Валентина Василева Ранковска

Микропроцесорна схемотехника

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
ГАБРОВО
2021

Валентина Василева Ранковска

Микропроцесорна схемотехника

ISBN 978-954-683-655-7

УНИВЕРСИТЕТСКО ИЗДАТЕЛСТВО
"ВАСИЛ АПРИЛОВ"
ГАБРОВО, 2021

Съдържание

Въведение

Настоящото учебно пособие е предназначено за студенти от специалност „Промишлена и автомобилна електроника", ОКС „бакалавър" на Технически Университет – Габрово за обучение по дисциплината „Микропроцесорна схемотехника". Същевременно то може да бъде полезно и за студенти и докторанти от други образователно-квалификационни степени и специалности, както и за инженери-проектанти, които биха искали да се запознаят с микроконтролерите на Microchip с PIC микропроцесорно ядро.

Като обект на разглеждане е приет един от множеството най-нови микроконтролери от фамилията PIC16 до момента - PIC16(L)F18855/75.

Темите от **раздел 1** включват въведение в дисциплината, като запознават читателя с основни термини и въпроси от областта на микропроцесорната схемотехника – микропроцесор, микроконтролер (едночипов микрокомпютър), микропроцесорна система, видове организация на паметта и архитектури на микропроцесори. Разглеждат се подробно блокови схеми на микроконтролерите от фамилията PIC16 и на микропроцесора – ядрото на микроконтролера, както и организацията на входно-изходните портове. Обяснява се механизмът на конвейерно изпълнение на инструкциите.

Накратко се представя организацията на различните типове памет и методите за адресирането им.

Темите в **раздели от 2 до 5** са посветени на периферните блокове на избрания съвременен микроконтролер от фамилията PIC16. Разглеждат се техните основни функционални възможности и характеристики, режими на работа, приложение. Изложението е съпроводено с множество блокови схеми и времедиаграми, поясняващи режимите на работа на някои от тях.

Раздел 6 включва теми, свързани с важни специални характеристики на микроконтролерите PIC16 - причини за установяване в начално състояние, механизъм на обслужване на прекъсвания, генериране на тактови сигнали, предназначение на следящия таймер и др.

Трябва да се отбележи, че най-новите микроконтролери от фамилията PIC16 (какъвто е споменатият по-горе) имат значителни подобрения и разширения, в сравнение с разглежданите в предходното издание на „Микропроцесорна схемотехника" [1], като:

- по-усъвършенствана архитектура на процесора;
- по-голямо разнообразие от различни типове периферни блокове при отделните представители на фамилията (напр. температурни индикатори, ЦАП, блокове за детектиране преминаване през нулата на аналогов сигнал, конфигурируеми логически клетки и др.);

- наличие на микроконтролери с повече на брой периферни блокове от един и същи тип;
- разширени функции и режими на работа на съществуващите блокове (напр. таймери, CCP и др.);
- значително увеличен обем памет при някои микроконтролери, както и по-развита организация на паметта и др.

Ето защо, поради ограничения обем на настоящата книга, за разлика от [1], информацията за някои блокове е съкратена до ниво, позволяващо на студентите да се запознаят с основната информация за тях, като архитектура, входно-изходни сигнали, режими на работа, основни моменти от конфигурирането им, без те да се разглеждат детайлно.

За да могат студентите да се възползват от тези знания по-нататък, при реално инженерно проектиране с използване на микроконтролери, те ще се нуждаят от подробно изучаване на техническата документация на избрания микроконтролер и в частност на използваните в конкретното приложение периферни блокове, откъдето ще трябва да направят справка за подробните особености, изисквания, алгоритми за конфигуриране при реализация на дадени функции, за подробната карта на паметта за данни и структурата на специалните функционални регистри, използвани при конфигурирането на даден периферен блок, както и за предназначението на битовете в тях, за електрическите характеристики, свързани с работата на даден блок и микроконтролера като цяло, за процесите на програмиране, тестване и настройка на микропроцесорно устройство и много други.

Паралелно тези знания се допълват от методическите ръководства за семинарни и лабораторни упражнения, публикувани на уеб-адрес *http://dmoodle.tugab.bg/*, където се дават допълнителни теоретични сведения за апаратната част, изучават се етапите и средствата за създаване на програмно осигуряване на Асемблер и C, както и се решават различни практически задачи.

По-нататък в курса на обучение студентите ще имат възможността да разширят познанията си и практическите си навици и умения по избираемата дисциплина „Вградени микрокомпютърни системи“.

Авторът изказва своята благодарност към проф. д-р инж. Анатолий Трифонов Александров за подробната рецензия и отправените препоръки за подобряване на качествата на тази книга, както ще приеме и всички съвети и забележки, отправени от читателя на електронна поща *rankovska@tugab.bg*, които биха допринесли за бъдещото ѝ подобряване при следващо издание.

Август, 2021 г. *От автора*

Раздел I. Въведение в микроконтролерите. Архитектура, организация на паметта и функциониране на микроконтролерите от фамилията PIC16.

1. Въведение в микропроцесорната схемотехника. Микропроцесор, микроконтролер, микропроцесорна система

1.1. Въведение

През 70-те години на XX век проектантите на системи за управление започват да използват изчислителни системи на базата на микропроцесори (МП), произвеждани от фирми, като Intel, Motorola и др. Това позволява да се повиши скоростта и ефективността на проектиране на нови системи, да се намалят разходите за оборудване и отстраняване на неизправности, а също и да се намали себестойността на производството. Сами по себе си обаче микропроцесорите не притежават възможността да решават непосредствени задачи в областта на управлението – за постигане на целта проектантите добавят допълнителни устройства, като памети за програми и данни и периферни блокове, като таймери, броячи, аналого-цифрови и цифрово-аналогови преобразуватели, програмируеми контролери за вход-изход и др. В резултат възниква идеята за интеграция на най-често използваните блокове в рамките на един чип.

Реализацията на тази идея е през 1976 г., когато фирмата Inel произвежда устройство с кодовото означение 8048, което по-късно получава наименованието „микроконтролер". То става основа на системи за управление в робототехническите комплекси, битовата електроника и др. и е първият микроконтролер, който получава широко разпространение.

Друга водеща фирма, която почти по същото време започва разработката и производството на микропроцесори е Motorola, която в средата на 80-те години произвежда първият 8-разряден микроконтролер от фамилията 6805.

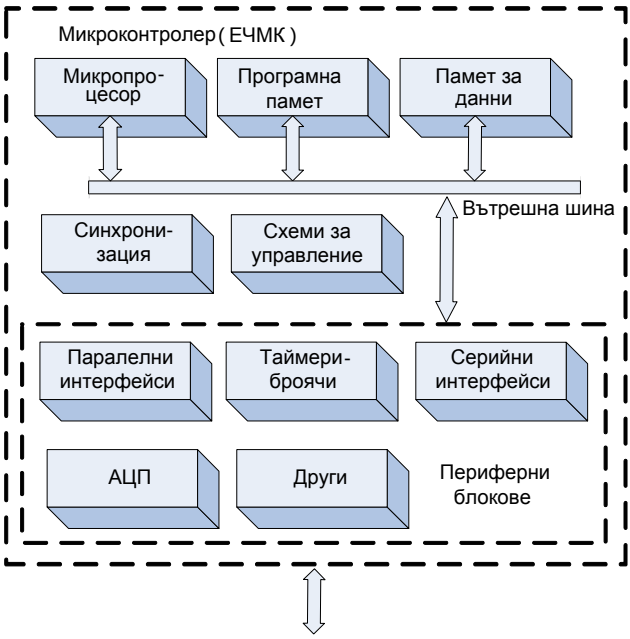
8048 е „завършен" микроконтролер, в смисъла, в който го разглеждаме днес. Но всъщност трябва да се отбележи, че фирмата Texas Instruments проектира схемата TMS1802 за използване в калкулатори, която през 1971 г. е рекламирана за използване в часовници, измервателна апаратура и др., а представената след нея TMS 1000, вече включва RAM, ROM и входно-изходни канали и може да се разглежда като един от първите микроконтролери, въпреки, че не е наричана така.

Съвременните микроконтролери намират приложение в най-разнообразни области, като битовата техника (телевизори, музикални уредби, хладилници, перални, микровълнови печки), в автомобилите, в съвременните средства за връзка, във военната и космическата техника, в промишлеността за управление на технологични процеси и др.

Системите за автоматично управление в промишлеността, например, могат вече ефективно да решават задачи на нива, започващи от управлението на отделни възли и устройства и завършващи с управлението на технологични установки и цели производства. В тези системи се използват различни средства за реализация на алгоритми за управление, като най-ефективни са цифровите методи и средства. „Интелектуализираха" се устройства, изпълняващи и най-прости функции, като измервателни сензори, изпълнителни устройства, средства за сигнализация и др. Стана възможно използването на едни и същи технически средства за решаване на разнообразни задачи, независимо от различните изисквания, реализираните алгоритми и функции. Често изборът на определени параметри, режими и алгоритми на работа трябва да се програмира, т.е. да се определи при специални процедури за настройка. Ето защо съвременните устройства и системи трябва да притежават функционална гъвкавост и възможност за промяна на параметрите.

1.2. Структура на хипотетичен микроконтролер. Класификация, основни характеристики, представители.

1.2.1. Структура на хипотетичен микроконтролер



Фиг. 1.1. Архитектура на хипотетичен микроконтролер

На фиг. 1.1 е показана структурната схема на хипотетичен микроконтролер.

