Real Time Operating System & Embedded – (I)

工海系 b06501018 朱紹勳

Content

- ARM Instruction Set
  - Function Call
  - Stack
- GPIO
- Interrupt

--

- Compile & Develop of Embedded
  - Debug

--

- Thread
- RTOS

ISA (ARM)

- Arithmetic/Logic

```
AND R1, R2, R3
EOR/ORR R1,R2,#-1
LSR R1,R1,#4
ADD{S} R1,R2,R3, LSL #2
SUB{S} R1,R3, ASR #2
CMP R2,R3
```

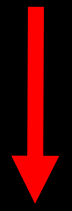
- Data movement

```
MOV R0,#100
ADR R0,Label
LDR R0,=Label
STR{H} R1,[R0]
LDR{{S}H} R1,[R0,#n]
```

- Control

```
B Target
BEQ/BNE Target
BLO/BLS/BHI/BHS Target
BLT/BLE/BGT/BGE Target
```

```
void delay (int cnt){
  while (cnt--);
}

void main(void) {
  delay(10);
}
```



```
delay
  SUB     R0,R0,#0x01
  BNE     delay
  BX      LR
main
  MOV     R0,#0x0A
  BL      delay
```

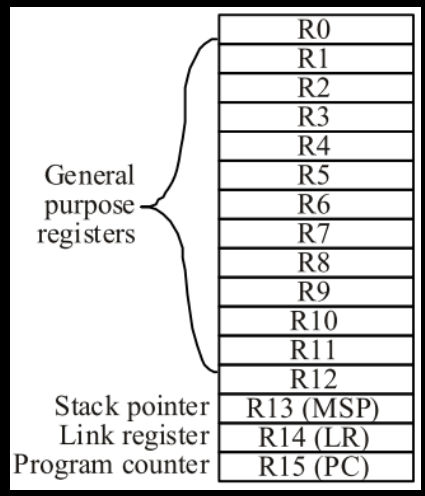
```
void function1 (void){
  output(0x01);
}

int main (void) {
  function1();
}
```

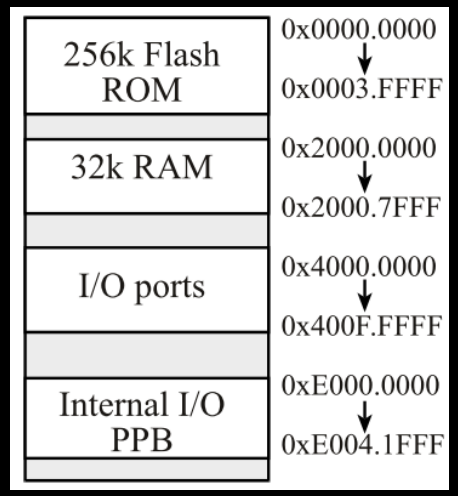
```
function1
  PUSH    {R4-R6,LR}
  MOV     R0,#0x01
  MOV     R4,#12
  BL      output
  POP     {R4-R6,PC}
main
  BL      function1
```

