input/output

① initialize ports:

```
movlw  0xFF    (11111111B)
movwf  PORTA
```

② adjust analog / digital pins:

```
movlw  0X0F    (00001111B) → all digital
movwf  ADCON1     [PCFG[3], PCFG[0]]
```

③ clean ports

void __interrupt (low-priority / high-priority) / empty if none) isr_name (void) { }

variable definition

```
global var1
var1: DS 1
```

clear RBIF flag:

```
movf  PORTB, 0   → read
nop   → one cycle
bcf   INTCON, 0  → clear
```

interrupts

① initialize ports and adjust analog /digital pins

② enable interrupt INT0IE / INT1IE / INT2IE / RBIE (in ADCONx)
RB0 / RB1 / RB2 / RB[4,7]

③ disable priority IPEN (in RCON)

④ clear flag INT0IF / INT1IF / INT2IF / RBIF

⑤ enable peripheral interrupts PEIE (in ADCON)

⑥ enable all / global interrupts GIE (in ADCON)

⑦ interrupt service routine =

```
check flag
 if set → goto func
  clear flag
  retfie 1 (to get register values back)
```

4 MHz clock rate/oscillator frequency
1 MHz ins. frequency
$= \frac{1}{1 \text{ MHz}} = 1 \mu s$

900 ms interrupt

900 ms = 900000 μs (micro seconds)

900000 μs / 1 μs (one cyle) = 900000 cycles

900000/16 (pre scalar value) = 56250 cycles needed

2^{16} (16 bit) 65536 − 56250 = 9286 → load this to TMRxH and TMRxL

timers

① initialize ports and adjust analog /digital pins

② enable timer interrupt TMR0IE / TMR1IE / TMR0IE

③ adjust 8-bit / 16-bit and pre-post scaler T0CON / T1CON / T2CON

④ load TMRxH / TMRxL with desired start value

⑤ enable peripheral and global interrupts PEIE & GIE

⑥ clear flag TMR0IF, TMR1IF, TMR2 IF

⑦ start timer TMR0ON, TMR1ON, TM2 ON

⑧ interrupt service routine = check flag

```
if set → clear flag an reload TMRx with desired start value
retfie 1
```

A/D conversion

① ADON enable

② select channel [CHS0, CHS3]

③ clear go flag

④ select analog inputs [PCFG0, PCFG3]

⑤ ADFM → right justified input

⑥ select acquasition time

⑦ isr func → check ADIF

2 0