

## Микропроцесорна системотехника

### Лабораторно упражнение №3

#### “Управление на входно-изходните портове на микроконтролера MSP430FR6989”

##### 3.1. Въведение.

MSP430FR6989(PZ100) има 100 извода, от които **83** могат да се използват като логически **входи/изходи за общо предназначение** (GPIO – General Purpose Input Output). Конфигурацията им се променя в зависимост от фърмуера, който изпълнява микроконтролера. Texas Instruments номерират всеки извод по следната схема:

$$P_{X,Y},$$

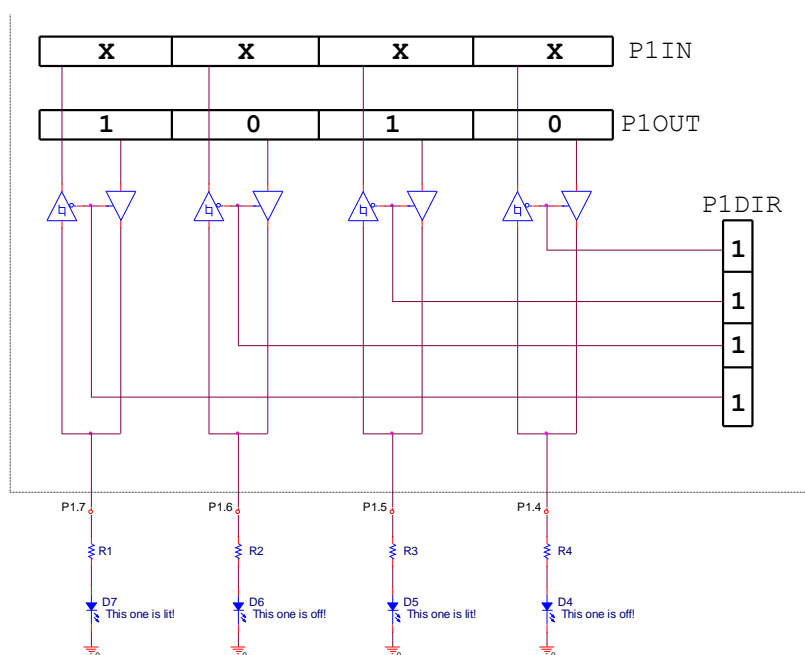
където X е номера на входно-изходния порт, а Y – номера на входно-изходния извод.

Портовете се номерират с цифра от 1 ÷ 9, като изключение прави PJ.y, който обозначава специален порт, към който е свързан JTAG интерфейсът. Ако JTAG не се използва, те могат да се използват като изводи с общо предназначение. Входовете са с тригер на Шмит. Един порт включва до **7 извода**. Номерът на извода може да приема стойности P1.(0 ÷ 7), P2.(0 ÷ 7), P3.(0 ÷ 7), P4.(0 ÷ 1), PJ.(0 ÷ 5) и т.н.

На всеки извод отговаря по един бит от даден регистър в микроконтролера. Най-важните регистри за управлението на портовете са:

- **PxDIR** – регистър, указващ типа на извода. Избира дали ще е вход или изход. При лог. **0** - **вход**, при лог. **1** - **изход**.
- **PxIN** – регистър, отразяващ логическото състояние на изводите, конфигурирани като входи. Ако на входа е подадено напрежение отговарящо на лог. 1 - съответния бит е в лог. 1. Ако на входа е подадено напрежение отговарящо на лог. 0 - съответния бит е в лог. 0.
- **PxOUT** – регистър, контролиращ логическото състояние на изводите, конфигурирани като изходи.

Ако конфигурираме изводите на микроконтролера като изходи и свържем към тях периферни устройства, например светодиоди, то чрез запис в PxOUT регистъра ще можем да управляваме тези устройства посредством логическите нива на изводите. Такъв пример е показан на **фиг. 3.1**.