Наименованието „микроконтролер" (microcontroller) отразява функционалността на този тип устройства – миниатюрно устройство за управление. В литературата на български език те се срещат още и като едночипови микрокомпютри (ЕЧМК) – миниатюрен микрокомпютър, разположен в един-единствен чип.

В микроконтролерите се използват принципите на организация на работа и особеностите на архитектурата на класическата компютърна техника. Всеки алгоритъм на функциониране трябва да бъде описан под формата на програма – крайна последователност от елементарни операции.

Сърцето на МК е **микропроцесорът** (МП, централен процесор – ЦП**)**, който изпълнява операциите по обработката на данните, извършва въвеждане и извеждане на всички необходими данни, в това число и машинните кодове на самата програма и управлява взаимодействието на всички елементи на МК.

Наборът от инструкции (изпълняваните от процесора команди) трябва да притежава функционална пълнота, т.е. да осигурява изпълнението на необходимото множество от операции за обработката и обмена на данни и за управление.

За съхраняване на данните и кода на програмата в МК са предназначени **блоковете памет**. Те се състоят от запомнящи елементи (клетки), номерата на които (прието е да се означават в шестнадесетична бройна система) представляват техните адреси. Данните и машинните кодове на програмата се разполагат на последователни адреси в паметта. За извършване на обмен на данни с паметта микропроцесорът трябва да генерира необходимия адрес и управляващи сигнали (например сигнал за посока на данните), извеждани на *адресна шина* и *шина за управление*, а данните се обменят по съответна *шина за данни***.**

Процедурите за въвеждане/ извеждане на данни трябва също да се състоят от стандартни за МП алгоритми. Те обаче в общия случай не съответстват на особеностите на работа на различните периферни устройства. Ето защо МК обикновено съдържа различни блокове за вход-изход, осигуряващи съгласуване на алгоритмите за управление на обмена на данни.

Таймери/ броячи - повечето микроконтролери съдържат един или повече таймери/ броячи, които могат да се използват за реализация на времезакъснения, измерване на времеви интервали, отброяване на събития и др.

Цифрови входове/изходи – броят на входно/изходните изводи варира от 3-4 до над 90 в съвременните МК.

Аналогови I/O: Голяма част от съвременните МК имат блокове АЦП, които се различават по броя на каналите (2-16) и разрешаващата им способност (8-12 бита). Те включват често и аналогов компаратор, а понякога и ЦАП.

Част от МК включват **серийни интерфейси**, които могат да се използват при програмиране, за връзка с ПК и за обмен на данни с други периферни устройства, външни за МК. Поддържат се различни стандарти и протоколи на обмен, като SPI и SCI, I^2C, CAN, PCI, USB или Ethernet.

Някои МК имат допълнителна апаратна част, позволяваща тестване и настройка на работоспособността им и проектираната система дистанционно, чрез персонален компютър.

Разнообразието от едночипови микрокомпютри в днешно време е изключително голямо. Множество фирми произвеждат МК с различни параметри, характеристики и функционални блокове - Intel, NXP Semiconductors, Microchip, Zilog, Texas Instruments, Renesas Electronics, Infineon и много други. Те обединяват своите продукти в серии или фамилии, базирани на еднотипни ядра, ориентирани към различни области на приложение – автоматизирани промишлени системи, комуникации, медицинска апаратура, роботика, автомобилна промишленост и др. Поради това е твърде трудоемко да се направи подробен анализ, който да обхваща всички типове МК, произвеждани в момента.

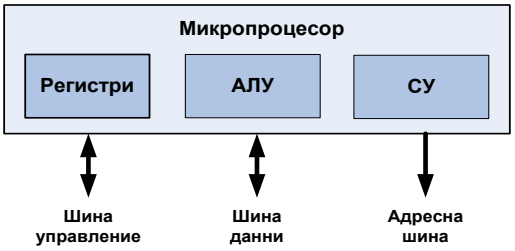
Според областите на приложение, за които са предназначени, МК могат да се разделят най-общо на две основни групи:

- *С общо предназначение;*
- *За специални цели* – например за цифрова обработка на сигнали.

Тъй като архитектурата, функционалните възможности и характеристиките на съвременните микроконтролери непрекъснато се обогатяват и развиват, при избора на МК за конкретно приложение проектантът разполага с големи възможности, съобразявайки се с определени критерии.

1.2.2. Структура на хипотетичен микропроцесор

На фиг. 1.2 е показана структурната схема на хипотетичен микропроцесор. Тя съдържа три основни компонента. Данните между тях се обменят по вътрешна шина. Предназначението им е следното:



Фиг. 1.2. Структурна схема на микропроцесор

- **АЛУ** - Аритметично-логическо устройство. Изпълнява една от най-важните функции – обработката на данните, като извършва аритметични и логически операции. Има два или повече входа и един изход. На входовете си има буферни регистри за временно съхраняване на постъпващите данни. Към АЛУ могат да постъпват данни или от вътрешната шина за данни, или от специален регистър, наречен *акумулатор*. Броят и видът на операциите, които може да изпълнява АЛУ е различен за различните микропроцесори.
- **Регистри –** участват в реализацията на основните функции на МП. Броят и предназначението им зависи от архитектурата на конкретния МП, но почти всеки МП има шест типа регистри:

○ **за състояние** - съдържа тригер-флагове, които се актуализират автоматично при изпълнение на текущата инструкция и по такъв начин могат да се използват за организиране на условни преходи в програмата.

○ **буферни –** за временно съхраняване на данни за обработка от АЛУ;

○ **за инструкции –** предназначен за съхраняване на текущата изпълнявана команда (инструкция);

○ **за адрес в паметта за данни –** съдържа адрес на клетка от паметта, която предстои да се използва;

○ **програмен брояч –** регистър, който във всеки момент съдържа адреса на инструкцията, която предстои да се изпълнява;

○ **акумулатор(и) –** регистър, съдържащ данните за обработка от АЛУ.

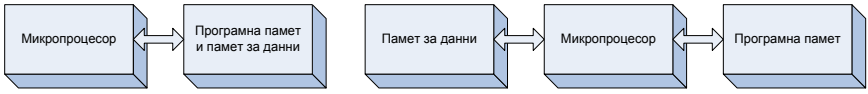
- **СУ –** Схема за управление, изпълнена като краен (микропрограмен) автомат и предназначена за изработване на управляващи сигнали за изпълнение на отделните микрооперации.

Принцип на действие на МП - МП извлича (чете) инструкция от паметта, дешифрира я и изпълнява съответната операция с операнди, намиращи се в регистри, предназначени за съхранение на операндите. В същото време ПБ увеличава съдържанието си, за да адресира следващата инструкция.

1.2.3. Архитектура на паметта Фон Нойман (Принстън) и Харвард. RISC и CISC архитектура на микропроцесорите.

Архитектури на паметта фон Нойман и Харвард

През 40-те години на миналия век правителството на Съединените щати ангажира два университета - Харвард и Принстън, да предложат компютърна архитектура, която да се използва от морската артилерия за изчисляване на разстояния при променливи условия. Двете разработени в последствие архитектури се различават основно по начина на взаимодействие на микропроцесора с паметта (фиг. 1.3). Архитектурата на Принстън включва единно адресно пространство за програмен код и данни, като става по-късно по-известна с името на учения - ръководител на проекта John Von Neumann. За разлика от нея в архитектурата на Харвард се използват отделни банки памет за съхранение на програмата, стека и RAM памет за данни.



а) *б)*

Фиг. 1.3. Компютърна архитектура Фон Нойман (а) и Харвард (б)

Предложената от Принстън архитектура (архитектура фон Нойман) печели състезанието поради развитието на технологиите в дадения момент – единното пространство памет е по-надеждно и изисква по-прост и сигурен интерфейс.

(Това е времето, преди изобретяването на транзистора!). Използването на обща шина за програмен код и данни значително опростява процесите на тестване и настройка на функционирането на системата. Освен това тя е по-гъвкава - цялото адресно пространство може оперативно да се разделя на области за програмен код, стек и данни. Неин основен недостатък е невъзможността за едновременен достъп до двата типа памети, което намалява бързодействието.

Харвардската архитектура е почти забравена до късните 70 години на XX век, когато производителите на микроконтролери оценяват предимството на паралелния достъп до паметта за инструкции и тази за данни. Този паралелизъм позволява извличането на инструкция да става едновременно с изпълнението на предишната. Недостатъци на Харвардската архитектура са различната понякога разрядност на запомнящите елементи (както е при МК на Microchip), което усложнява използването на програмната памет, например за запис на таблични данни, по-голям брой шини, липсата на гъвкавост при разпределението на адресното пространство – налага се да се използва по-голямо количество памет за код и данни, което не може да се преразпределя. Част от тези недостатъци при съвременното развитие на технологиите са преодолени и тя е все повече използвана при съвременните микроконтролери. Тя е подходяща за процесори, които не са предназначени за обработка на големи количества данни (за които архитектурата фон Нойман е по-подходяща - например сървъри и работни станции), и за които бързият достъп до по-малко количество памет е твърде важен.

