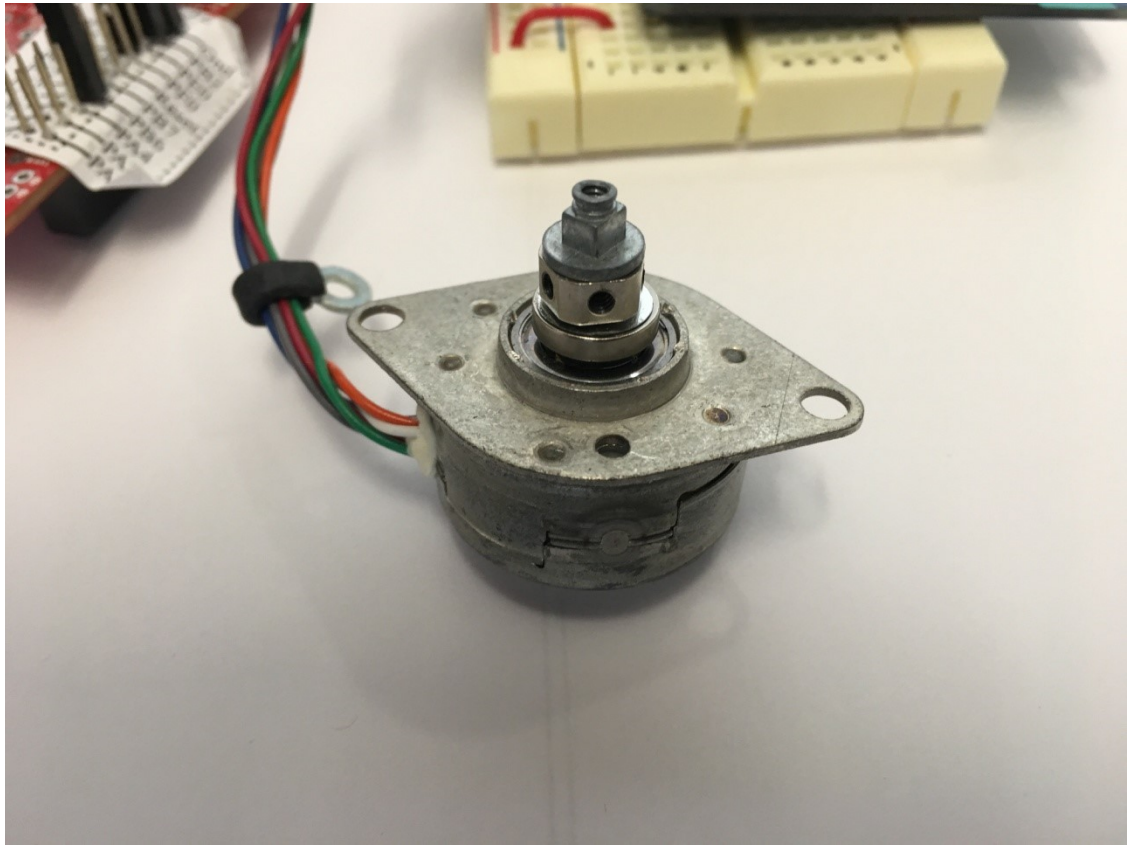
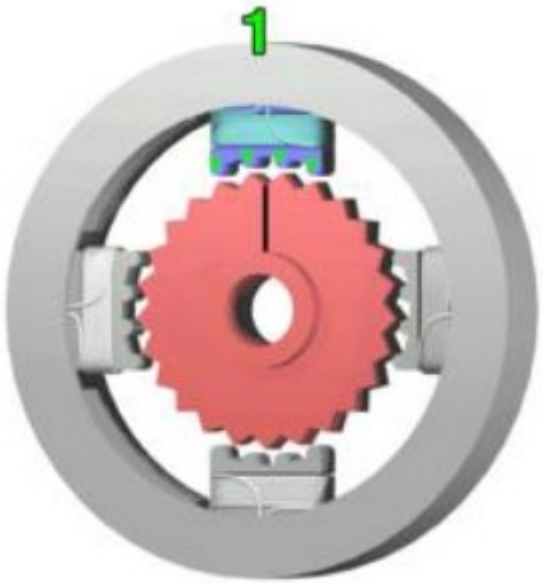
Introduction to Embedded Systems