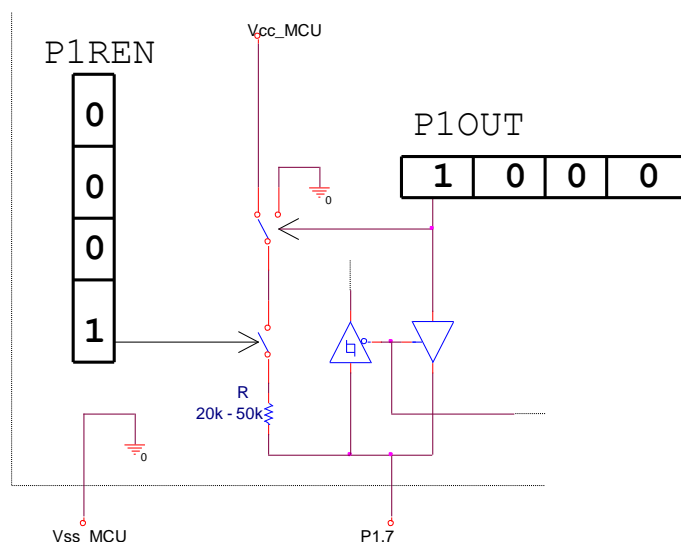


Фиг. 3.1. Пример за управление лог. ниво на входно-изходните изводи

За простота е показан само най-старшият квартет (битове 4 ÷ 7) от използваните регистри. При запис на 10100000<sub>(b)</sub> в P1OUT ще светят само светодиоди D7 и D5.

Аналогично при конфигуриране на изводите като входове, ако подадем логическа 1 на P1.7 и P1.5, то бит 7 и 5 от P1IN ще се установят в 1. Тогава програмата може да прочете регистър P1IN, което ще върне резултат 10100000<sub>(b)</sub> и така да разбере на кой извод какво ниво е подадено.

Често в микроконтролерите се включват вътрешни резистори, „издърпващи“ даден вход към логическа 0 или 1 (фиг. 3.2). Това позволява намаляване броя на външно-включваните елементи. Най-често това се използва при свързване на бутони и схеми с отворен колектор/дрейн.



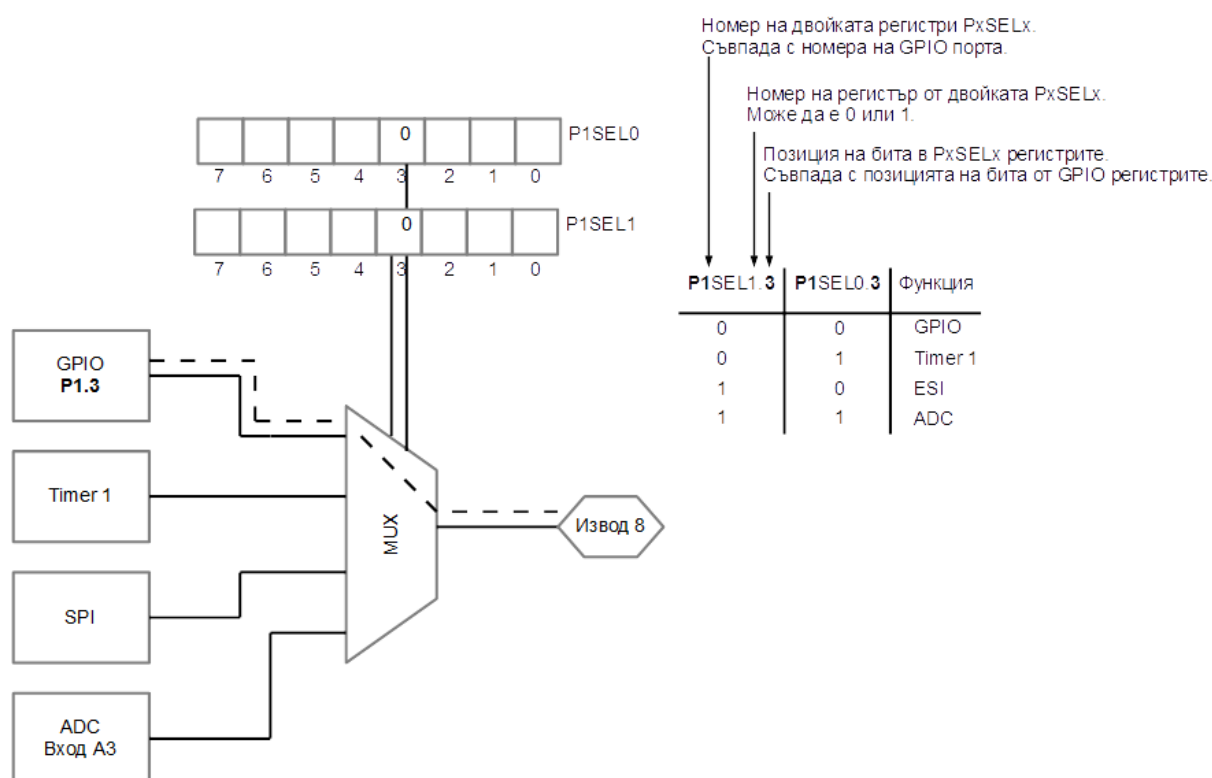
Фиг. 3.2. Свързване на „издърпващ“ резистор към извод на контролера

- **PxREN** – регистър за включване/изключване на вътрешния издърпващ резистор (лог. **1** – **включен**, лог. **0** – **изключен**). Когато дадения извод е конфигуриран като вход, PxOUT задава вида на резистора (лог. **1** – **pull-up**, лог. **0** – **pull-down**). Стойността на резистора е в границите **20 ÷ 50 kΩ** (по каталожни данни).
- **PxIE** – регистър, разрешаващ прекъсванията (interrupts). Ако даден извод е конфигуриран като вход, той може да се използва за генериране на прекъсване (лог. **1** – **прекъсването от даден извод е разрешено**, лог. **0** – **прекъсването е забранено**).