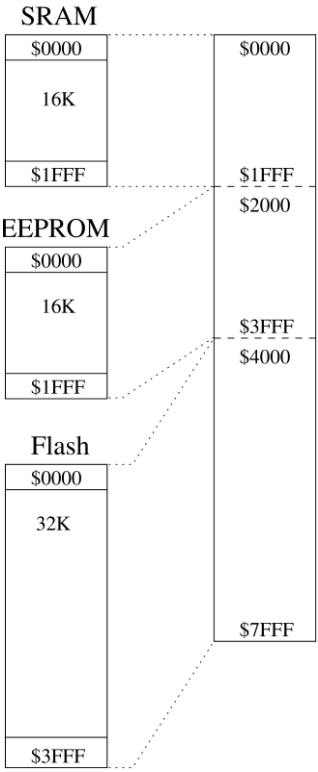
И така, съществуват два начина за „конструиране" на общото адресно пространство на микропроцесорна система:

- **При архитектура Фон Нойман** адресните пространства на различните видове памети съставляват общо адресно пространство с последователно разположени адреси, без да се припокриват (например при фамилията HCS12 на фирмата NXP Semiconductors - фиг. 1.4).
- **При архитектура Харвард** всяка памет се адресира поотделно (например при ATtiny1627 на фирмата Microchip - фиг. 1.5).

В този случай общото адресно пространство на МПС е съставено от отделни адресни пространства за всеки тип памет, като диапазоните на адресите на различните адресни пространства се припокриват частично или напълно (в зависимост от обемите на паметите).

RISC и CISC архитектура

От друга страна съществуват два типа микропроцесори от гледна точка на тяхната сложност на архитектурата и оттам набора инструкции, които могат да изпълняват.



Фиг. 1.4. Единно адресно пространство на МПС, без припокриване на адресите

Първият вариант е сравнително сложен МП, способен да изпълнява голям набор от операции, кодирани в множество инструкции – CISC (Complete Instruction Set Computer - компютър с пълен набор инструкции) архитектура. В следствие на сложността на структурата му, извършваните операции се изпълняват с разклонения, включващи предварителна подготовка и изпълнение на сложни, а оттам и забавящи целия процес операции. Друга особеност на тази архитектура е и тази, че отделните инструкции имат различно ниво на сложност – простите действия се кодират с кратък код, заемащ един-два байта памет, който се изпълнява бързо, а сложните – няколко байта, изпълнявани по-бавно.

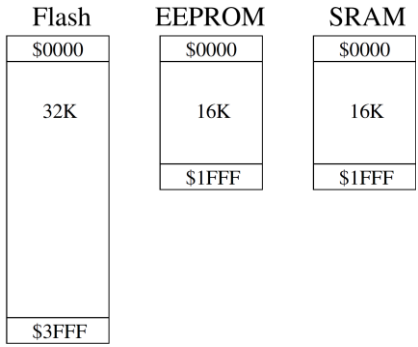
Другият вариант е прост МП, изпълняващ ограничен набор от по-елементарни инструкции – RISC архитектура (Reduced Instruction Set Computer - компютър с намален набор инструкции). Характерно за RISC микропроцесора е това, че едно двоично число, записано в клетка от програмната памет съдържа, както машинния код на инструкцията (указание за действието, което трябва да извърши МП), така и информация за обекта (обектите), над който ще се извърши това действие - адрес на операнд и/или операнд. Тоест всички инструкции са с размерност една програмна дума. Освен това повечето от тях се изпълняват за едно и също кратко време.

1.3. Вградени микропроцесорни системи

1.3.1. Структура на ВМПС

В ролята си на управляващи устройства в разнообразни приложения микроконтролерите изпълняват редица функции, най-често са свързани със:

- събиране и обработка на информация от наблюдаван и/или управляван обект;
- генериране на управляващи въздействия под формата на сигнали с различни характеристики и параметри;
- изобразяване на информация от различен характер;
- комуникация с други устройства и системи и др.

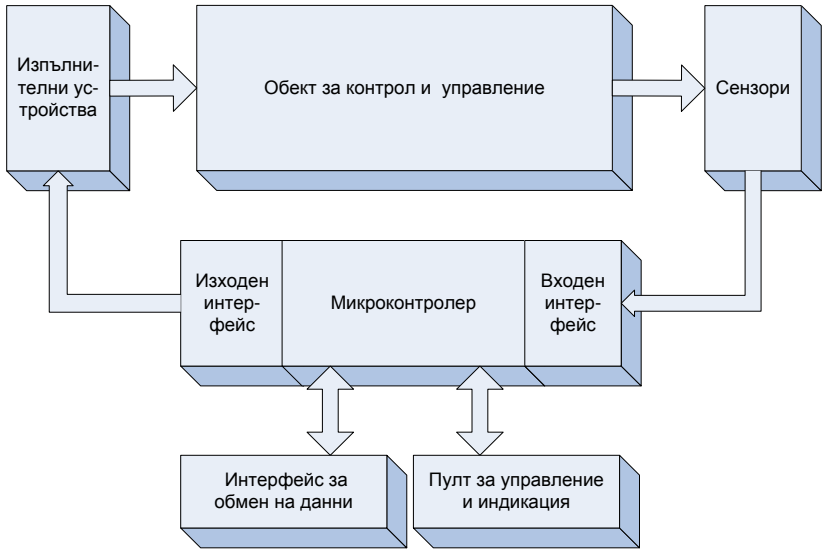


Фиг. 1.5. Адресно пространство на МПС, съставено от отделни адресни пространства на различните типове памети

При реализацията на тези свои функции, те се нуждаят от допълнителни възли, блокове, устройства, като най-често се "вграждат" в управляваното устройство или система.

Ето защо устройствата и системите, реализирани на базата на микропроцесори и микроконтролери са придобили в съвременната литература наименованието **Вградени микропроцесорни (микрокомпютърни) системи (ВМПС).**

Най-общата архитектура на една ВМПС включва сензори, микроконтролер и изпълнителни устройства (фиг. 1.6). Ядрото на съвременните ВМПС е най-често микроконтролер. Той получава информация от сензорите, прилага управляващ алгоритъм и въздейства обратно на управлявания обект чрез изпълнителните механизми.



Фиг. 1.6. Обобщена структурна схема на ВМКС

Основните функции на ВМКС са следните:

- **Входен интерфейс** – приемане и преобразуване на сигналите от датчиците във формат, удобен за по-нататъшна обработка;
- **Изходен интерфейс** – преобразуване на изходните данни в сигнали за управление на изпълнителните устройства;
- **Микроконтролер** - основен елемент на системата за управление, реализиращ алгоритмите за управление и обработката на данните в съответствие с поставените задачи, потока данни от входния интерфейс, управляващите команди от пулта за управление и интерфейса за обмен с други средства за управление;
- **Пулт за управление и индикация** - средства за управление и изобразяване на данни за параметрите и режимите на работа за оператора на обекта за управление;
- **Интерфейс за обмен на данни** - средства за организация на взаимодействието и координация на работата на системата за управление с други средства за управление, който в локалните системи за управление не е задължителен, по правило трябва да се предвижда с цел възможна надстройка на системата.

ВМКС може да бъде независима локална система или интегрирана в по-сложна такава, състояща се от много нива.

Интерфейсите за обмен на данни, използвани при ВМПС, както и интерфейсът с потребителя/ оператора варират изключително според предназначението на системата. Възможно е изобщо да липсва потребителски интерфейс, когато системата е предназначена да извършва точно определена функция; да използва елементарен интерфейс под формата на бутони, светодиоди и буквено-цифрова индикация, като 7-сегментна и/или LCD (Liquid Cristal Display) индикация и/или да включва сложен графичен потребителски интерфейс, управляван от съвременна операционна система за персонален компютър. В наши дни световната мрежа предоставя на проектантите дори възможността потребителският интерфейс да бъде отдалечен от самата вградена система, която да ползва мрежата за обмен на данни с него.

Съвременните МК поддържат разнообразие от интерфейси за обмен на данни, като RS-232, SPI/I^2C, USB, CAN, LIN, Ethernet, IrDA и много други.

1.3.2. Класификация на ВМПС

Въпреки голямото разнообразие в предназначението, функциите, структурата и др., ВМПС могат да се класифицират най-общо по следните признаци:

- **В зависимост от архитектурата на ВМПС:**
  - автономни ВМПС;
  - мрежи от ВМПС (разпределени и йерархични).

- **В зависимост от типа на използвания процесор -** ВМПС на базата на**:**
  - Микропроцесори (остаряла практика);
  - Едночипови микрокомпютри с общо предназначение:
    - с Harvard (RISC - Reduced Instruction Set Computer) архитектура, напр. ЕЧМК на фирмата Microchip, фамилията AVR на Atmel и др.
    - с Von Neumann (CISC - Complete Instruction Set Computer) архитектура, напр. фамилията MCS-51 на Intel, MC68HC11 на Motorola и др.
    - комбинирана архитектура, напр. XC16x (Infineon).
  - Едночипови микрокомпютри за специални цели – за цифрова обработка на сигнали, за управление на монитори и др.;
  - Едночипови микрокомпютри, съчетаващи възможностите на тези с общо предназначение и цифрова обработка на сигнали, напр. XC166 (Infineon).
- **В зависимост от набора вградени блокове и функционални възможности на използваните ЕЧМК:**
  - с детерминирана архитектура (неподлежаща на модификация структура) – на базата на „класически" ЕЧМК с общо предназначение и/ или DSP процесори;
  - с гъвкава архитектура – ЕЧМК, вграждани в FPGA.
- **В зависимост от сложността на управлението на обекта,** съответно изискван обем изчислителни операции и/или такива за събиране и обработка на данни:
  - с ниска сложност – техника в бита и офиса, охранителни системи, индустриални процеси и др.;
  - с висока сложност – такива, при които се налага голям обем сложни изчислителни операции и използване на специализирани ЕЧМК, като процесори за цифрова обработка на сигнали (DSP), като комуникации, авиационна техника и др.; такива, към които се отправят строги изисквания за надеждност и сигурност, като авиационна, космическа, автомобилна, медицинска техника и др.