CSE 211

Stepper Motor

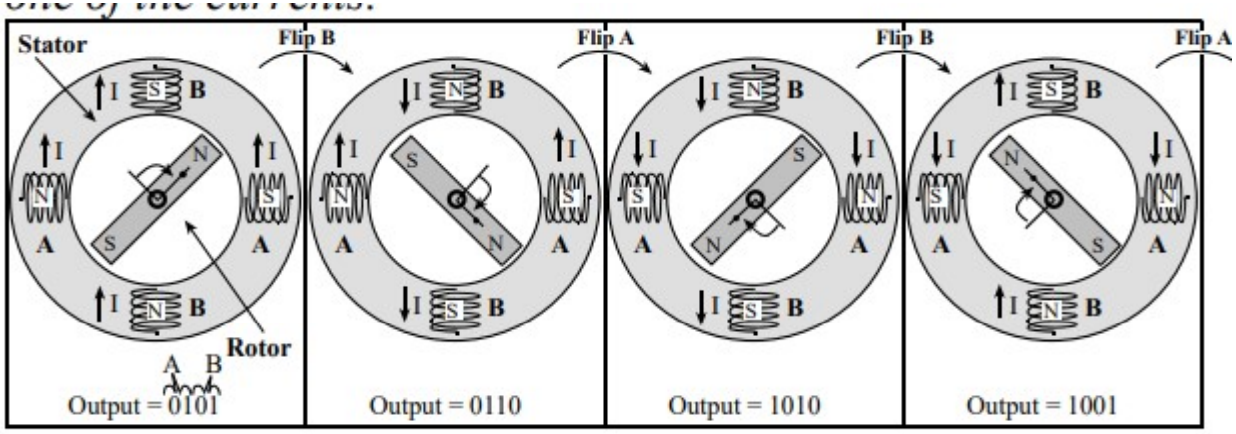


Stepper Motor

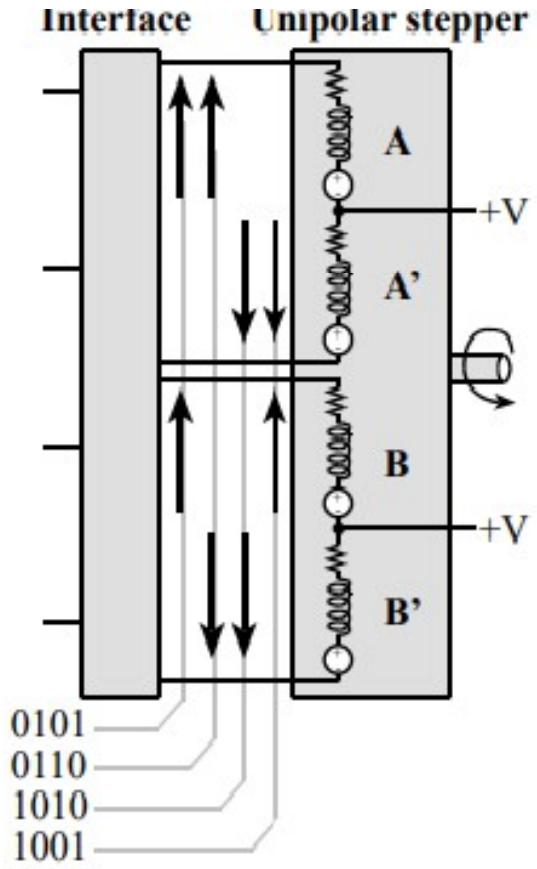


Steps / rev

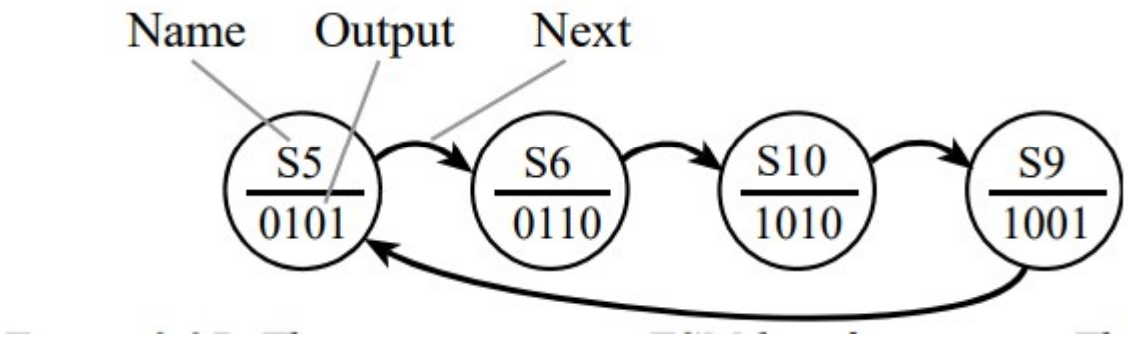