**Прекъсването** е процес на **спиране изпълнението на главната програма** и стартиране изпълнението на **специална програма** (interrupt handler), разположена на различен адрес в паметта. Този процес е **асинхронен** спрямо изпълнението на главната програма. Той позволява микроконтролерът свободно да извършва дадени операции и **само при настъпване на събитие (прекъсване)** да обслужва заявката чрез специален сорс код. Алтернативен метод на работа е с постоянна проверка (polling) за настъпило събитие, но докато тя се извършва, микроконтролерът не може да прави нищо друго.

- **P1IFG** – регистър с битове, отговарящи на **источника на прекъсване**. Специалната програма (interrupt handler) в повечето случаи е само една за GPIO модула. При работа с два или повече извода, програмата няма как да знае кой от тях е генерирал прекъсването. Затова регистър P1IFG съдържа флагове, осигуряващи ни информацията кой точно извод е генерирал прекъсването. Позицията на флага (бита) носи тази информация.
- **PxIES** – регистър, определящ **активния фронт** на прекъсването. Прекъсването от извод, конфигуриран като вход, може да се осъществи по два начина – по **нарастващ фронт** ( $0 \rightarrow 1$ ) или по **спадящ фронт** ( $1 \rightarrow 0$ ) на входния сигнал. Лог. **0** – **конфигурира прекъсване по нарастващ фронт** (rising edge), лог. **1** – **по спадящ фронт** (falling edge). Други микроконтролери позволяват прекъсване по четири начина – по фронт ( $0 \rightarrow 1$  или  $1 \rightarrow 0$ ), както и по ниво ( $1 / 0$ ).

- **PxSEL0** и **PxSEL1** – регистри, конфигуриращи **функцията на извода**. При всички микроконтролери даден извод понякога може да извършва **повече от една функция**. В настоящото упражнение разглеждаме функцията GPIO, но изводите могат да се използват и от ADC, компаратор, PWM, SPI и други модули. За да се превключи функцията на извода, производителят на ИС реализира **мултиплексор**, като всеки негов вход се избира от регистри PxSEL0 и PxSEL1 (за MSP430). Данни за функциите на всеки извод може да се намерят в каталога (datasheet) на микроконтролера. На **фиг. 3.3** е показан извод 63 на MSP430FR6989 (PZ100), който може да се използва от GPIO, Timer1, ESI, и ADC модула. Както се вижда от таблицата на фигурата, името на регистъра се формира от номера на GPIO порта, свързан към дадения извод. Това значи, че регистри P1SEL0 – P1SEL1 мултиплексират изводи P1.0 – P1.7, P2SEL0 – P2SEL1 мултиплексират P2.0 – P2.7 и т.н. Комбинацията от допълнителни периферни модули, свързани към изводите, зависи от конкретния микроконтролер.



**Фиг. 3.3.** Мултиплексиране на изводите

### Пример за използване на прекъсване с използване на входно-изходен извод

Пример за програма, използваща прекъсвания е показана на **фиг. 3.4**. Представена е блоковата схема на устройството и управляващия алгоритъм.

Най-вляво е представен алгоритъма на основната програма. Тя включва инициализация на термометъра (извършва се еднократно след рестартиране на микроконтролера) и безкрайния цикъл на основната програма, включващ измерване на температурата от датчика чрез измерване на напрежението и визуализация на температурата в градуси Целзий или градуси Фаренхайт в зависимост от състоянието на булева променлива (флаг), отговаряща за това в каква мерна единица да се визуализира температурата.