Въпроси за самоконтрол

1. Опишете структурата и предназначението на основните компоненти на хипотетичен микропроцесор.
2. Какво представляват микропроцесорните архитектури на паметта фон Нойман и Харвард и какви са техните предимства и недостатъци?
3. Какво представляват микропроцесорните архитектури RISC и CISC?
4. Опишете структурата и предназначението на основните компоненти на хипотетичен микроконтролер.?
5. Какви видове микроконтролери по функционален признак има?

6. Избройте някои важни характеристи и ресурси на микроконтролерите.
7. Опишете структурата и предназначението на основните компоненти на вградена микропроцесорна система. Дайте примери за области на приложение.

2. Фамилии МК на фирмата Microchip. Представители на фамилията PIC16 – архитектура, функционални възможности и ресурси, принцип на работа

PIC микроконтролерите са проектирани първоначално от фирмата General Instruments за изпълнение на прости задачи - от тук и названието им Periferal Interface Controller. През 70-те години на миналия век фирмата пуска на пазара процесорите PIC 1650 и 1655. Въпреки своите несъвършенства, те могат да работят в напълно автономен режим и съдържат някои новости. Простият централен процесор е изпълнен по RISC архитектура с работен регистър и само 30 инструкции. Изводите имат възможност да пропускат ток като изходи и като входове със значително по-голяма стойност, отколкото съществуващите по това време процесори. Още по това време се проявяват някои от характеристиките на тези МК – простота, автономност на работата, високо бързодействие и ниска цена.

По късно фирмата General Instruments продава подразделението, занимаващо се с полупроводници. През 90-те години се появяват нови, конкурентно способни МК. За разлика от другите производители по това време, фирмата Microchip разработва свои средства за развойна дейност, които са прости и евтини, а в някои случаи и безплатни.

2.1. Фамилии микроконтролери на фирмата Microchip. Основни характеристики и ресурси

В днешно време фирмата Microchip произвежда голямо разнообразие от 8-, 16- и 32-битови микроконтролери с общо предназначение и такива за цифрова обработка на сигнали. Всяка от тези три групи включва по голямо разнообразие от фамилии микроконтролери с различни типове микропроцесорни ядра (PIC, MIPS, AVR, ARM и др). МК от определена фамилия се базират на едно и също ядро, сходни ресурси, ефективност.

В табл. 2.1 са обобщени ресурсите на фамилиите 8-битови МК с PIC ядро. В зависимост от приложението - необходимите градивни блокове, габаритни размери, цена и т.н., проектантът има възможност да избира от широк набор от микроконтролери. По-нататък в изложението ще разгледаме по-подробно микроконтролери от групата PIC16(L)F188XX, които представляват съвременни устройства с разнообразен набор от периферни градивни блокове и функционални възможности.

2.2. Основни характеристики и ресурси на МК PIC16(L)F188XX

Групата микроконтролери PIC16(L)F188XX включва седем микроконтролера с почти идентични структури, ресурси и характеристики, обобщени в табл. 2.2. Освен тях, всички МК притежават следящ таймер, функция CRC и сканиране на паметта, блок за откриване на преминаване на променливо-токов

Product Family	Pin Count	Program Flash Memory (KB)	RAM (KB)	Data EE (B)	Intelligent Analog						Waveform Control											Logic and Math			Safety and Monitoring			Communications					User Interface			Low Power and System Flexibility			
					ADC (# of bits)	Comp	HSComp	DAC (# of bits)	OPA	SlopeComp / PRG	ZCD	CCP/ECCP	10-bit PWM	16-bit PWM	COG	CWG	NCO	DSM	HLT (8-bit)	Universal Timer	NCO (20-bit)	SMT (24-bit)	RTCC	TEMP/TS	CLC	MULT	MathACC	CRC/SCAN	HLT	WWDT	USART	UART with Protocols	I2C/SPI	USB with ACT	LIN Capable	mTouch® Sensing	HCVD	LCD w/ charge pump	PPS	IDLE/DOZE/PMD	DMA/VI	DIA/MAP
PIC10(L)F3XX	6	384–896 B	0.064	HEF	8								✓			✓	✓				✓			✓	✓										✓						
PIC16F152XX	8–40	3.5–28	0.5–2	–	10							✓	✓			✓	✓	✓	✓					✓							1		✓		✓				✓	✓		✓
PIC12/16 LF155X/6X	14–20	7–14	1.024	HEF	10[3]								✓											✓							✓		✓		✓	✓	✓				
PIC16(L)F145X	14–20	14	1.024	HEF	10	✓							✓			✓								✓							✓		✓	✓	✓	✓						
PIC1X(L)F157X	8–20	1.75–14	1.024	HEF	10	✓		5						✓		✓				✓				✓							✓				✓	✓			✓			
PIC16(L)F153XX	8–48	3.5–28	2.048	HEF	10	✓		5			✓	✓	✓			✓	✓		✓		✓			✓	✓			✓	✓		2		✓		✓	✓			✓	✓		✓
PIC1X(HV)F752/53	8–14	1.75–3.5	0.128	–	10		✓	5/9	✓	SC		✓			✓				✓									✓							✓						
PIC1X(L)F161X	8–14	3.5	0.256	HEF	10	✓		8			✓	✓				✓			✓			✓		✓			✓	✓	✓	✓					✓						
PIC16(L)F161X[3]	14–20	7–14	1.024	HEF	10	✓		8			✓	✓	✓			✓			✓			✓		✓	✓		✓	✓	✓	✓	✓		✓		✓	✓			✓			
PIC18-Q40/1	14–20	16–32	1–4	512	12[4]	✓		8	✓[5]		✓	✓		✓		✓	✓	✓	✓			✓		✓	✓	✓		✓	✓	✓	2	1	✓		✓	✓	✓		✓	✓	✓	✓
PIC16(L)F170X/71X	14–40	3.5–28	2.048	HEF	10		✓	5/8	✓		✓	✓	✓		✓		✓				✓			✓	✓						✓		✓		✓	✓	✓		✓			
PIC16(L)F176X/7X	14–40	7–28	2.048	HEF	10		✓	5/10	✓	✓	✓	✓	✓	✓	✓			✓	✓	✓				✓	✓			✓	✓		✓		✓		✓	✓			✓			
PIC16(L)F183XX	8–20	3.5–14	2.048	256	10	✓		5				✓	✓			✓	✓	✓			✓			✓	✓						✓		✓		✓	✓			✓	✓		
PIC16(L)F184XX	14–28	7–28	2.048	256	12[4]	✓		5			✓	✓	✓			✓	✓		✓		✓	✓		✓	✓					✓	✓		✓			✓	✓		✓	✓	✓		✓
PIC16(L)F188XX	28–40	7–56	4.096	256	10[4]	✓		5			✓	✓	✓			✓	✓	✓	✓		✓	✓		✓	✓			✓	✓	✓	✓		✓			✓	✓		✓	✓		
PIC18-Q10	28–40	16–128	1–3.6	256-1K	10[4]	✓		5			✓	✓	✓			✓		✓	✓					✓	✓	✓		✓	✓	✓	2		✓		✓	✓	✓		✓	✓		
PIC18-Q43	28–48	32–128	2–8	1024	12[4]	✓		8			✓	✓		✓		✓	✓	✓	✓			✓		✓	✓	✓		✓	✓	✓	4	1	✓		✓	✓	✓		✓	✓	✓	✓
PIC18-Q84[6]	28–48	64–128	8–13	1024	12[7]	✓		8			✓	✓		✓		✓	✓	✓	✓	✓	✓	✓		✓	✓			✓	✓	✓	3	2	✓		✓		✓		✓	✓	✓	✓
PIC16(L)F191XX	28–64	14–56	4.096	256	12[4]	✓		5				✓	✓	✓		✓			✓			✓	✓	✓	✓				✓	✓	✓		✓		✓	✓	✓	✓	✓	✓		✓
PIC18-K40	28–64	16–128	3.728	256–1K	10[4]	✓		5			✓	✓	✓			✓		✓	✓					✓		✓		✓	✓	✓	5		✓		✓	✓	✓		✓	✓		
PIC18-K42	28–48	16–128	8.192	256–1K	12[4]	✓		5			✓	✓	✓			✓	✓	✓	✓		✓	✓		✓	✓	✓		✓	✓	✓	1	1	✓		✓	✓	✓		✓	✓	✓	✓
PIC18-J94	64–100	32–128	4.096	–	12	✓						✓											✓		✓						4		✓	✓	✓	✓		✓	✓	✓		

Notes: (1) In addition to standard 8-bit and 16-bit timers (2) Independent Dual ADC Modules (3) PIC16F1615/9 include an angular timer. (4) ADCC: Analog-to-Digital Converter with Computation (5) PIC18-Q41 has an OPAMP
(6) CAN-FD & JTAG capable (7) Analog-to-Digital Converter with Computation and Context Switching

