NMK322 - Microcontroller

Lecture 08 – System Management

Embedded Systems Requirements

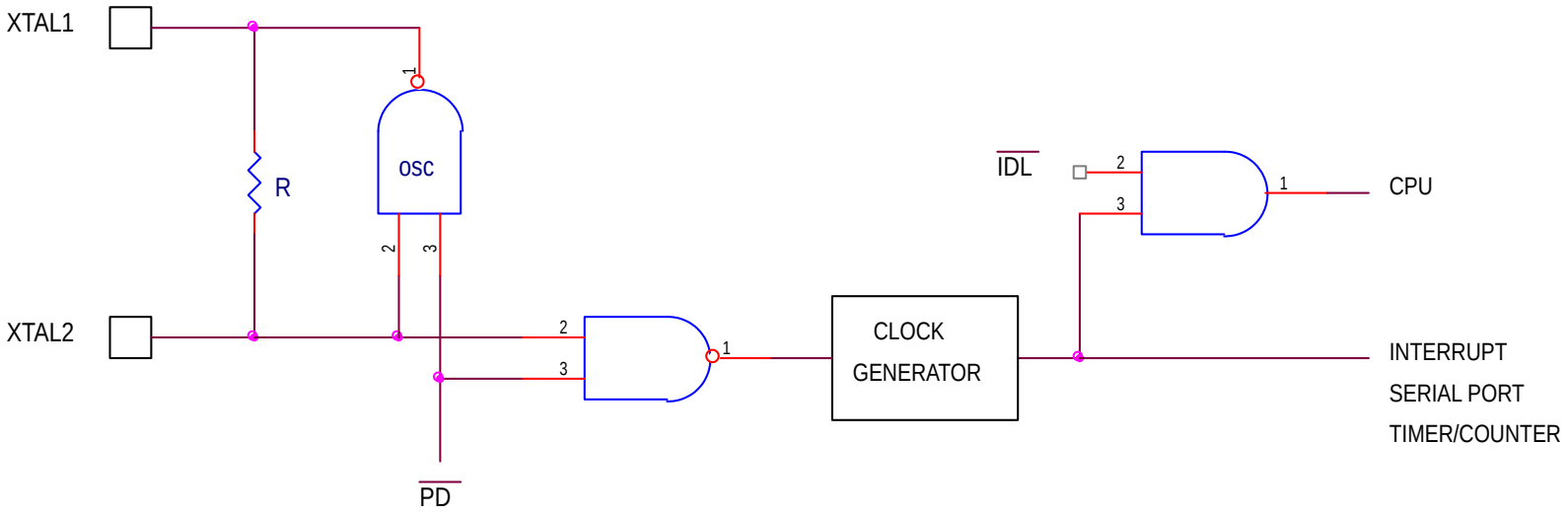
- For portable, battery-operated systems
  - low-power components
  - improved power management
- For reliable, remote systems
  - robust hardware
  - error tolerant software → watchdog!

Power Management

8051 Power Management

- CMOS versions → lower power consumption
  - 2 software-activated *power-reducing* modes
- **Idle** mode
  - internal clock gated-off from CPU
  - CPU status (e.g. SP,PC,IRAM) preserved
  - only oscillator and interrupt circuit running
  - exit on interrupt or hardware reset
- **Power-Down** mode
  - internal oscillator disabled, all functions stopped
  - IRAM (including SFR) preserved
  - exit on hardware reset

80C51 Idle/PD Hardware Circuit



**Note*: *$\overline{PD}$ and $\overline{IDL}$ are bits from PCON Register (SFR @0x87)*

80C51 Idle/PD Power Consumption

Mode / Freq.	0.5 MHz	16 MHz
Operating	2.2 mA	20.5 mA
Idle	0.9 mA	5.0 mA
Power Down	50 uA	50 uA

**Note*: For 80C51 at V_{cc} = 5V*

Power Control Register (PCON)

- 8-bit register (@0x87)
  - **NOT** bit-addressable
  - in HMOS versions only SMOD available

MSB							LSB
SMOD	-	-	-	GF1	GF0	PD	IDL

Name	Description
SMOD	Double baud rate bit. If SMOD=1 , Timer1-generated baud rate doubled.
GF1	General purpose flag bit
GF0	• General purpose flag bit
PD	Power Down bit. Activates Power Down operation (80C51)
IDL	• Idle Mode bit. Activates Idle Mode operation (80C51)

