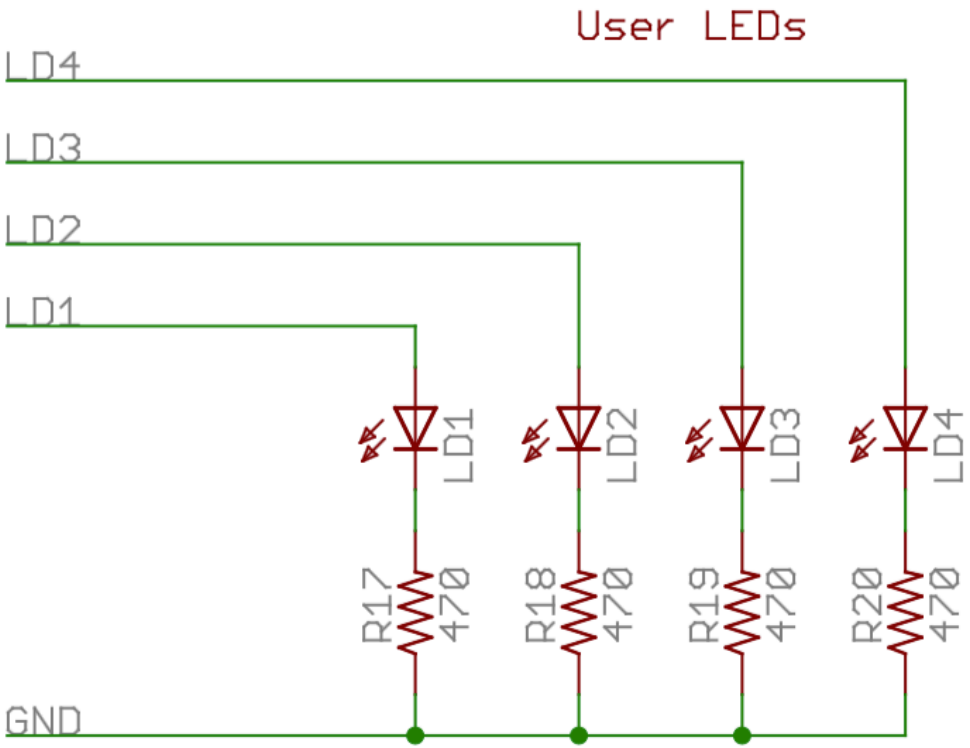
Pushbuttons: There are two pushbutton switches labeled BTN1 and BTN2. A read to the corresponding GPIODATA register bits will return a ‘0’ when the button is released and a ‘1’ when the button is pressed.



Slide Switches: There are two slide switches labeled SW1 and SW2. A read to the corresponding GPIODATA register bits will return a ‘0’ when a switch is down (toward the LEDs) and a ‘1’ when a switch is up (toward the OLED display).

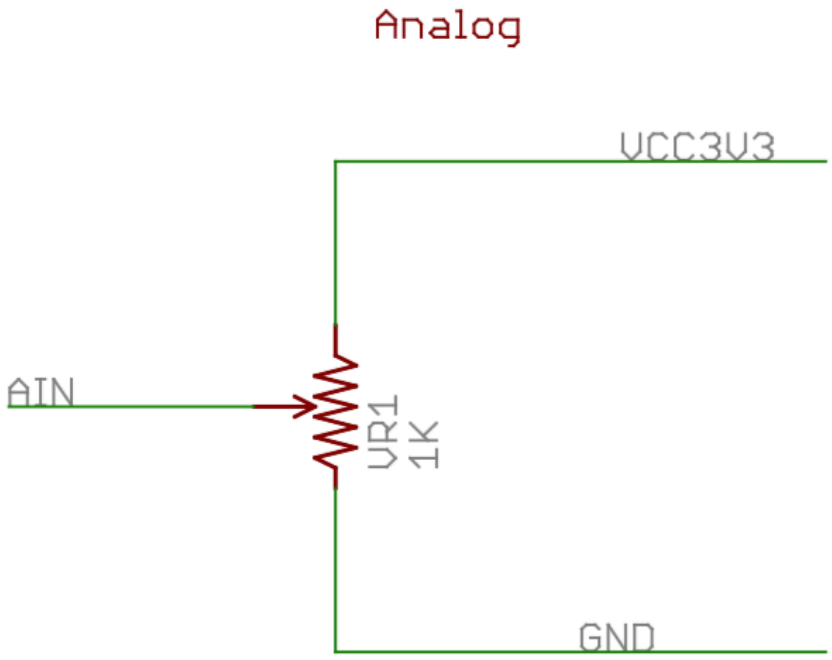


LEDs: There are four LEDs, labeled LD1 – LD4. An LED will be illuminated when the corresponding GPIODATA register bit is set to a '1' (given the corresponding direction bit has been set in the GPIODIR registers) and off when set to a '0'.