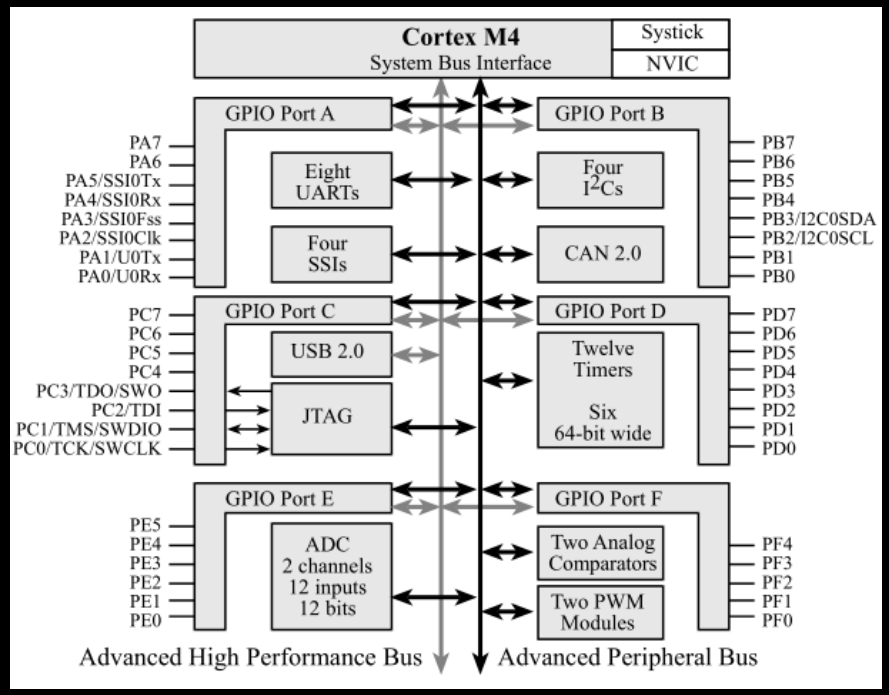
TM4C123

- ARM Cortex-M4

- 256K EEPROM, 32K RAM
- Register: GP x 13, stack pointer, Link register, Program Counter
  total 16 registers
- GPIO: 4 x 8-bit, 1x 6-bit, 1 x 5-bit





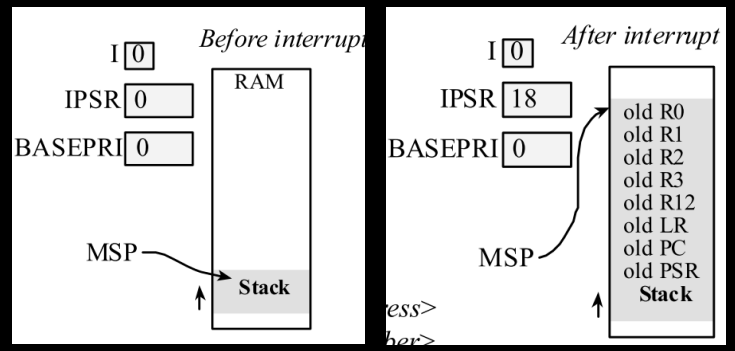


GPIO

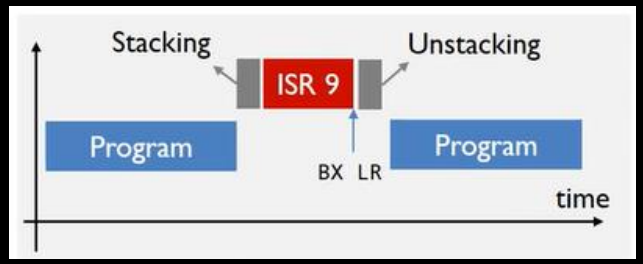
- Initialization
  1. Turn on clock in SYSCTL_RCGCGPIO_R
  2. Wait two bus cycles (two NOP instructions)
  3. Set DIR to 1 for output or 0 for input
  4. Clear AFSEL & AMSEL bits to 0 to select regular I/O
  5. Set DEN bits to 1 to enable data pins
- Input/output from pin
  6. Read/write GPIO_PORTx_DATA_R

Address	7	6	5	4	3	2	1	0	Name
400F.E608	-	-	GPIOF	GPIOE	GPIOD	GPIOC	GPIOB	GPIOA	SYSCTL_RCGCGPIO_R
xxxx.x3FC	DATA	DATA	DATA	DATA	DATA	DATA	DATA	DATA	GPIO_PORTx_DATA_R
xxxx.x400	DIR	DIR	DIR	DIR	DIR	DIR	DIR	DIR	GPIO_PORTx_DIR_R
xxxx.x420	SEL	SEL	SEL	SEL	SEL	SEL	SEL	SEL	GPIO_PORTx_AFSEL_R
xxxx.x510	PUE	PUE	PUE	PUE	PUE	PUE	PUE	PUE	GPIO_PORTx_PUR_R
xxxx.x51C	DEN	DEN	DEN	DEN	DEN	DEN	DEN	DEN	GPIO_PORTx_DEN_R

