Микропроцесорски системи

8. блок: Додатне опције микроконтролера

др Срђан Т. Митровић, дипл. инж.

зимски семестар, 2016./2017. год.

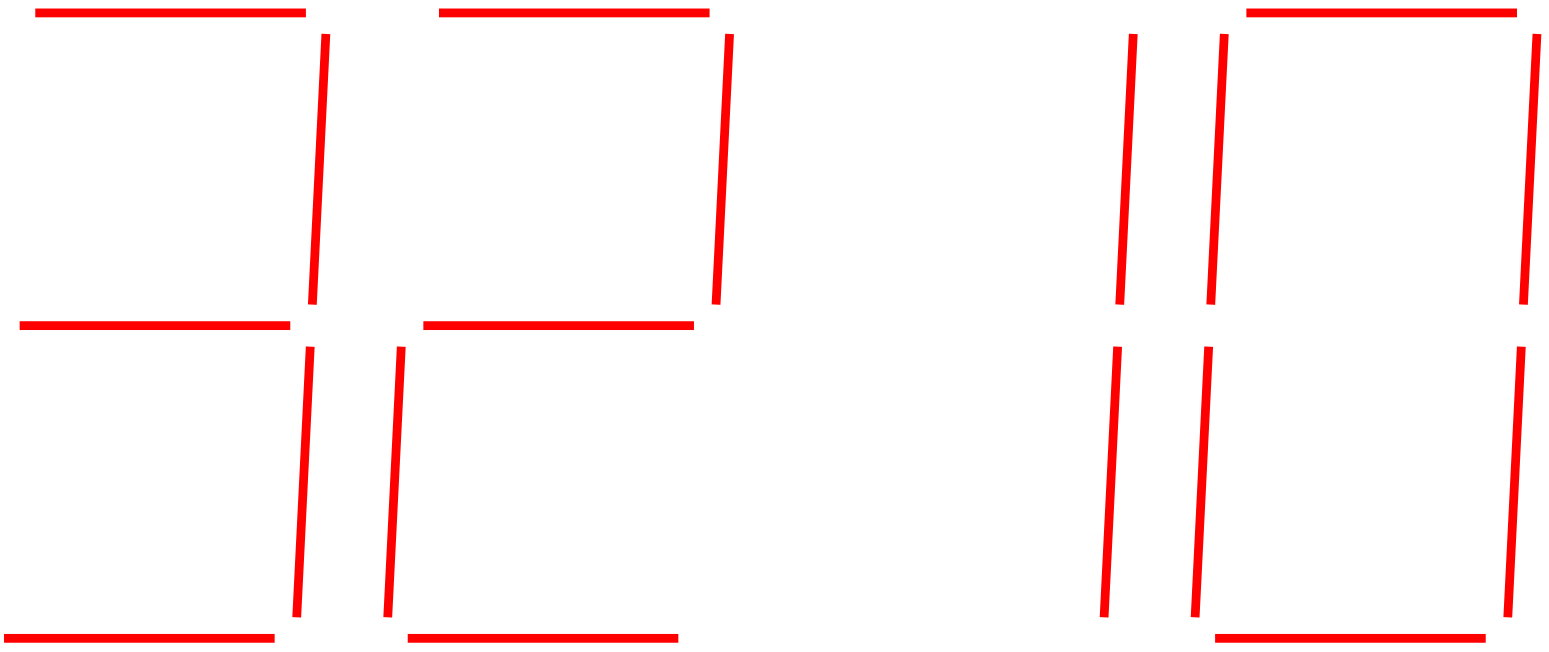
Наставна питања

1. **Watchdog Timer**
   - COP – Computer Operates Properly
2. **Потрошња енергије и *Sleep***
   - Power Consumption and Sleep
   - Clocking Frequency Reduction
   - Voltage Reduction
   - Shutdown of Unused Modules
   - Optimized Design
3. **Ресет**
   - Reset
   - Power-On Reset
   - Brown-Out Reset
   - External Reset
   - Watchdog Reset
   - Internal Reset

Watchdog Timer – “Пас који чува време”
COP – Computer Operates Properly

Намена:

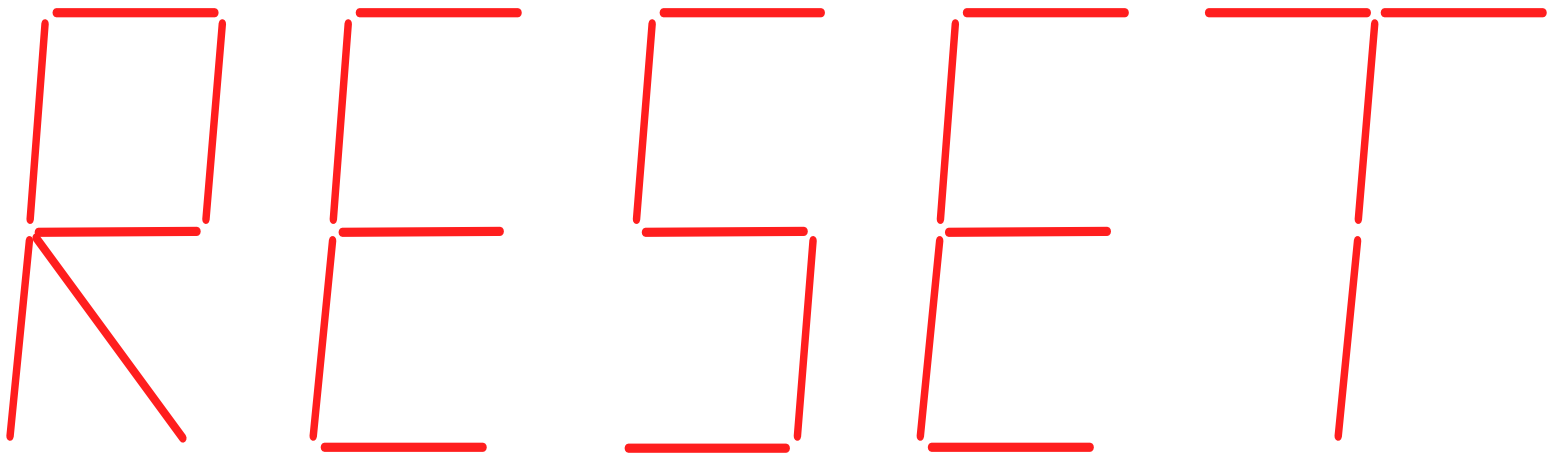
Надгледа извршавање софтвера. Када се укључи – почиње одбројавање.



Watchdog Timer – “Пас који чува време”
COP – Computer Operates Properly

Намена:

Надгледа извршавање софтвера. Када се укључи – почиње одбројавање.



Watchdog Timer – "Пас који чува време"

"kick the dog"

Да би се избегло ресетовање контролера, софтвер мора ресетовати *watchdog* пре него што одброји до нуле.

Помаже

... Наћи пример ...

Не помаже

... Наћи пример ...

Основна намена је утврђивање да ли су се одређени делови програма извршили у оквиру задатог временског интервала. Када програм одступи од свог нормалног тока извршавања, *watchdog* ће ресетовати контролер, што ће можда решити проблем. Свакако постоје ситуације у којима *watchdog* није од помоћи: програм одступа од нормалног тока, али успева да ресетује *watchdog*. Даље, могуће је да разлог због кога програм не успева да ресетује *watchdog* остаје да постоји и након ресета.

Неке особине

- Надгледа коректно извршавање програма
- Функционише независно од осталих модула μC
- Поседује властити интерни осцилатор
- *Sleep* мод не утиче на њега
- Поштовати процедуре за његово искључење (избегавање случајног искључења):
  - код HCS12:
    - $0x55 \rightarrow$ *watchdog reset register*;
    - $0xAA \rightarrow$ *watchdog reset register*
  - ATmega16: захтева сетовање два бита на вредност 1, и ресетовање *watchdog enable* бита

Since the watchdog is used to monitor correct program execution, which means that it both checks whether the controller executes the correct instructions and whether the software at least manages to execute the watchdog reset instructions in time, it is set apart from the other controller modules to allow autonomous operation. As a consequence, the watchdog possesses its own internal oscillator and is hence not affected by sleep modes which shut down the system clock. The watchdog timer features its own enable bit and generally provides some mode bits which control its timeout period. To avoid turning off the watchdog accidentally (after all, if the controller behaves erratically, it may well accidentally clear the watchdog enable bit), a certain procedure has to be followed to turn off the watchdog or to modify its settings. The HCS12, for example, requires that the program first writes 0x55 and then 0xAA to the watchdog reset register. The ATmega16 requires the program to set two bits in a register to 1, and then to reset the watchdog enable bit within four cycles.

Потрошња енергије и *Sleep*

Микроконтролери се често користе у преносним уређајима који се напајају помоћу батерије. Мала потрошња је важна карактеристика микроконтролера. Неколико различитих техника се може користити како би се смањила потрошња електричне енергије:

1. *Clocking Frequency Reduction*
2. *Voltage Reduction*
3. *Shutdown of Unused Modules*
4. *Optimized Design*

Снижавање фреквенције такта

- Контролер ради нормално, али на нижој фреквенцији.
- Потрошња енергије је пропорционална фреквенцији:

$$E \sim f$$

- Контролер има статички дизајн: $f(t) = \text{const}$
- Минимална фреквенција која испуњава временске захтеве.
- Динамичко управљање фреквенцијом $\rightarrow$ додатно коло.