Stepper Motor



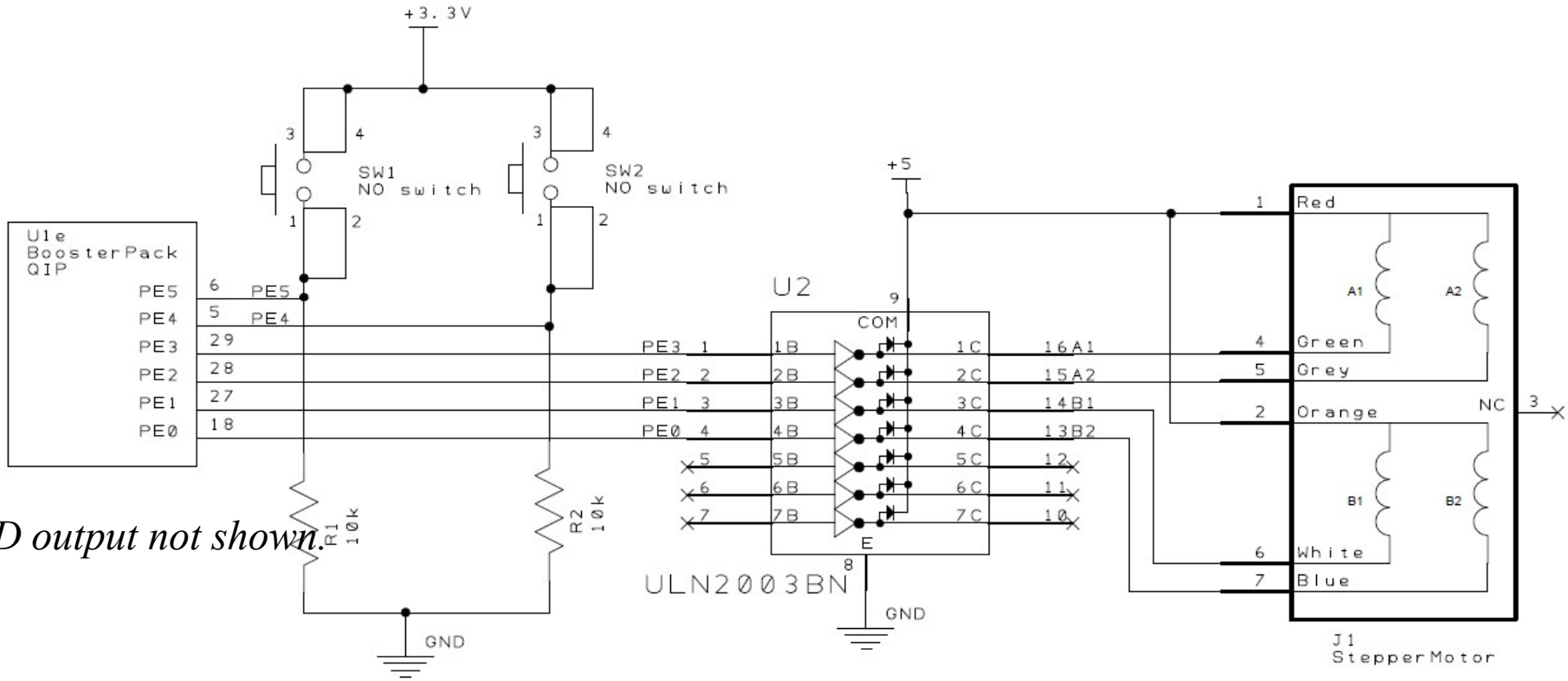
Stepper Motor



Stepper Motor FSM

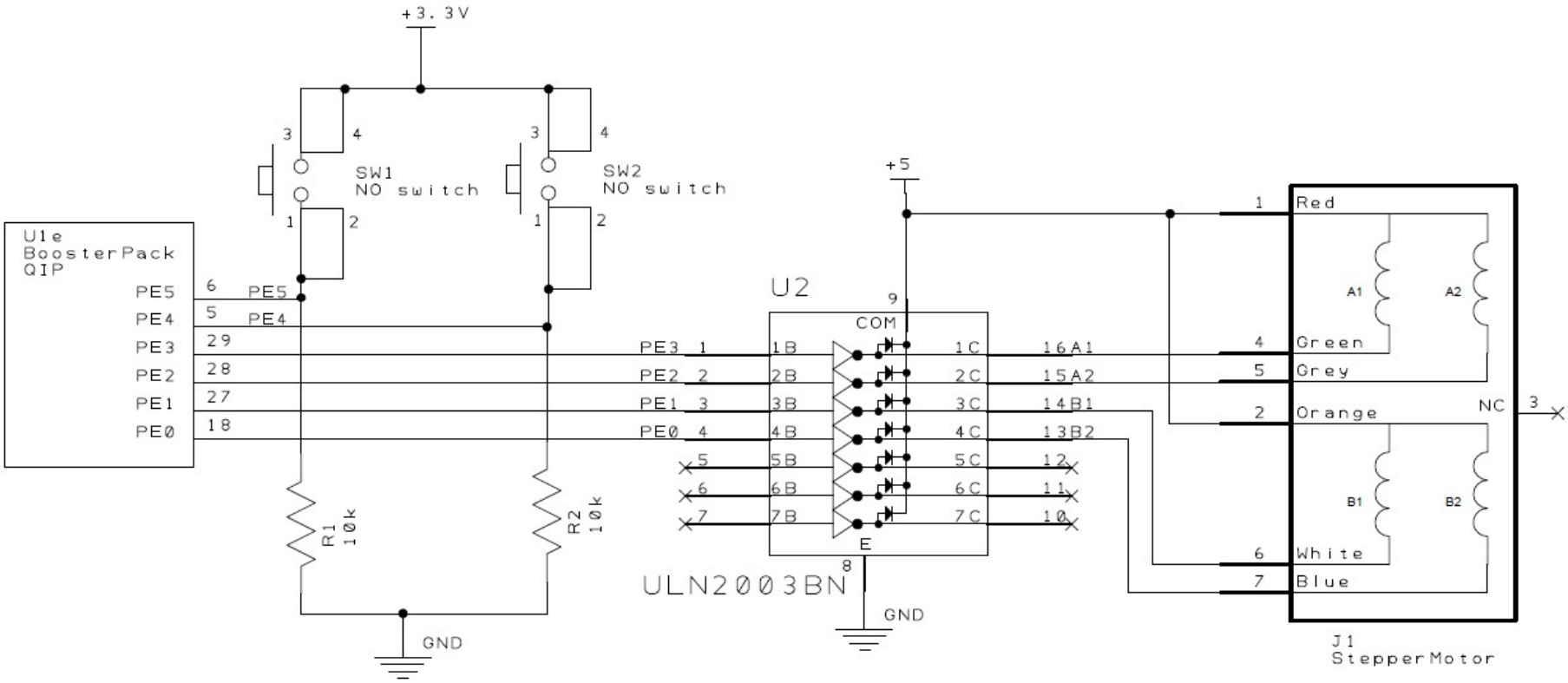


Stepper Motor Interface