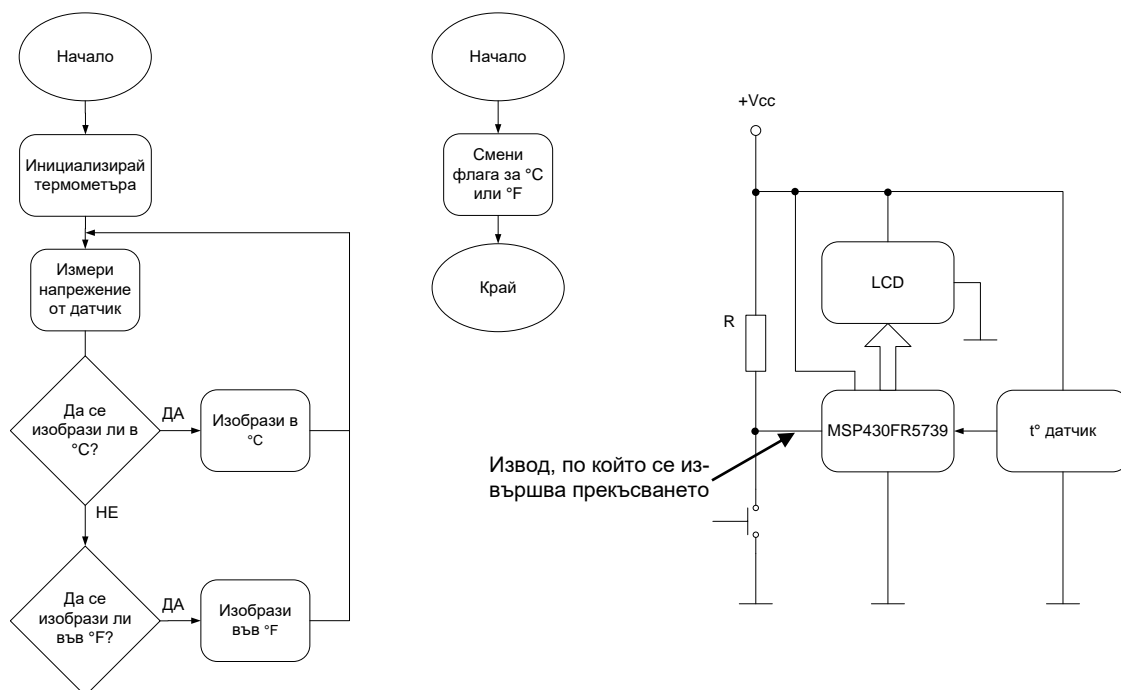
Реализирано е възможност за прекъсване чрез промяна на лог. ниво на вход на микроконтролера. Тази промяна се случва при натискане на бутона. Ако бутона е **отпуснат** - на входа има лог. **1**, а ако е **натиснат** - лог. **0**.

Когато потребителят натисне бутона, микропроцесорът получава прекъсване и обслужва по-малка програма (алгоритъма е в средата на фигурата). Тази промяна сменя състоянието на флага

за мерната единица на температурата. Ако е била градуси Целзий - става градуси Фаренхайт и обратно.

След това изпълнението на главната програма продължава там, откъдето е била прекъсната. От тук нататък температурата ще се изобразява в новата мерна единица, докато не бъде натиснат пак бутона.

Чрез този пример се вижда ясно предимството на използването на прекъсване. Настройка на мерната единица на температурата няма да бъде извършвана често, дори вероятно само еднократно при смяна на батерии. Ако не се използва прекъсване, трябва в главната програма да се извършва проверка дали бутона е натиснат. Ако главната програма се изпълнява за 1 секунда, това означава че ние ще „проверяваме“ всяка секунда дали е натиснат бутон, който може да бъде натиснат веднъж на няколко месеца или дори никога.



**Фиг. 3.4.** Пример за използване на прекъсване с използване на входно-изходен извод

## ЗАДАЧИ ЗА ИЗПЪЛНЕНИЕ

1. Да се създаде нов проект с име **Lab\_3\_1** в папка **/Desktop/MSHT\_GR\_XX\_N** и да се копира програмата на асемблер, която включва и изключва светодиод от демоплатката с MSP430FR6989. Да се изпълни програмата стъпка по стъпка. Да се наблюдават регистрите на микропроцесора и изходния порт. Да се постави точка на прекъсване.

Принципната схема е показана в **Schema.pdf**. С цел опростяване, част от елементите на макета са пропуснати. Разположението на елементите на макета е показано в **Maket.pdf**.

2. Да се създаде нов проект с име **Lab\_3\_2** в папка **/Desktop/MSHT\_GR\_XX\_N** и да се реализира задача 1, но с програмния език C и друг (произволен) светодиод.

3. Да се създаде нов проект с име **Lab\_3\_3** в папка **/Desktop/MSHT\_GR\_XX\_N** и да се реализира програма на C, която да чете състоянието на бутон **S1** и да превключва светодиод **D8** от платката. Да се използва метода “polling”. Да се наблюдава регистъра **P1IN**.

4. Да се създаде нов проект с име **Lab\_3\_4** в папка **/Desktop/MSHT\_GR\_XX\_N** и да се реализира програма на C, която да чете състоянието на бутон **S1** и да превключва светодиод **D8** от платката. Да се използват прекъсвания.