80C51 Idle Mode

- Activated by setting bit 0 in PCON
  - `PCON |= 0x01;`
- Idle Mode:
  - internal clock gated-off from CPU
  - Interrupt, Serial Port & Timer continue to function
  - Stack Pointer, Program Counter, Program Status Word, Accumulator and all register hold data
    - thus, port pins hold last data
  - $\overline{\text{ALE}}$ and $\overline{\text{PSEN}}$ hold at logic HI

80C51 Power Down

- Activated by setting bit 1 in PCON
  - `PCON |= 0x02;`
- Power Down:
  - on-chip oscillator stopped (@disabled)
  - clock frozen → all functions stopped
  - on-chip RAM & SFRs hold data
    - thus, port pins hold last data
  - $\overline{\text{ALE}}$ and $\overline{\text{PSEN}}$ output logic LO

Watchdog

Watchdog Analogy

- imagine a warehouse with a dog guarding
  - the dog will bark when it ‘smells’ intruders
  - the bark will trigger defense mechanism
- a spy intends to break in
  - knows the guard dog is normally hungry
  - prepares pieces of meat and thrower machine
- the spy sets to feed dog every 2 minutes
  - the dog will be busy with the meat and NOT bark
- the spy free to execute his mission
  - if the machine breaks OR runs out of meat, the mission needs to abort (and restart)

Watchdog Timer

- Based on that same principle
  - a timer can act like a watchdog
  - will reset the system it runs out (@overflow)
  - needs to be regularly refreshed to avoid that
- Can be used to recover from system error
  - if the system is frozen (@'hangs'), a watchdog can reboot the system
  - makes the system 'error tolerant'

8051 Watchdog

- Original 8051 is without one
  - software implementation (timer & refresh function)
  - external hardware
- 80C51 uses PCA
  - 16-bit counter/compare → triggers internal reset
- STC12C5A60S2 has one built-in
  - exclusive use, does not use PCA

8051 Watchdog (Software)

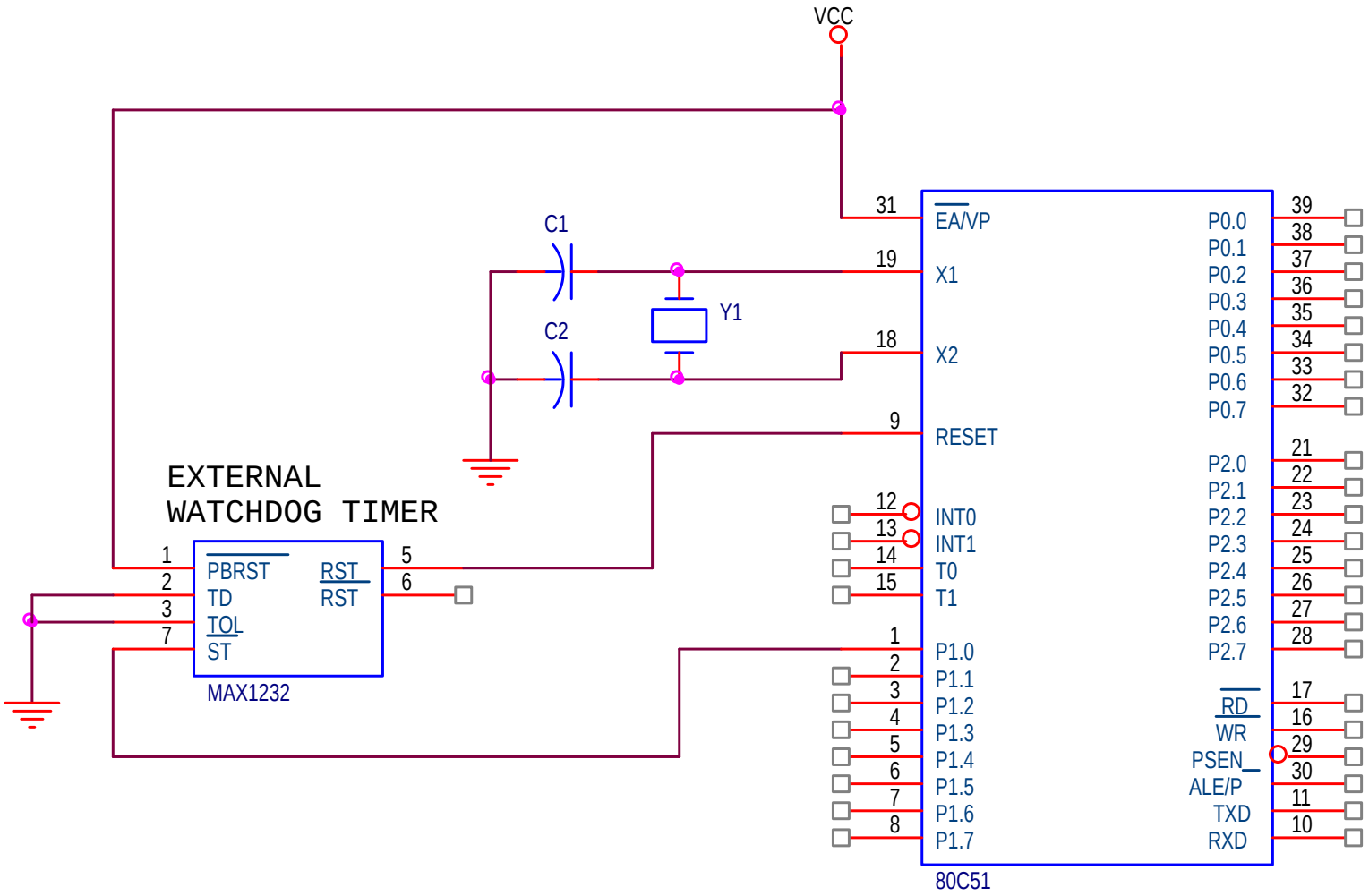
- Set timer to overflow at certain interval
  - greater than maximum time of processing cycle
- Create a refresh function
  - called in every processing cycle
  - restarts the timer (i.e. prevent reset)
- Error recovery strategy
  - balanced refresh interval and timer overflow