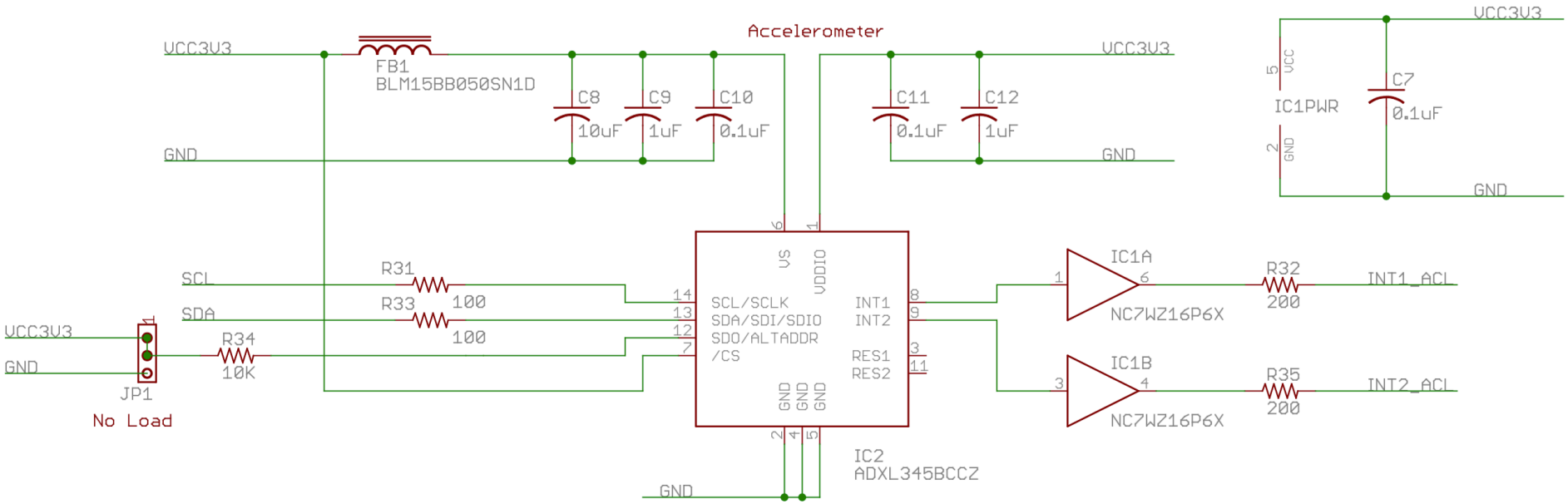


Potentiometer

A potentiometer (pot) is provided on the board to be used as an analog signal source or analog control input. The pot is a 1 K-ohm trimmer pot connected between the VCC3V3 supply and ground. The wiper of the pot is connected to analog input AIN0.



Accelerometer: A 3-axis accelerometer is provided using an Analog Devices ADXL345. This accelerometer, IC2, is located to the right of BTN2. The silkscreen shows the axis configuration.
The 7 bit I2C device address for the accelerometer is '0011101', or 0x1D.
The ADXL345 has two output pins for setting configurable interrupts, which include activity, inactivity, and DATA_READY.
For complete technical documentation on the ADXL345, refer to the data sheet available at www.analog.com.



Connector #	Pin #	Port and Bit	Function	Description	Notes
J1	1	-	VCC3V3	Power supply	
J1	2	PB5	LD4	User LED	
J1	3	PB0	JB_03/RX	Pmod connector B pin 3/Rx	UART1
J1	4	PB1	JB_02/TX	Pmod connector B pin 2/Tx	UART1
J1	5	PE4	INT2_ACL	Accelerometer Interrupt Output	
J1	6	PF5	/RES_OLED	OLED reset	
J1	7	PB4	INT1_ACL	Accelerometer Interrupt Output	
J1	8	PA5	JA_02/MOSI	Pmod Connector A pin 2/Serial Out	SSI0
J1	9	PA6	SW2	Slide switch	
J1	10	PA7	SW1	Slide switch	
J2	1	-	GND	Ground	
J2	2	PB2	SCL	I^2C clock	I2C0
J2	3	PE0	BTN2	Push button	
J2	4	-	-	-	Not connected
J2	5	-	-	-	Not connected
J2	6	-	-	-	Not connected
J2	7	-	-	-	Not connected
J2	8	PA4	JA_03/MISO	Pmod Connector A pin 3/Serial In	SSI0
J2	9	PA3	JA_01/SS	Pmod Connector A pin 1/Slave Select	SSI0
J2	10	PA2	JA_04/SCK	Pmod Connector A pin 4/Serial Clock	SSI0
J3	1	-	-	-	Not connected
J3	2	-	GND	Ground	

J3	3	PD0	SCK_OLED	OLED serial clock	SSI3/SSI1
J3	4	PD1	/CS_OLED	OLED chip select	SSI3/SSI1
J3	5	PD2	BTN1	Push button	
J3	6	PD3	SDI_OLED	OLED serial data in	SSI3/SSI1
J3	7	PF1	VBAT_OLED	OLED VBAT enable	
J3	8	PE2	VDD_OLED	OLED VDD enable	
J3	9	PE3	AIN	Potentiometer	AIN0
J3	10	-	-	-	Not connected
J4	1	-	-	-	Not connected
J4	2	-	-	-	Not connected
J4	3	PB3	SDA	I^2C data	I2C0
J4	4	PF0	JB_04/RTS	Pmod connector B pin 4/Request to Send	UART1
J4	5	PF1	JB01/CTS	Pmod connector B pin 1/Clear to Send	UART1
J4	6	PC6	LD1	User LED	
J4	7	PC7	LD2	User LED	
J4	8	PD6	LD3	User LED	
J4	9	PD7	/DC_OLED	OLED data/command select	
J4	10		-		Not connected