LED output not shown.

Stepper Motor Interface



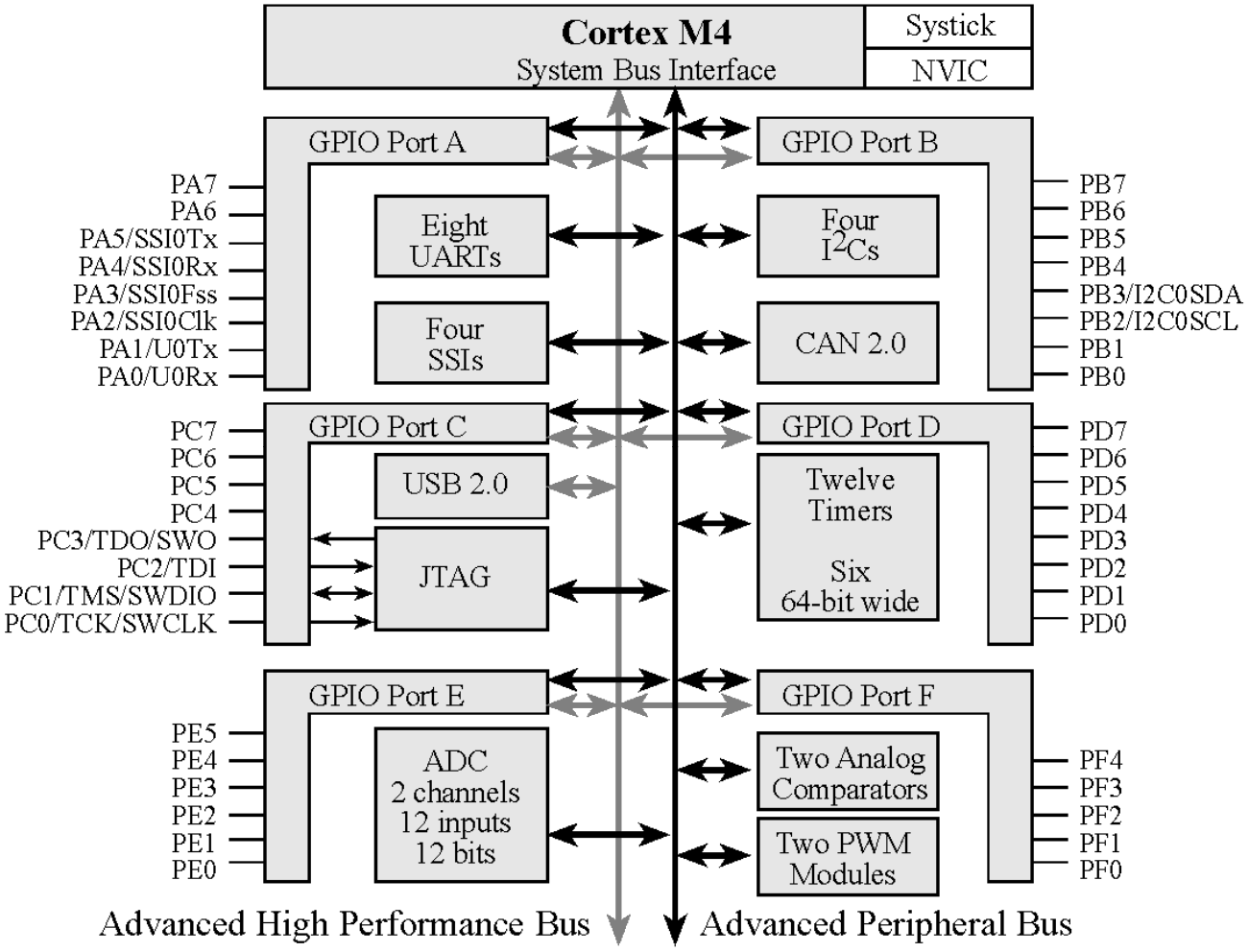
Clockwise 5,6,10,9,5,6,10,9,5,6,10,9,5,6,10,9,5,6,10,9,5,6,10,9,5,6,10,9,…