This technique allows the controller to operate as usual, but with a slower frequency. The energy consumption is: $E \sim f$, that is, it is proportional to the frequency. Since controllers have a static design, the frequency can be reduced arbitrarily (as opposed to processors, which have a dynamic design and hence rely on a minimum operating frequency to work properly). In order to utilize this feature, the designer can of course statically clock the controller with the minimum frequency required to meet the timing requirements of the application. But with an appropriate circuit it is also possible to dynamically reduce the frequency whenever the controller does not have to meet tight timing constraints. So although the frequency may have to be high to do some urgent but infrequent computations, it can be turned down during the long intervals in which the controller only goes about its routine tasks.

Снижавање напона напајања

- Потрошња енергије микроконтролера:

$$E \sim U^2$$

- Напон напајања се не може смањити призвољно.
- Када је U нижи од минимално декларисаног, μC се понаша недефинисано.
- U зависи окружења.

This method utilizes the fact that $E \sim U^2$, that is, the energy consumption is proportional to the square of the operating voltage. Hence, a reduction of the operating voltage has a significant impact on the power consumption. Unfortunately, it is not possible to reduce the voltage arbitrarily. The controller is generally specified for a certain voltage range. If the voltage drops below this level, the controller may behave arbitrarily. The minimum voltage that still allows the controller to function correctly depends on the environmental conditions. As with frequency reduction, voltage reduction may either be done statically or dynamically. It may be combined with a sleep mode, as in the 8051.

Искључивање модула ван употребе

- Модули који се не користе → искључити
- I/O или CPU
- I/O - пинови се сетују као улазни
- Овај метод се користи за *Sleep* мод.
- Искључење спољашњег осцилатора → стабилизација ∼ms
- Деактивирати модуле ван употребе пре уласка у *Sleep* мод.

Since each active module draws power, it would obviously be a good idea to shut down unused modules. So if the controller only has to do internal computations, its bus or I/O components can be turned off for this duration. On the other hand, if the controller just waits for some external event, its CPU and other parts may be shut down until the event occurs. Note that shutting down the (digital) I/O module may entail that all pins are set to input, so you may not be able to drive an output pin and turn off the I/O at the same time.
This method is generally used for the sleep modes of a controller. Controllers tend to provide several different sleep modes, which differ in the components they shut down. Note that since the oscillator is shut down in this mode, it cannot be used to recognize external interrupt conditions. Therefore, controllers tend to use some internal oscillator, e.g. the watchdog oscillator, to sample the input line. However, this implies that the timing for these interrupt conditions will differ from the usual one where the external oscillator is employed.
Waking up from a sleep mode takes a few cycles at best, and may well take milliseconds if the oscillator was shut down as well. This is due to the fact that an oscillator needs some time to stabilize after it has been activated. Also be certain of the condition the modules are in after a wake-up. Some modules may erroneously trigger interrupts if the interrupt was enabled before the module was shut down, so take appropriate precautions before entering a sleep mode. In some cases, it is also necessary to manually deactivate unused modules before entering a sleep mode so they do not draw current needlessly. An example is the analog module of the ATmega16, which will remain active during sleep mode if not disabled first.

Оптимизирано пројектовање микроконтролера

Finally, it is of course possible to optimize the controller's energy consumption up front during its design. A good example for this technique is the MSP430 family of Texas Instruments, which has been optimized with regard to energy consumption and hence only requires less than $400\mu A$ during normal operation. In comparison, other controllers tend to have a nominal consumption in the mA range. The ATmega16, for instance, consumes 1.1 mA during normal operation and $350\mu A$ in its idle mode (which turns off CPU and memory, but keeps all other modules running).

Ресет

Дефиниција:

Довођење система у добро познато стање, у ком не делују откази.

μC може реаговати на различите узроке ресета који су дефинисани посебним *reset flags*. Док год је ресет активан μC се "прави мртав".

Након ресета:

1. Сви регистри се иницијализују на подразумеване вредности
2. I/O су подешени као улазни.
3. μC остаје у стању ресета док постоје узроци који су га изазвали.
4. Чека се одређено време док се осцилатор стабилизује.
5. Извршава прву инструкцију програма са адресе $0x0000$
6. $\uparrow$ ту се уобичајено поставља скок на процедуру за обраду ресета.
7. Чекање се може конфигурисати у интервалу [μs . . . ms].
8. Неки μC чак проверавају квалитет сигнала такта (HCS12).

Ресет при укључењу

Power-On Reset–POR

Кад напон напајања пређе унапред дефинисани праг долази до POR ресета.Овим се обезбеђује ресетовање контролера након укључења напајања, при чему се он иницијализује.