Interrupts, Interrupt Vector, NVIC



- Interrupt → Context Switch
  1. Push register
  2. Set PC to Interrupt Vector address (next instruction)
  3. Set IPSR(Interrupt Program Status Register) to instruction number
  4. LR(Linker Register) set to 0xFFFFFFxx



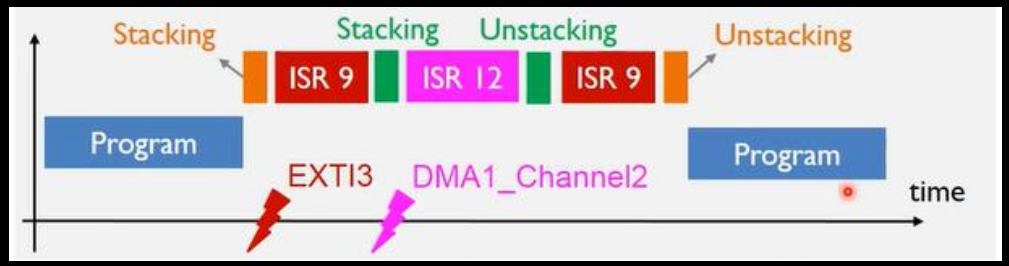
Interrupt Vector:

Vector address(0x), Interrupt num(8-bits), IRQ, ISR name, NVIC, priority bit
Interrupt-num can be negative, ex: -1 for SysTick_Handler, (address of ptr)

NVIC(Nested Vectored Interrupt Controller)