Табл. 2.1. Основни ресурси на 8-битови микроконтролери с PIC микророчесорно ядро

Табл. 2.2. Основни характеристики и ресурси на МК PIC16(L)F188XX

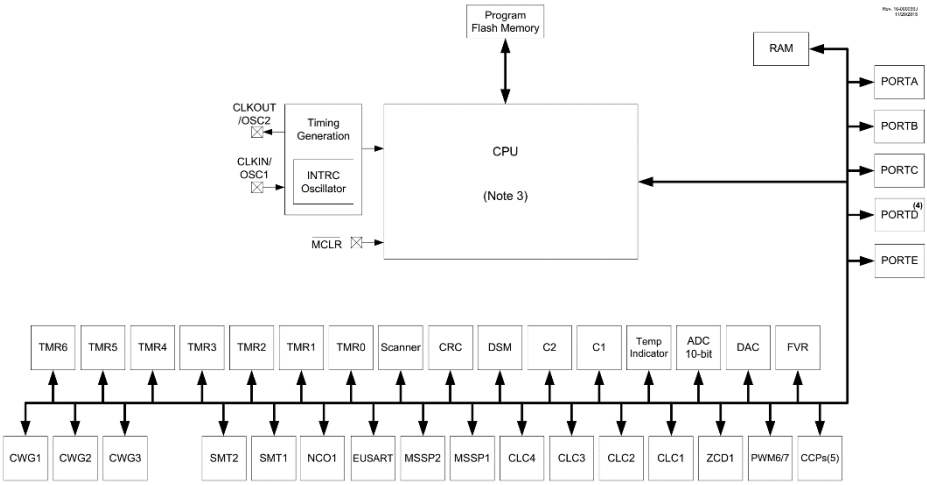
	Прог. Flash памет (kB)	EEPROM	SRAM за данни	I/O изводи	10-битов АЦП (бр.	5-битов	Компаратор	8-битови/16-битови	SMT	CCP/10-битов ШИМ	CWG	NCO	CLC	DSM	EUSART/ I2C/SPI
PIC16(L)F18854	7	256	512	25	24	1	2	3/4	2	5/2	3	1	4	1	1/2
PIC16(L)F18855	14	256	1024	25	24	1	2	3/4	2	5/2	3	1	4	1	1/2
PIC16(L)F18856	28	256	2048	25	24	1	2	3/4	2	5/2	3	1	4	1	1/2
PIC16(L)F18857	56	256	4096	25	24	1	2	3/4	2	5/2	3	1	4	1	1/2
PIC16(L)F18875	14	256	1024	36	35	1	2	3/4	2	5/2	3	1	4	1	1/2
PIC16(L)F18876	28	256	2048	36	35	1	2	3/4	2	5/2	3	1	4	1	1/2
PIC16(L)F18877	56	256	4096	36	35	1	2	3/4	2	5/2	3	1	4	1	1/2

сигнал през нулата, възможност за избор на изводи за периферни функции и опция за забрана на периферен модул.

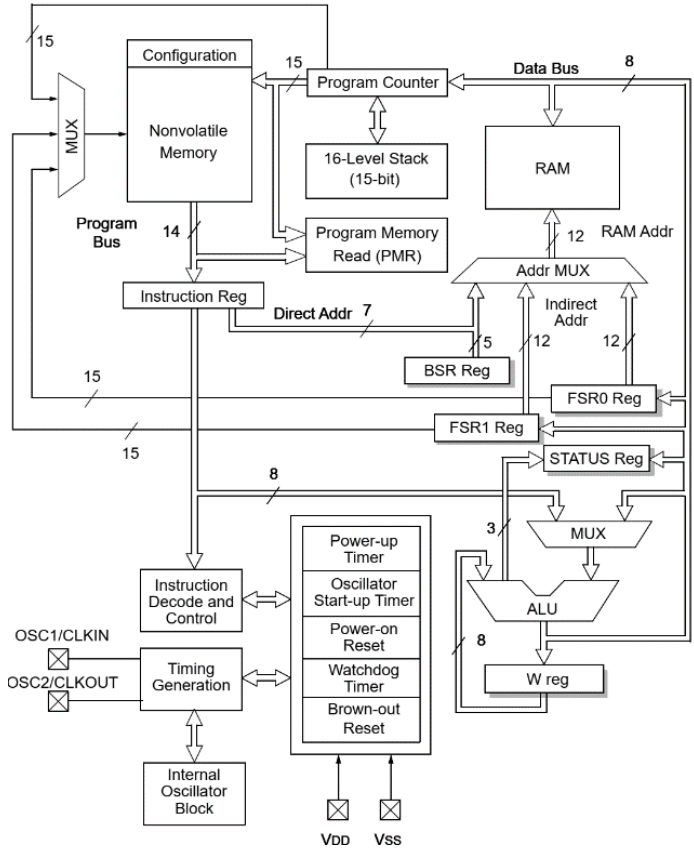
2.3. Блокова схема на МК PIC16(L)F18855/75

На фиг. 2.1 е дадена блоковата схема на МК PIC16(L)F18855/75, а на фиг. 2.2 – блоковата схема на ядрото на микроконтролера.

МК от фамилията PIC16 имат архитектура на МП тип Харвард и 8-битова магистрала за данни (т.е. максималната разрядност на данните, които обработват е 8 бита). Програмната магистрала е с различна разрядност при отделните фамилии, като при PIC16 от среден клас е 14-разрядна (т.е. програмните думи



Фиг. 2.1. Блокова схема на МК PIC16(L)F18855/75



Фиг. 2.2. Блокова схема на ядрото на МК PIC16(L)F18855/75

са 14-битови).Централният процесор изпълнява 49 инструкции. При адресиране на паметта за данни може да се използва пряко, непряко и относително адресиране. Два специални регистъра – FSR, служат за непряко адресиране на програмната памет и паметта за данни.

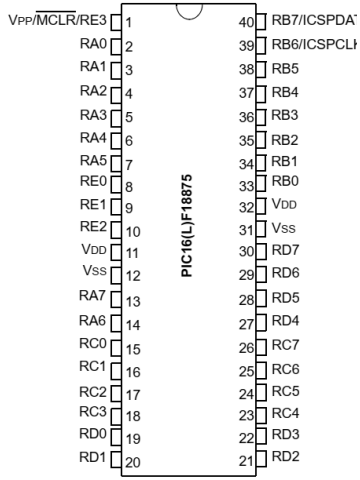
Микроконтролерите имат стекова памет с 16 нива, с автоматично съхраняване на контекста при прекъсване и опция за начално установяване (НУ) при препълване отгоре и отдолу.

Извличането и изпълнението на инструкциите се извършва, като МП изпраща адреса им по 15-битовата магистрала, в резултат на което 14-битовата дума на инструкцията се извежда на програмната магистрала и се записва в регистъра на инструкциите. Той служи за временното й съхраняване, преди тя да бъде декодирана и изпълнена.

Декодирането се състои в определяне (на базата на извлечения код на операцията - КОП) на вида на операцията, която процесорът трябва да изпълни. Ако при изпълнението на инструкцията се наложи четене/ запис в RAM паметта, при пряко адресиране, това става, като първо по 7-битовата магистрала се изпрати адресът на клетката, в която
ще се чете или записва, а след това чрез шината за данни се извърши конкретната операция (четене или запис).

В зависимост от инструкцията, която се изпълнява, в операцията, извършвана от АЛУ, могат да участват един или два операнда. При операции с два операнда единият се намира в регистър W (който изпълнява ролята на акумулатор), а вторият е операнд от инструкцията или от RAM паметта. При операции с един операнд, той се намира или в регистър W, или в RAM паметта. След извършване на операцията, в зависимост от инструкцията, резултатът отново се записва или в W, или в RAM паметта.

Микроконтролерите PIC16(L)F18855/75 се произвеждат в разнообразни правоъгълни и квадратни типове корпуси, като на фиг. 2.3 е показано разположението на изводите им в 40-изводен PDIP корпус.



Фиг. 2.3. Разположение на изводите на МК PIC16(L)F18855/75 в PDIP корпус

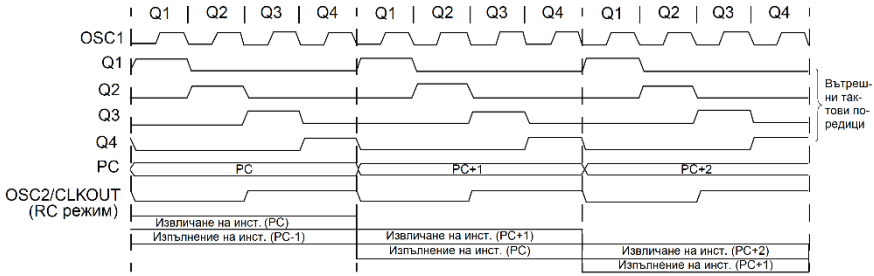
2.4. Цикъл на инструкция. Конвейеризация на инструкциите

Съчетаването на RISC и Харвардската архитектури при PIC микроконтролерите дава възможност инструкциите на МП да се изпълняват в конвейерен режим. В процеса на изпълнение на инструкциите от процесора всяка инструкция трябва първо да бъде извлечена от програмната памет и след това изпълнена.

Харвардската архитектура позволява тези две действия да се изпълняват паралелно. Конвейерното изпълнение на инструкциите също така е ефективно и тъй като при RISC архитектурата циклите за извличане и изпълнение са винаги с еднаква продължителност.

Бързодействието на микроконтролера зависи от честотата на тактовия генератор. Системният тактов сигнал се използва освен за работата на МП, и за други важни функции при работата на периферните блокове.

При МК от фамилията PIC16 сигналът от тактовия генератор (от входа OSC1) се разделя вътрешно на 4, като се получават четири неприпокриващи се импулсни пореди, означени с Q1, Q2, Q3 и Q4. Тяхната съвкупност представлява "цикъл на инструкция" (фиг. 2.4).