36 steps/revolution means each step changes angle by 10 degrees

Analog Watch Example

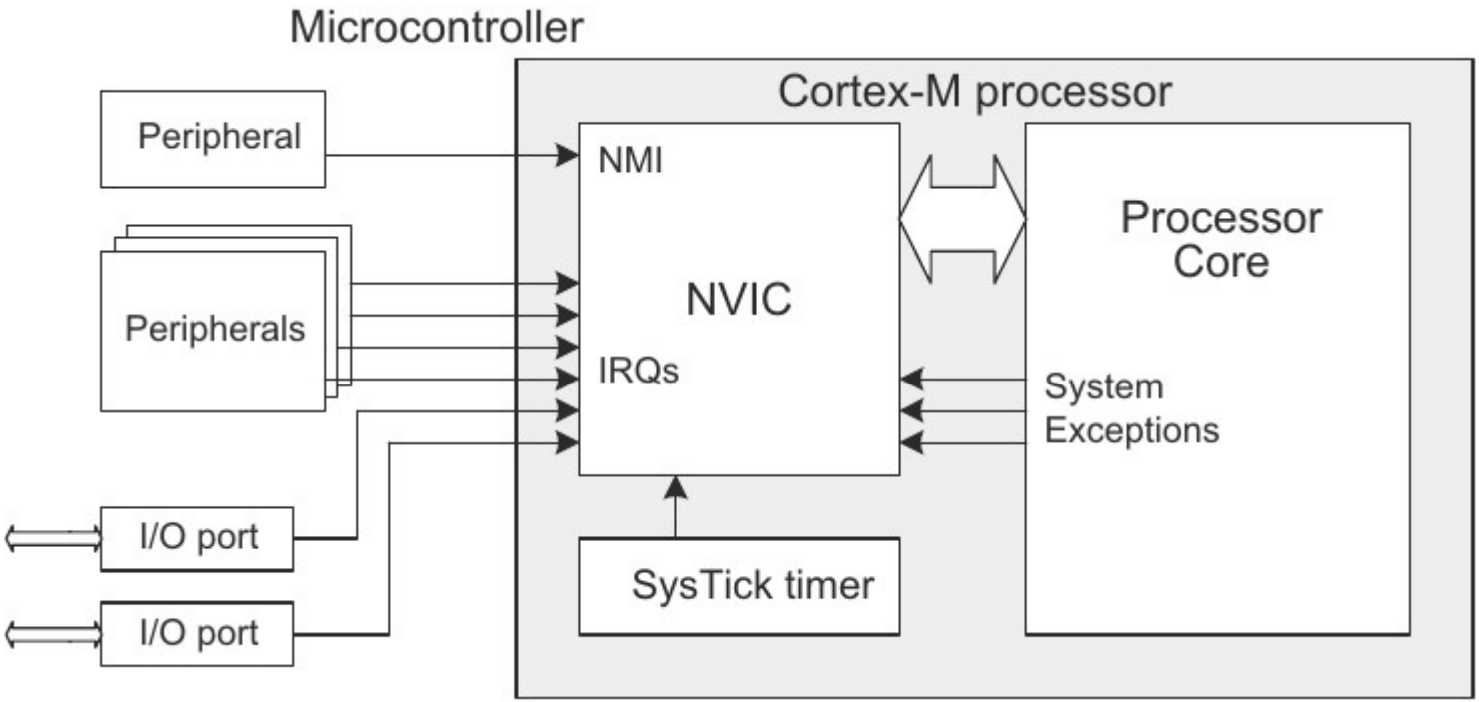
```
void SysTick_Wait(uint32_t delay){
        NVIC_ST_RELOAD_R = delay -1;
        NVIC_ST_CURRENT_R =0;
        while ((NVIC_ST_CTRL_R & 0x00010000) ==0){}
}
void SysTick_wait10ms(uint32_t delay){
                uint32_t i;
                for(i=0; i<delay;i++){ SysTick_Wait(800000);}
};
#define STEPPER (*((volatile unit32_t *) 0x4000703C))
        int main(void)
{
                int i,j;
                SYSCTL_RCGCGPIO_R |=0x08;
                GPIO_PORTD_DIR_R |=0xF;
                GPIO_PORTD_DEN_R |=0xF;
                while(1){
                                STEPPER = 10;
                                SysTick_wait10ms(5);
                                STEPPER = 9;
                                SysTick_wait10ms(5);
                                STEPPER = 5;
                                SysTick_wait10ms(5);
                                 STEPPER = 6;
                                 SysTick_wait10ms(5);}
}
```