Code: 8051 Watchdog (Software)

```
#include <reg51.h>
#define TIME_HI 0x4b
#define TIME_LO 0xfd

void refresh(void) {
  TR0 = 0; TH0 = TIME_HI;
  TL0 = TIME_LO; TR0 = 1;
}

void task1(void) {
  // some stuffs
}

void task2(void) {
  // some more stuffs
}

void watcher(void) interrupt 1 {
  // system_reset
}
```

```
void main(void) {
  refresh();
  EA = 1; ET0 = 1;
  while (1) {
    task1(); task2();
    refresh();
  }
}
```

8051 Watchdog (External HW)



8051 Watchdog (External HW)

- Using hardware like MAX1232
  - configurable timeout (using TD pin)
  - pulse ST pin regularly to avoid restart
  - will reset (RST pin) upon timeout

	Minimum Timeout	Typical Timeout	Maximum Timeout
TD to GND	62.5 ms	150 ms	250 ms
TD floating	250 ms	600 ms	1000 ms
TD to VCC	500 ms	1200 ms	2000 ms

Code: 8051 Watchdog (Ext. HW)

```
#include <reg51.h>
sbit ST = P1^0;

void refresh(void) {
  ST = ~ST;
}

void task1(void) {
  // some stuffs
}

void task2(void) {
  // some more stuffs
}
```

```
void main(void) {
  while (1) {
    task1(); task2();
    refresh();
  }
}
```