Brown-Out Reset–BOR

Користи се у ситуацијама када се не може гарантовати стабилан напон напајања. Он једноставно поставља контролер у стање ресета када напон напајања излази ван дозвољених граница. Овај ресет није потребан у стабилним системима, и код неких контролера се дозвољава кориснику да га укључи (искључи) по портеби.

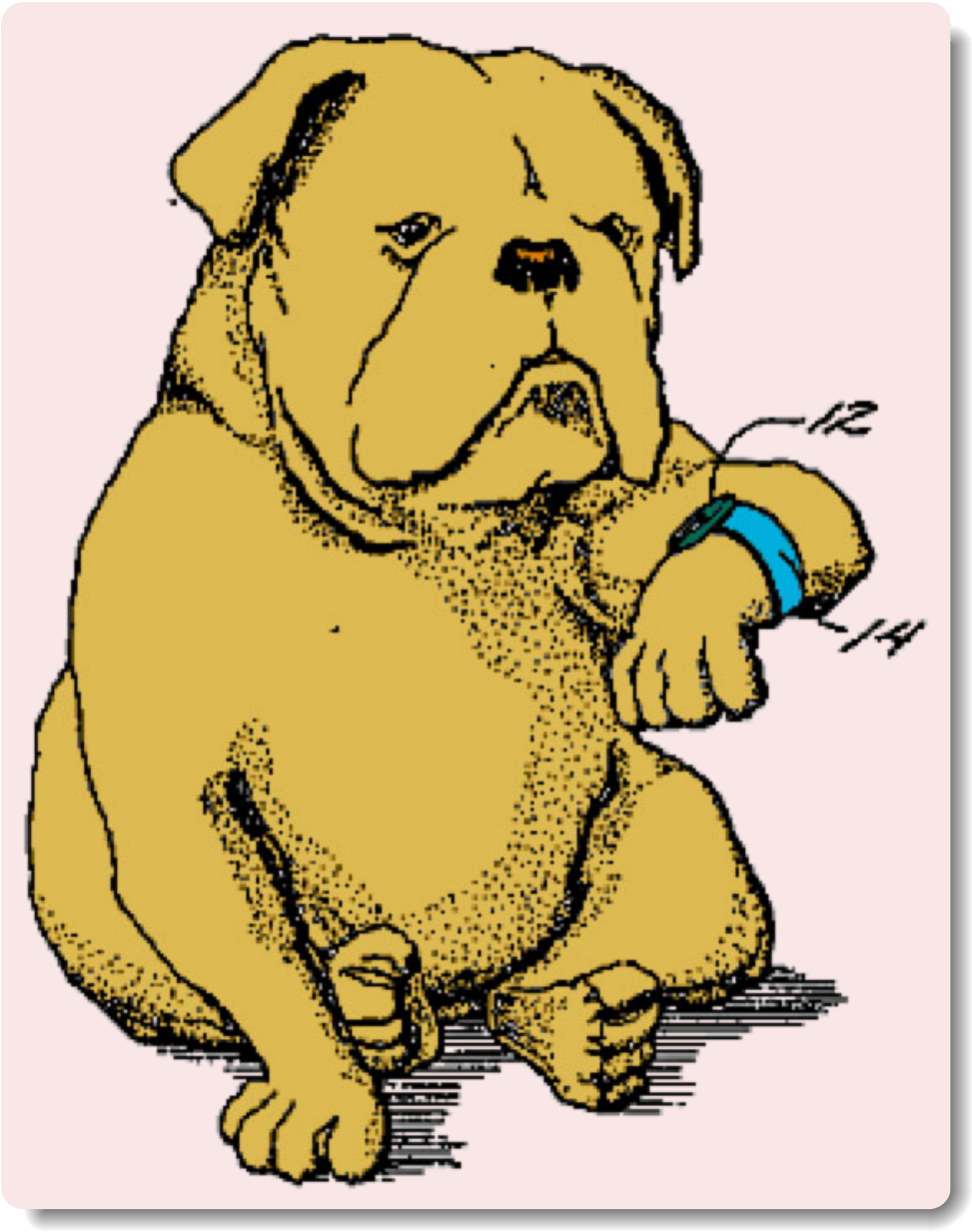
Екстерни ресет

External Reset

Уобичајено је да се овај ресет покреће преко специјалног пина.

- стање "*high*" – нормалан рад
- стање "*low*" – ресет
- стање пина се узоркује по такту локалног осцилатора (watchdog oscillator)
- ресет пин физички увек мора бити спојен (чак и када се не користи)

Watchdog Reset



Интерни ресет

Internal Reset

Some controllers offer an instruction that causes a software reset. This can be useful if a data corruption or some other failure has been detected by the software and can be used as a supplement to the watchdog.