Фиг. 2.4. Цикъл на инструкция

Цикълът на инструкция представлява основна единица за време при работата на процесора.

В табл. 2.3 е показана продължителността на един цикъл на инструкция при няколко примерни тактови честоти.

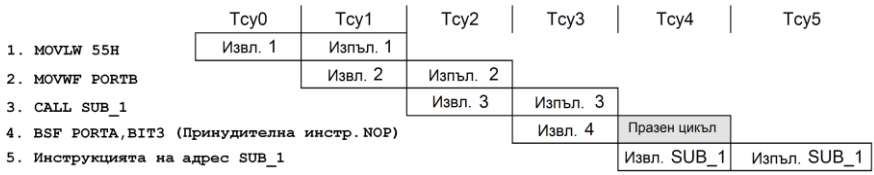
Табл. 2.3. Продължителност на цикъл на инструкция при различни тактови честоти

Тактова честота	Цикъл на инструкция	
	Честота	Период
20 MHz	5 MHz	200 ns
4 MHz	1 MHz	1 μs
1 MHz	250 kHz	4 μs
32,768 kHz	8,192 kHz	122,07 μs

Програмният брояч PC увеличава съдържанието си с 1 на всеки цикъл Q1. В този момент започва извличането на инструкция от програмната памет и завършва по време на цикъла Q4. Тя се декодира и изпълнява по време на следващите цикли Q1 до Q4. Извлечената инструкция се буферира в регистъра на инструкциите по време на цикъла Q1. След това тя се декодира и изпълнява по време на циклите Q2, Q3 и Q4. Паметта за данни се чете по време на цикъла Q2 и се записва по време на Q4.

Тъй като в същото време, когато тя се изпълнява, се извлича следващата инструкция, то изпълнението на една инструкция отнема един цикъл.

Ако инструкцията е за преход (напр. CALL), тогава са необходими два цикъла (фиг. 2.5). Това е така, защото едновременно с декодирането и изпълнението ѝ, се извлича инструкцията, разположена на следващия адрес в програмната памет. Налага се тя да се игнорира и да се извлече друга инструкция – тази на посочения адрес в инструкцията за преход.



Фиг. 2.5. Конвейризация на инструкциите

2.5. Прекъсвания на работата на микропроцесора

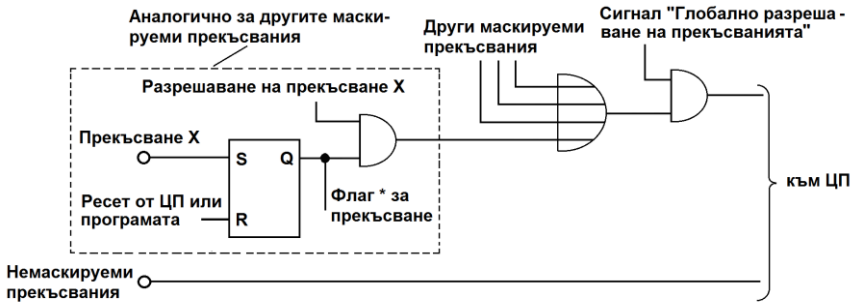
Централният процесор, като изключително добре организиран обект, изпълнява инструкциите в програмата една след друга и прави точно и предсказуемо това, което указват те. Прекъсванията нарушават този ред, като информират процесора за настъпването на събитие, на което е необходимо веднага да се реагира, като при това се прекрати текущата му работа. Първоначално те са използвани за това външни аварийни събития да привлекат вниманието му – отказ на подсистема, прегряване на системата, нарушения на захранването. С течение на времето този механизъм се оказва много удобен за обслужване на различни подсистеми и блокове. Същевременно, от една страна системата за обслужване на прекъсванията става все по-сложна, а от друга – става ясно, че не всички настъпващи събития са еднакво важни, тоест имат различен приоритет.

Използването на прекъсвания предполага познаване, както на апаратния механизъм, така и на програмните методи за обслужването им.

Различните фамилии микроконтролери на различни фирми имат различен механизъм на обслужване на прекъсванията. Общото при тях е, че те имат повече от един източници на прекъсвания и освен това някои от тях са външни за микроконтролера, а други – вътрешни, от вградените периферни блокове на микроконтролера.

Обобщената структура, която илюстрира основните апаратни принципи на прекъсванията, е показана на фиг. 2.6.

Да разгледаме прекъсването X. При постъпването си то се регистрира в RS тригера, дори и да е било с краткотраен характер. Тригерът представлява бит в специален регистър, при което състоянието на изхода му може да бъде прочетено по всяко време. То представлява «флаг за заявка за прекъсване». Изходът на тригера се стробира със сигнал за «разрешаване на прекъсването». Този разрешаващ сигнал постъпва от друг подобен бит (тригер), програмируем от потребителя. В нашия пример, ако той има стойност «1», сигналът за прекъсване достига до логическия елемент ИЛИ, заедно със заявки за прекъсвания от други източници. От своя страна изходният сигнал от логическото ИЛИ се



* *Бит в специален функционален регистър*

Фиг. 2.6. Обобщена структура на апаратното обслужване на прекъсванията

стробира по подобен начин с управляващ сигнал за общо (глобално) разрешаване на всички прекъсвания (също програмируем от потребителя), след което сигналът за прекъсване X ще достигне до микропроцесора.

Действието за забрана на обслужването на определено прекъсване обикновено се нарича маскиране на прекъсването, а битовете за разрешаване/ забрана - *маски*. Някои микропроцесори и микроконтролери имат прекъсвания, което не могат да се маскират. Те винаги идват от външни източници и са предназначени за реакция на събития от изключителна важност. При МК PIC16 няма немаскируеми прекъсвания.

По-подробно всички аспекти на обслужването на прекъсванията ще бъдат разгледани в *Тема 18: Прекъсвания*. Причината да се спираме на тях сега е, че те, както и флаговете и разрешаващите битове (маски), ще бъдат споменавани в отделните теми, посветени на съставните блокове на разглеждания микроконтролер.

Въпроси за самоконтрол

1. Посочете основни характеристики и градивни блокове на 8-битовите микроконтролери на фирмата Microchip от фамиията PIC16.
2. Колко разрядни са адресните магистрали на паметите за данни и код на микроконтролерите PIC16(L)F18855/75? Какво количество памет максимум могат да адресират?
3. Колко разрядна е програмната дума при PIC16?
4. Кои са основните блокове и какъв е механизмът на функциониране на МК PIC16 и неговия микропроцесор?
5. Какво представлява цикълът на инструкция и какъв е механизмът на конвейеризация на инструкциите при изпълнението им?
6. Какво представляват прекъсванията? Опишете обобщения механизъм на обслужването им?

3. Входно-изходни портове на МК PIC16(L)F18855/75

Задължителен компонент в структурата на едночиповите микрокомпютри са паралелните входно-изходни портове, чрез които се осъществява комуникацията им с външния свят. Те представляват набори от различен брой входно-изходни изводи и съпътстващи ги регистри. При някои фамилии ЕЧМК всички I/O изводи са двупосочни (напр. PIC16/PIC18 и тези с AVR процесорни ядра на Microchip), а при други има както двупосочни, така и само входове или само изходи – според предназначението им (напр. HC11 на NXP Semiconductors).

3.1. Предназначение, конфигуриране и основни характеристики на входно-изходните портове на МК PIC16(L)F18855/75

Входно-изходните портове на МК PIC16(L)F18855/75 са пет. Четири от тях са 8-битови: PORTA, PORTB, PORTC и PORTD. PORTE е 1-битов при PIC16(L)F18855 и 4-битов при PIC16(L)F18875.

Всеки порт има по десет регистъра, свързани с неговата работа:

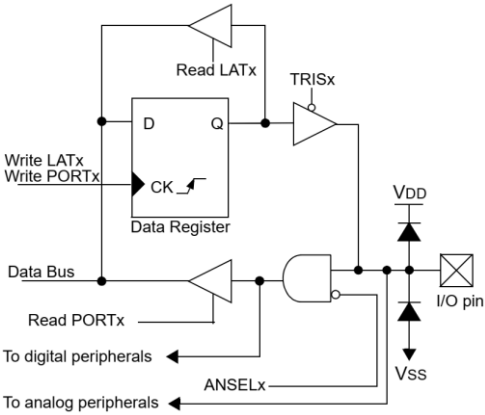
- Регистри за данни PORTx (четат се нивата на изводите на схемата);
- Изходни буферни регистри LATx;
- Регистри TRISx за посока на данните;
- Регистри ANSELx за управление на изводите като аналогови входове;
- Регистри WPUx за управление на изтеглящите към захранване резистори;
- Регистри INLVLx за управление на входните нива;
- Регистри SLRCONx за контрол на стръмността на фронтовете;
- Регистри ODCONx за управление на изходите с отворен дрейн.

На фиг. 3.1 е дадена опростена блокова схема на I/O извод, без да е показан интерфейсът към периферни модули.

Конфигуриране на двупосочните I/O изводи

Всички портове, с изключение на единствения извод на PORTE на PIC16(L)F18855, който е вход, са двупосочни. Конфигурирането им като входове или изходи се извършва чрез запис на подходящи константи в съответните битове на регистри TRISx. Това става, като установяването в единица на определен бит от него конфигурира съответния входно-изходен извод на порта като вход (тоест изходния драйвер); нулирането на бит конфигурира съответния I/O извод като изход (тоест разрешава изходния драйвер и извежда съдържанието на изходния буфер на съответния извод).