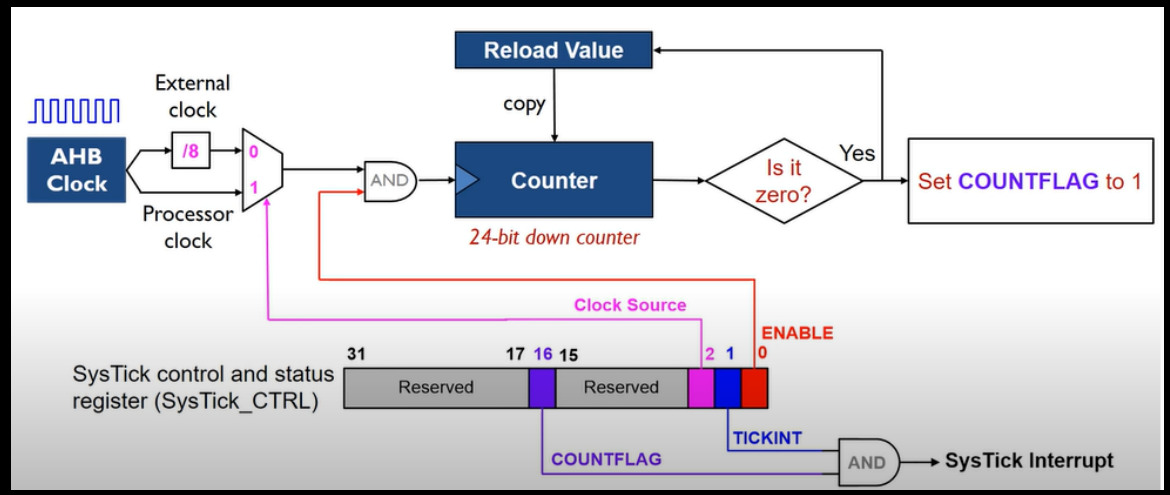
Maskable Channel

Priority level

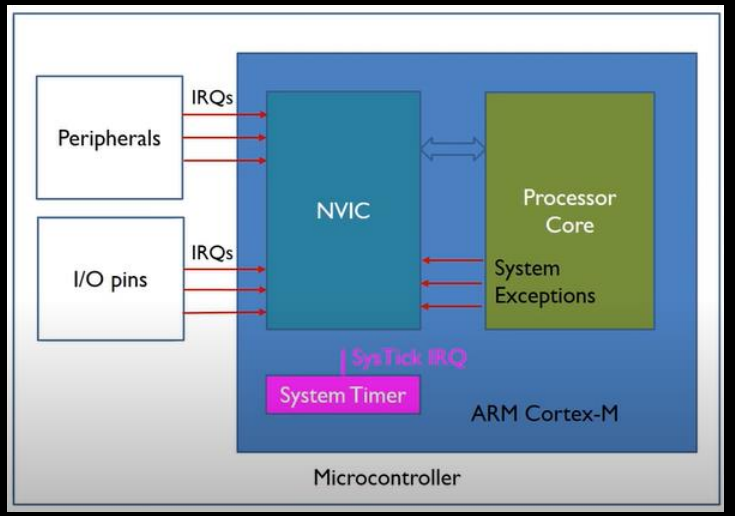


System Tick – (I)

- Use for time elapse & performance measurement
- 24-bits counter from N → 0
  - Clk period = 12.5ns (80MHz)




```
void SysTick_Init(unsigned long period) {
    volatile unsigned long delay;
    SYSCTL_RCGC2_R |= SYSCTL_RCGC2_GPIOD; // activate port D
    Counts = 0;
    delay = SYSCTL_RCGC2_R;   // init, allow time to finish

    GPIO_PORTD_DIR_R |= 0x01; // make PD0 output
    GPIO_PORTD_DEN_R |= 0x01; // enable digital I/O on PD0

    NVIC_ST_CTRL_R = 0;       // disable SysTick during setup
    NVIC_ST_RELOAD_R = period - 1; // reload value
    NVIC_ST_CURRENT_R = 0;    // any write to current clears it
    // Systick=priority 2
    NVIC_SYS_PRI3_R =  (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000;
    NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC
                                        +NVIC_ST_CTRL_INTEN;
    EnableInterrupts();
}
```

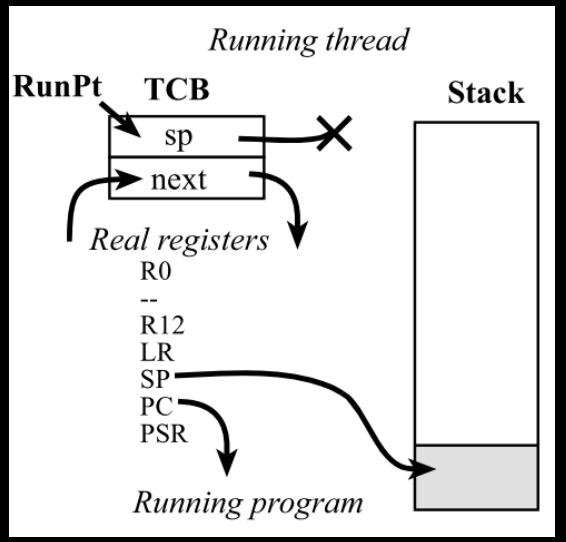


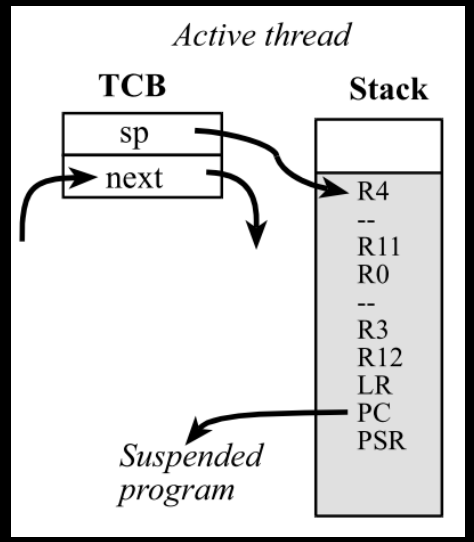
Thread – (I)

- Thread (light-weight process) vs Process (Program, task)
  Share : Code, Data, OS resource
  Own : Stack, Register, PC
- Interrupt-Based Threading (Foreground, Background)
- Classification
  1. Periodic : ex : ADC DAC
  2. Aperiodic : Event Trigger
  3. Sporadic : Faults, error (less frequent and not anticipate)
- Scheduler
  1. Thread Management : Thread states (sleep, active, run, dead, block)
  2. Scheduling Algorithm : (RR, priority), (static, dynamic), (preemptive)
  3. Performance : Utilization, Latency, Bandwidth