Texas Instruments TM4C123



ARM Cortex-M4
+ 256K Flash
+ 32K RAM
+ JTAG
+ 43 Ports
+ SysTick
+ ADC
+ UART

ARM Cortex-M Interrupts



ARM Cortex-M Interrupts

- Each interrupt source has a separate **arm** bit
  - Set for those devices from which it wishes to accept interrupts,

```
GPIO_PORTF_IM_R |= 0x10;      // arm interrupt on PF4
```

  - Deactivate in those devices from which interrupts are not allowed
- Each interrupt source has a separate **flag** bit
  - hardware sets the flag when it wishes to request an interrupt

```
GPIO_PORTF_ICR_R = 0x10;      // acknowledge flag4
```

  - software clears the flag in ISR to signify it is processing the request
- Interrupt **enable** conditions in processor
  - Global interrupt enable bit, I, in PRIMASK register

```
EnableInterrupts();
```

Interrupt Processing

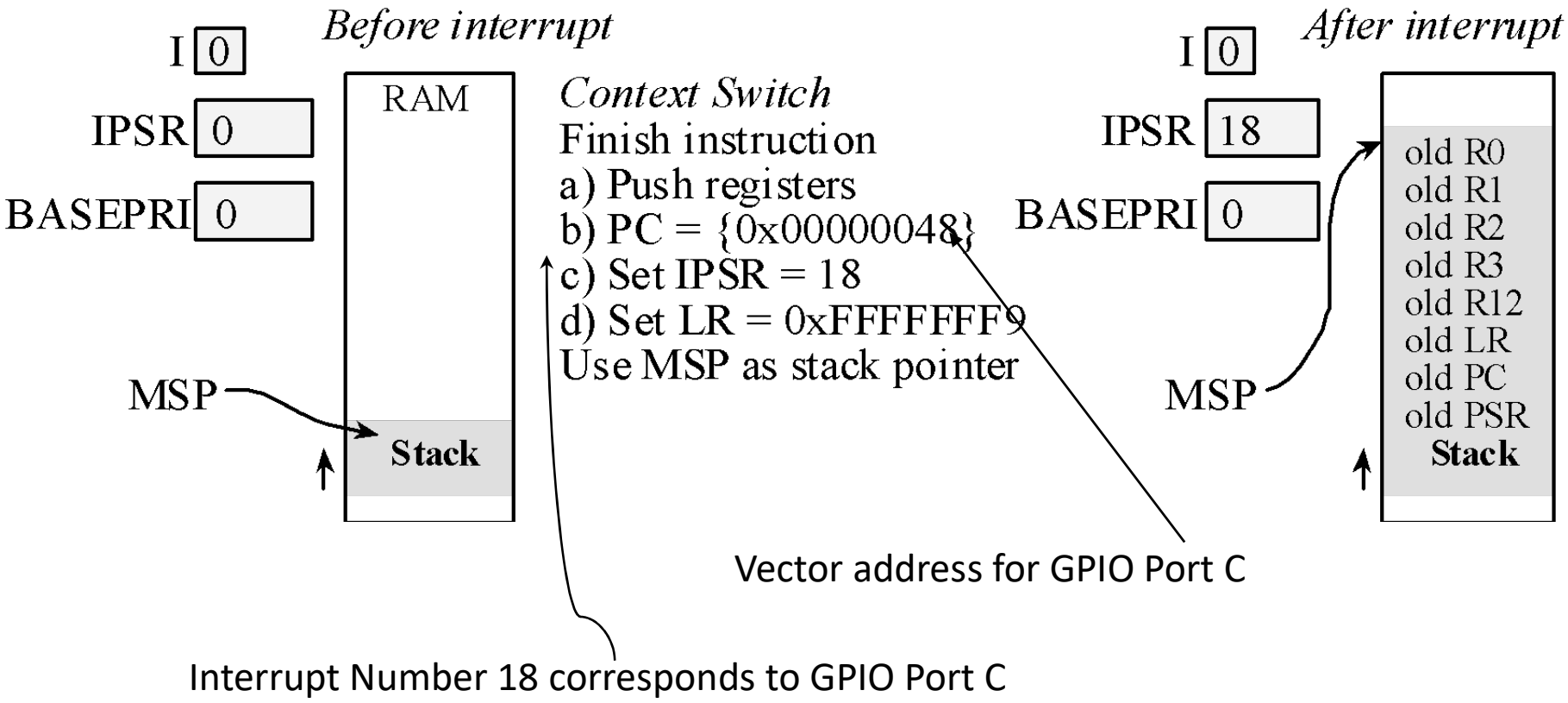
1. The execution of the main program is suspended
    1. the current instruction is finished,
    2. suspend execution and push 8 registers (R0-R3, R12, LR, PC, PSR) on the stack
    3. IPSR set to interrupt number
    4. sets PC to ISR address
2. The interrupt service routine (ISR) is executed
    - ❖ clears the flag that requested the interrupt
    - ❖ performs necessary operations
    - ❖ communicates using global variables
3. The main program is resumed when ISR executes **BX LR**
    - ❖ pulls the 8 registers from the stack