Регистрите LATx са полезни при операции от типа „четене-модифициране-запис” при стойности, зависещи от състоянието на самите изводи. Запис в LATx има същия ефект, както запис в PORTx. При четене на LATx се чете



Фиг. 3.1. Блокова схема на входно-изходен извод с общо предназначение

стойността в I/O буфер на порта, докато при четене на регистър PORTx се чете директно състоянието на извода.

Портове, чиито изводи могат да функционират и като аналогови входове, имат и съответен регистър ANSELx. Когато бит от него е установен в 1, буферът на съответния цифров вход е забранен.

В примера по-долу е показано примерно конфигуриране на порт A. Останалите портове се инициализират по аналогичен начин.

Пример 3.1. Инициализация на PORTA

```
BANKSEL PORTA
CLRF    PORTA      ;Инициализация на PORTA
BANKSEL LATA       ;Избор на банката с буфера за данни
CLRF    LATA
BANKSEL ANSELA
CLRF    ANSELA     ;цифрови I/O
BANKSEL TRISA
MOVLW   B'00111000' ;Конфигурира RA<5:3> като входове,
MOVWF   TRISA      ;а RA<2:0> като изходи
```

Функция отворен дрейн

Функцията за изводи с отворен дрейн се управлява от регистри ODCONx. Избира се независимо за всеки извод чрез установяване в 1 на съответния бит. Тя е налична за всички портове, с изключение на PORTE на МК PIC16(L)F18855, който е 1-битов вход.

Контрол на стръмността на фронтовете

Контролът на стръмността на фронтовете се управлява също индивидуално чрез регистри SLRCONx. При установен в 1 бит, контролът е ограничен, докато когато даден бит е нулиран, се задава максимална стръмност. Опцията е налична за всички портове, с изключение на PORTE на МК PIC16(L)F18855.

Управление на прага на входното напрежение

Управлява се от регистри INLVLx индивидуално за всеки извод. Има възможност да се избира между CMOS тригер на Шмит и TTL нива. Входните нива са важни при определяне на прочетените стойности в PORTx, както и за регистриране на прекъсване при промяна на състоянието на извод.

Аналогови функции на изводите на портовете

Те са налични за всички портове, с изключение на PORTE на МК PIC16(L)F18855.

Конфигурирането на изводи като аналогови входове се извършва чрез регистри ANSELx. Установяването в 1 на даден бит в него води до това, при цифрово четене да се чете 0. Състоянието му няма значение при извеждане на цифрови стойности, но в режим на вход ще е аналогов. Това може да доведе до неочаквано поведение пи операции от типа „четене-модифициране-запис“.

Регистри TRISx управляват драйверите на I/O изводи, дори и когато те са конфигурирани като аналогови входове.

Важно е да се знае, че *при начално установяване при включване на захранването тези изводи са конфигурирани като аналогови входове.*

Изтеглящи към захранване резистори на изводите

Управляват се индивидуално чрез регистри WPUx.

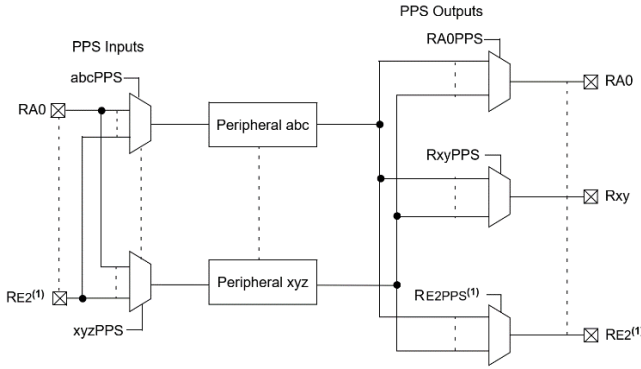
3.2. Блок за свързване на I/O изводи с тези на периферните блокове (Peripheral Pin Select – PPS)

Повечето от I/O изводи на PIC16(L)F18855/75 споделят функциите си с общо предназначение с алтернативни функции, свързани с периферни блокове. Свързването на изводите на портовете с входовете и изходите на периферните блокове се извършва от специален блок – PPS. Той отговаря само за цифровите сигнали. Всички аналогови входове и изходи остават фиксирани към назначените им изводи. Аналоговите входове (за аналогово-цифровия преобразувател и компараторите) се задават чрез регистри ANSELx. Функцията за цифров изход може да продължава да управлява извод, конфигуриран като аналогов изход. Но режимът на аналогов изход има приоритет пред този на цифров изход, като поставя драйвера на цифровия изход във високо-импедансно ниво.

Опростената блокова схема на блок PPS е показана на фиг. 3.2.

Всеки периферен блок има свой PPS регистър, чрез който се задават входовете (изводи на портове) към него. Всеки I/O извод също има свой PPS регистър, чрез който се задава с кой източник е свързан.

Настройките в PPS за периферни блокове с двупосочни сигнали на даден извод трябва да се направят нака, че входовете и изходите на PPS да избират един и същи извод. Такива периферни блокове са EUSART (в режим на синхронен обмен) и MSSP (в режим I^2C).



RD<7:0> и RE<2:0> са съществуващи само при устройствата с 40/44 извода.
RE3 е само вход за PPS (когато MLCR е забранен).

Фиг. 3.2. Блокова схема на блок PPS

PPS включва режим, при който всички селекции на входове и изходи могат да бъдат „заключени“, за да се предотвратят нежелани промени. Това става чрез установяване в 1 на бит PPSLOCKED в регистър PPSLOCK чрез специална процедура, описана в [7].

PPS може да бъде също забранен за постоянно чрез конфигурационен бит PPS1WAY. Когато той е установен в 1, бит PPSLOCKED може да бъде нулиран или установен в 1 еднократно след НУ. Това позволява нулиране на PPSLOCKED, така че да позволи селектирането на входовете и изходите по време на инициализация на устройството. Когато бит PPSLOCKED се установи в 1 веднага, след като бъдат направени всички селекции, той ще остане установен и не може да се нулира преди следващото НУ.

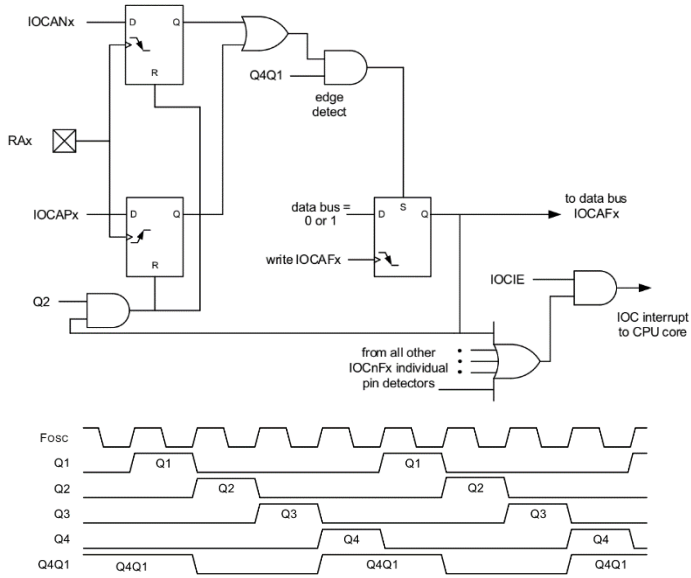
НУ при включване на захранването (Power-On-Reset - POR) нулира всички PPS входни и изходни селекции до техите подразбиращи се стойности, дадени в [7].

3.3. Прекъсване при промяна на състоянието на извод на порт

Всички изводи на PORTA, PORTB, PORTC и извод RE3 на PORTE на МК PIC16(L)F18855/75 могат да бъдат конфигурирани като такива с функция за прекъсване при промяна на състоянието им (interrupt-on-change - IOC). Прекъсване се генерира, както при падащ, така и при преден фронт на сигнала на извода. Могат да се конфигурират един или повече изводи да генерират прекъсване. Всеки изводи си има индивидуален флаг за заявка за прекъсване.

На фиг. 3.3 е дадена блоковата схема на блок IOC.

Разрешаването на прекъсване от промяна на състоянието на извод става чрез установяване в 1 на бит IOCIE в регистър PIE0.



Фиг. 3.3. Блокова схема на блок IOC

Възможно е задаване на регистриране на прекъсване по преден или заден фронт на сигнала. За регистриране на прекъсване по преден фронт съответният бит в регистър IOCxP трябва да е установен в 1. За да се регистрира прекъсване по заден фронт, трябва да се установи в 1 съответният бит в регистър IOCxN. Ако е необходимо да се регистрират прекъсвания и по преден, и по заден фронт, се установяват битовете и в двата регистъра IOCxP и IOCxN.

Битовете в регистри IOCxF представляват флагове за състояния на IOC изводите на всеки порт. Прекъсванията се разрешават чрез установяване на бит IOCIE в 1. Бит IOCIF в регистър PIR0 отразява състоянието на всички битове IOCxF.

Индивидуалните флагове за състояние (IOCxF) могат да се нулират. Ако бъде открит фронт по време на операция за нулиране, съответните флагове за състояние ще се установят в края на последователността, независимо от записаната в тях стойност.

Прекъсване при промяна на състоянието на извод може да изведе МК от режим Sleep, ако бит IOCIE е установен в 1. Ако бъде открит фронт по време на режим Sleep, съответният регистър IOCxF ще се обнови, преди първата инструкция след излизането от Sleep.