Thread – (II)

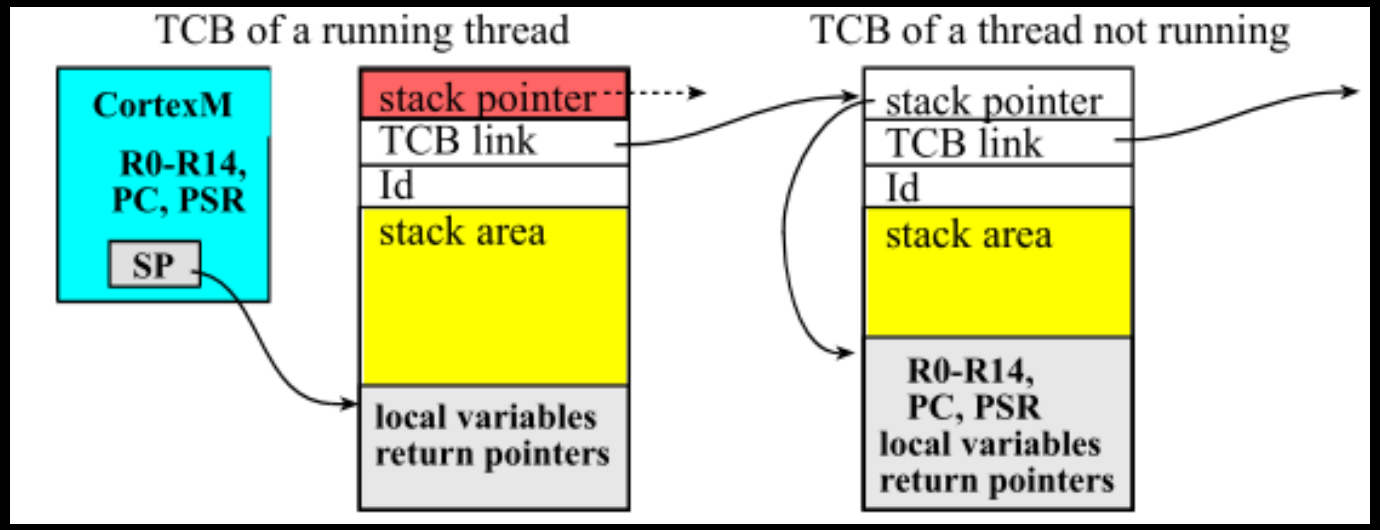
- Thread Control Block(TCB)
  - Nodes of Linked List
  - tid, stack pointer, PC, sleep counter, priority, state
- Thread Switch
  - Context switch need to save context to memory, change page table…
  - Thread switch only involves PC, register, SP…





Thread – (III)

- EX: PendSV (pended system call) Thread Switch
  - Give PendSV handler lowest priority (Prevent switching out background tasks)