Interrupt Program Status Register (ISPR)

Bits	Description
Bits 31:9	Reserved
Bits 8:0	**ISR_NUMBER:** This is the number of the current exception: 0: Thread mode 1: Reserved 2: NMI 3: Hard fault 4: Memory management fault 5: Bus fault 6: Usage fault 7: Reserved .... 10: Reserved 11: SVCall 12: Reserved for Debug 13: Reserved 14: PendSV 15: SysTick 16: IRQ0(1) ....

Figure 2-3, The IPSR Register.

Interrupt Context Switch



Interrupt Vectors

Vector address	Number	IRQ	ISR name in **Startup.s**	NVIC	Priority bits
0x00000038	14	-2	`PendSV_Handler`	`NVIC_SYS_PRI3_R`	23 – 21
0x0000003C	15	-1	`SysTick_Handler`	`NVIC_SYS_PRI3_R`	31 – 29
0x00000040	16	0	`GPIOPortA_Handler`	`NVIC_PRI0_R`	7 – 5
0x00000044	17	1	`GPIOPortB_Handler`	`NVIC_PRI0_R`	15 – 13
0x00000048	18	2	`GPIOPortC_Handler`	`NVIC_PRI0_R`	23 – 21
0x0000004C	19	3	`GPIOPortD_Handler`	`NVIC_PRI0_R`	31 – 29
0x00000050	20	4	`GPIOPortE_Handler`	`NVIC_PRI1_R`	7 – 5
0x00000054	21	5	`UART0_Handler`	`NVIC_PRI1_R`	15 – 13
0x00000058	22	6	`UART1_Handler`	`NVIC_PRI1_R`	23 – 21
0x0000005C	23	7	`SSI0_Handler`	`NVIC_PRI1_R`	31 – 29
0x00000060	24	8	`I2C0_Handler`	`NVIC_PRI2_R`	7 – 5
0x00000064	25	9	`PWM0Fault_Handler`	`NVIC_PRI2_R`	15 – 13
0x00000068	26	10	`PWM0_Handler`	`NVIC_PRI2_R`	23 – 21
0x0000006C	27	11	`PWM1_Handler`	`NVIC_PRI2_R`	31 – 29
0x00000070	28	12	`PWM2_Handler`	`NVIC_PRI3_R`	7 – 5
0x00000074	29	13	`Quadrature0_Handler`	`NVIC_PRI3_R`	15 – 13
0x00000078	30	14	`ADC0_Handler`	`NVIC_PRI3_R`	23 – 21
0x0000007C	31	15	`ADC1_Handler`	`NVIC_PRI3_R`	31 – 29
0x00000080	32	16	`ADC2_Handler`	`NVIC_PRI4_R`	7 – 5
0x00000084	33	17	`ADC3_Handler`	`NVIC_PRI4_R`	15 – 13
0x00000088	34	18	`WDT_Handler`	`NVIC_PRI4_R`	23 – 21
0x0000008C	35	19	`Timer0A_Handler`	`NVIC_PRI4_R`	31 – 29
0x00000090	36	20	`Timer0B_Handler`	`NVIC_PRI5_R`	7 – 5
0x00000094	37	21	`Timer1A_Handler`	`NVIC_PRI5_R`	15 – 13
0x00000098	38	22	`Timer1B_Handler`	`NVIC_PRI5_R`	23 – 21
0x0000009C	39	23	`Timer2A_Handler`	`NVIC_PRI5_R`	31 – 29
0x000000A0	40	24	`Timer2B_Handler`	`NVIC_PRI6_R`	7 – 5
0x000000A4	41	25	`Comp0_Handler`	`NVIC_PRI6_R`	15 – 13
0x000000A8	42	26	`Comp1_Handler`	`NVIC_PRI6_R`	23 – 21
0x000000AC	43	27	`Comp2_Handler`	`NVIC_PRI6_R`	31 – 29
0x000000B0	44	28	`SysCtl_Handler`	`NVIC_PRI7_R`	7 – 5
0x000000B4	45	29	`FlashCtl_Handler`	`NVIC_PRI7_R`	15 – 13