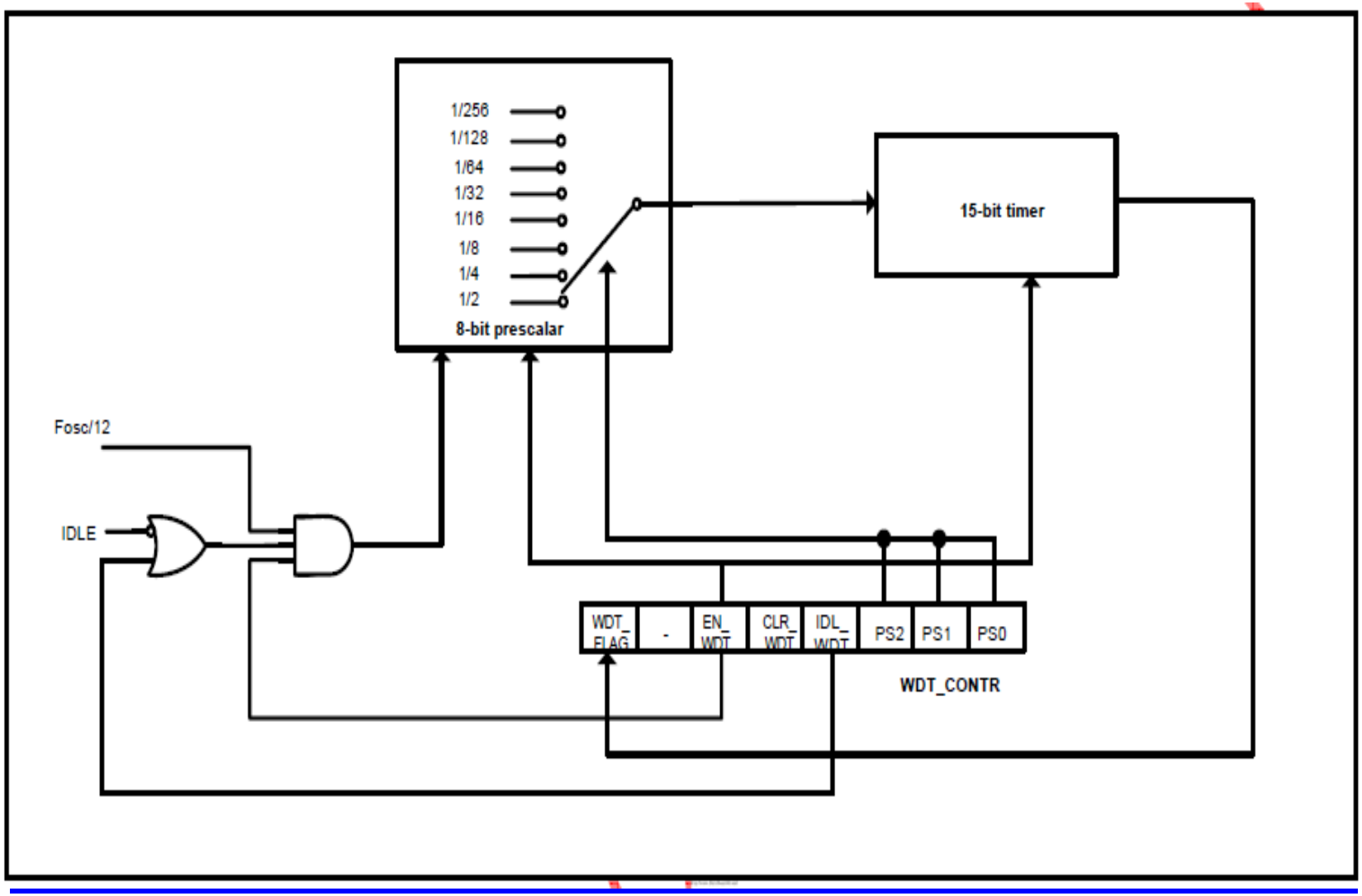
STC12C5A60S2 Watchdog

- A 15-bit timer with 8-bit prescaler
  - 15-bit timer triggered by scaled clock ($f_{sysclk}/12$)
  - configured using WD_CONTR register (@0xC1)

B7	B6	B5	B4	B3	B2	B1	B0
WDT_FLAG	-	EN_WDT	CLR_WDT	IDLE_WDT	PS2	PS1	PS0

WDT_FLAG: WDT reset flag. Cleared by software. Set by hardware to indicate a WDT reset.
EN_WDT : Set to enable WDT.
CLR_WDT : WDT clear bit. Set to restart WDT. Cleared by hardware.
IDLE_WDT : WDT IDLE mode bit. Set to enable WDT in IDLE mode.
PS2, PS1, PS0: WDT Prescale value set bit. (000→ 2, 001 → 4, … , 111 → 256)

STC12C5A60S2 Watchdog



STC12C5A60S2 Watchdog Timer Period

- Given by

$$WDT_{period} = (Prescale\ x\ 32768)/(SYSclk/12)$$

PS2	PS1	PS0	Pre-scale	WDT overflow Time @11.0592MHz
0	0	0	2	71.1 mS
0	0	1	4	142.2 mS
0	1	0	8	284.4 mS
0	1	1	16	568.8 mS
1	0	0	32	1.1377 S
1	0	1	64	2.2755 S
1	1	0	128	4.5511 S
1	1	1	256	9.1022 S

Code: STC12C5A60S2 Watchdog

```
#include <reg51.h>

void refresh(void) {
  WD_CONTR |= 0x10;
}

void task1(void) {
  // some stuffs
}

void task2(void) {
  // some more stuffs
}
```

```
void main(void) {
  WD_CONTR = 0x34;
  while (1) {
    task1(); task2();
    refresh();
  }
}
```

End of Lecture08