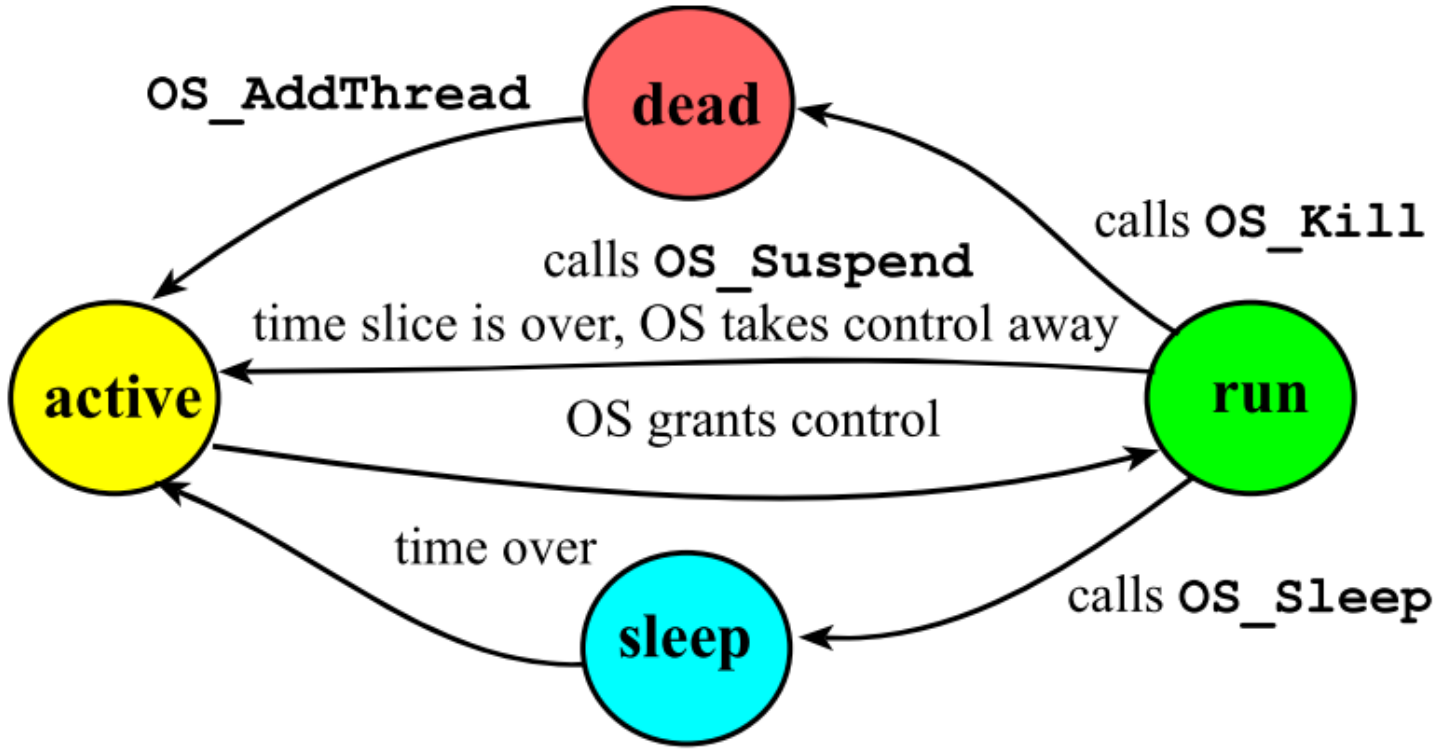


Thread – (IV)

1. Disable interrupts
2. Save registers R4 to R11 on the user stack
3. Save stack pointer into TCB
4. Choose next thread
5. Retrieve new stack pointer
6. Restore registers R4 to R11
7. Reenable interrupts
8. Return from interrupt

```
PendSV_Handler          ; 1) Saves R0-R3,R12,LR,PC,PSR
    CPSID I             ; 2) Make atomic
    PUSH  {R4-R11}      ; 3) Save remaining regs r4-11
    LDR   R0, =RunPt    ; 4) R0=pointer to RunPt, old
    LDR   R1, [R0]      ;    R1 = RunPt
    STR   SP, [R1]      ; 5) Save SP into TCB
    LDR   R1, [R1,#4]   ; 6) R1 = RunPt->next
    STR   R1, [R0]      ;    RunPt = R1
    LDR   SP, [R1]      ; 7) new thread SP; SP=RunPt->sp;
    POP   {R4-R11}      ; 8) restore regs r4-11
    CPSIE I             ; 9) tasks run enabled
    BX    LR            ; 10) restore R0-R3,R12,LR,PC,PSR
```

Thread States



Lab 3 will add **Blocked**

Time Management

- Real Time
  - Hard RT(Guaranteed) > Firm RT(useful = 0) > Soft RT(useful drop) > Not RT
- Usage

  System time, Time stamps (performance measurement)

  Thread sleeping, input capture

Lab (I) Preparation