Interrupt Vectors

Vector address	Number	IRQ	ISR	NVIC	Prority bit
0x000000B8	46	30	GPIOPortFHandler	NVIC_PRI7_R	23-21

Priority registers on the NVIC

Address	31 – 29	23 – 21	15 – 13	7 – 5	Name
0xE000E400	GPIO Port D	GPIO Port C	GPIO Port B	GPIO Port A	NVIC_PRI0_R
0xE000E404	SSI0, Rx Tx	UART1, Rx Tx	UART0, Rx Tx	GPIO Port E	NVIC_PRI1_R
0xE000E408	PWM Gen 1	PWM Gen 0	PWM Fault	I2C0	NVIC_PRI2_R
0xE000E40C	ADC Seq 1	ADC Seq 0	Quad Encoder	PWM Gen 2	NVIC_PRI3_R
0xE000E410	Timer 0A	Watchdog	ADC Seq 3	ADC Seq 2	NVIC_PRI4_R
0xE000E414	Timer 2A	Timer 1B	Timer 1A	Timer 0B	NVIC_PRI5_R
0xE000E418	Comp 2	Comp 1	Comp 0	Timer 2B	NVIC_PRI6_R
0xE000E41C	GPIO Port G	GPIO Port F	Flash Control	System Control	NVIC_PRI7_R
0xE000E420	Timer 3A	SSI1, Rx Tx	UART2, Rx Tx	GPIO Port H	NVIC_PRI8_R
0xE000E424	CAN0	Quad Encoder 1	I2C1	Timer 3B	NVIC_PRI9_R
0xE000E428	Hibernate	Ethernet	CAN2	CAN1	NVIC_PRI10_R
0xE000E42C	uDMA Error	uDMA Soft Tfr	PWM Gen 3	USB0	NVIC_PRI11_R
0xE000ED20	SysTick	PendSV	--	Debug	NVIC_SYS_PRI3_R

NVIC enable registers

Address	31	30	29-7	6	5	4	3	2	1	0	Name
0xE000E100	G	F	…	UART1	UART0	E	D	C	B	A	NVIC_EN0_R
0xE000E104			…						UART2	H	NVIC_EN1_R

PortF Interrupt Initialization

```
void EdgeCounter_Init(void){
  SYSCTL_RCGCGPIO_R |= 0x00000020; // activate port F
  FallingEdges = 0;
  GPIO_PORTF_DIR_R &= ~0x10;    //  make PF4 in (built-in button)
  GPIO_PORTF_DEN_R |= 0x10;     //     enable digital I/O on PF4
  GPIO_PORTF_PUR_R |= 0x10;     //     enable weak pull-up on PF4
  GPIO_PORTF_IS_R &= ~0x10;     //  PF4 is edge-sensitive
  GPIO_PORTF_IBE_R &= ~0x10;    //     PF4 is not both edges
  GPIO_PORTF_IEV_R &= ~0x10;    //     PF4 falling edge event
  GPIO_PORTF_ICR_R = 0x10;      //  clear flag4
  GPIO_PORTF_IM_R |= 0x10;      //  arm interrupt on PF4
  NVIC_PRI7_R = (NVIC_PRI7_R&0xFF00FFFF)|0x00A00000; // priority 5
  NVIC_EN0_R = 0x40000000;      // enable interrupt 30 in NVIC
  EnableInterrupts();           // Enable global Interrupt flag (I)
}
```

PortF Handler

```
void GPIOPortF_Handler(void){

   GPIO_PORTF_ICR_R = 0x10;      // acknowledge flag4

   FallingEdges = FallingEdges + 1;

}
```