Въпроси за самоконтрол

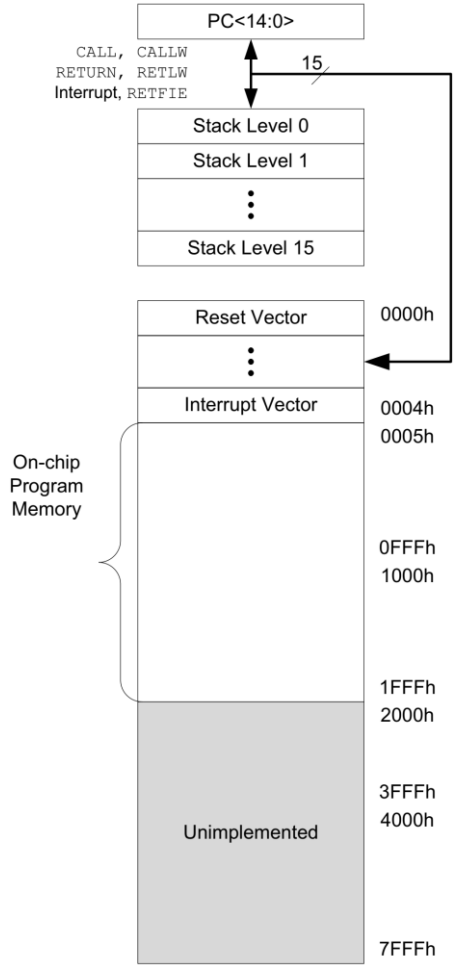
1. Какво е предназначението на паралелните входно-изходни портове?
2. Как се извършва конфигурирането на двупосочните им изводи като входове или изходи?
3. Разгледайте блоковата схема на входно-изходен извод и обяснете процеса на конфигуриране и на четене и запис.
4. Посочете основните характеристики на входно-изходните изводи.
5. Какво е предназначението на блок PPS?
6. В какво се състои регистрирането на прекъсване при промяна на състоянието на извод на I/O порт? Обяснете работата на блок IOC.

4. Организация и адресиране на паметта на МК PIC16(L)F18855/75

Архитектурата на паметта – програмна и за данни на МК от фамилията PIC16, е тип Harvard - всеки блок има своя собствена адресна, даннова и управляваща шина, което ускорява достъпа до тях.

Микроконтролерите PIC16(L)F18855/75 имат следните типове памети:

- *Програмна памет, включваща:*
  - Конфигурационни думи;
  - Идентификационен номер на устройство (Device ID);
  - Потребителски идентификационен номер (User ID);
  - Програмна Flash памет.
- *Памет за данни, включваща:*
  - Регистри на ядрото;
  - Специални функционални регистри (Special Function Registers - SFR);
  - RAM памет с общо предназначение (General Purpose RAM - GPR);
  - Обща RAM памет;
  - EEPROM памет за данни.



Фиг. 4.1. Карта на програмната памет на МК PIC16(L)F18855/75

4.1. Програмната памет

Програмната памет представлява масив от *14-битови клетки*, в които се записват машинните кодове на инструкциите, изпълнявани от централния процесор.

Картата за разпределение на програмната памет на МК PIC16(L)F18855/75 е показана на фиг. 4.1. На нея са означени три области, неразривно свързани функционално: програмният брояч PC, стекът и самата програмна памет.

Програмният брояч (Program Counter – PC) представлява регистър, чието предназначение е

да сочи (съдържа адреса на) инструкцията, която предстои да бъде изпълнена от централния процесор. Тоест той представлява указател в програмната памет. При ЕЧМК от фамилията PIC16 той е 15-битов, от където следва, че може да адресира обем до 32Kx14 програмна памет.

Младшият байт на PC се намира в регистър PCL, който е програмно достъпен за четене и запис. Старшият байт (PC<14:8>) не е програмно достъпен. Той се попълва от програмно достъпния си буферен регистър PCLATH. При всички варианти на начално установяване PC се зарежда с 0.

Съдържанието на PC се променя автоматично, като това зависи от вида на изпълняваната операция. При определени обстоятелства то може също да се записва и извлича в/от стека.

Векторът за начално установяване (НУ) е на адрес 0000h. Това е адресът, от който ще започва изпълнението на програмата винаги при настъпване на събитие, предизвикващо начално установяване, като например включване на захранването на микропроцесорното устройство. Тъй като това е апаратно заложен механизъм, мястото на вектора за НУ не може да се променя. Други причини за начално установяване ще бъдат разгледани в *Тема 17: Причини и условия за начално установяване на микроконтролера*.

На адрес 0004h се намира т.нар. **вектор за прекъсване**. Векторът за прекъсване е мястото, от където започва подпрограмата за обслужване на прекъсванията, т.е. на този адрес трябва да се намира първата инструкция от нея. Както и векторът за НУ, така и векторът за прекъсване (местоположението му) не може да се променя.

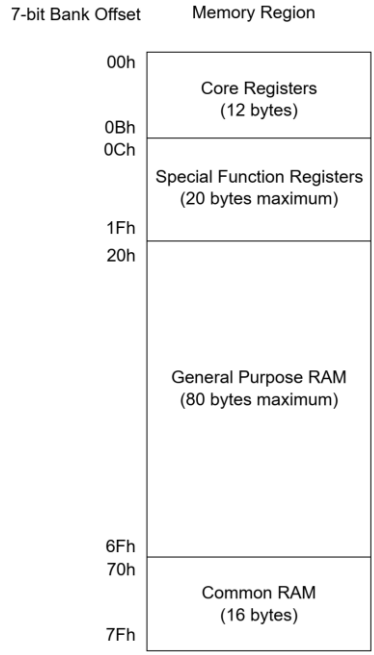
Програмната памет може също да съдържа константни данни. Достъпът до тях може да се осъществява по два начина: чрез използване на таблица от инструкции RETLW, които да връщат константите, и чрез специален адресен регистър – FSR, който се зарежда с необходимия адрес в програмната памет.

4.2. Памет за данни

Паметта за данни на МК PIC16(L)F18855/75 се състои от 8-битови клетки и е разделена на 32 банки, всяка от които съдържа 128 байта. Всяка банка съдържа (фиг. 4.2):

- 12 регистъра на ядрото;
- 20 специални функционални регъстра (SFR);
- До 80 байта RAM с общо предназначение (GPR);
- 16 байта обща RAM.

Паметта за данни се адресира от 12-битови адреси. Разделянето ѝ на банки се налага, тъй като в дадена инструкция за работа с нея се поместват само 7 бита от адреса, чрез които не е възможно да се достъпва цялото налично адресно пространство като линейна структура (вж. фиг. 2.2). Това означава, че във всеки момент имаме достъп само до една банка памет. Изборът на банка става,



Фиг. 4.2. Карта на банка памет за данни на МК PIC16(L)F18855/75

Регистър STATUS

Регистър STATUS съдържа флагове за състояние, които се влияят от работата на аритметично-логическото устройство и други, зависещи от условия за начално установяване на микроконтролера. Структурата и предназначението на битовете му са показана по-долу:

като се запише номерът ѝ в регистър Bank Select Register (BSR). Или старшите 5 бита от 12-битовия адрес представляват номерът на банката, записан в BSR, а младшите 7 бита, които се извличат от инструкцията, са адресът на регистъра.

Цялата памет за данни е достъпна пряко, чрез инструкции, които използват файловите регистри, и непряко, чрез адресните регистри FSR.

Регистри на ядрото

Това са регистрите, които пряко се засягат от основните операции. Те заемат първите 12 адреса на всяка банка от паметта за данни и са показани в табл. 4.1.

Табл. 4.1. Регистри на ядрото

Адреси	Банка x
x00h или x80h	INDF0
x01h или x81h	INDF1
x02h или x82h	PCL
x03h или x83h	STATUS
x04h или x84h	FSR0L
x05h или x85h	FSR0H
x06h или x86h	FSR1L
x07h или x87h	FSR1H
x08h или x88h	BSR
x09h или x89h	WREG
x0Ah или x8Ah	PCLATH
x0Bh или x8Bh	INTCON

Структура и предназначение на регистър STATUS:

U-0	U-0	U-0	R -1/q	R -1/q	R/W-0/u	R/W-0/u	R/W-0/u
-	-	-	$\overline{TO}$	$\overline{PD}$	Z	DC	C
бит 7							бит 0

Легенда:

R - бит за четене — *q – стойността зависи от условие*

W – бит за запис — *U – бит, който не се използва; чете се „0”*

u – не се променя *-n/n – стойност при POR и BOR/*
1 – установен в 1 *стойност при други условия за НУ*
0 – установен в 0

битове 7-5 **Нереализирани** (четат се като 0):

бит 4 $\overline{\textbf{TO}}$: **Таймаут (изтекъл период) на следящия таймер WDT**:
1 = При включване на захранването, при инструкция *CLRWDT* или *SLEEP*
0 = Настъпил таймаут на WDT

бит 3 $\overline{\textbf{PD}}$: **Бит за режим на ниска консумация**:
1 - При включване на захранването или инструкция *CLRWDT*
0 - При изпълнение на инструкция *SLEEP*

бит 2 **Z: Флаг за нулев резултат**:
1 = Резултатът от аритметичната или логическата операция е нула
0 = Резултатът от аритметичната или логическата операция не е нула

бит 1 **DC: Бит за цифров пренос/**$\overline{\text{заем}}$ (при инструкции *addwf, addlw, sublw, subwf*)
1 = Наличие на пренос от 4-ти към 5-ти бит на резултата при събиране или
липса на заем от 5-ти бит при изваждане
0 = Липсва пренос от 4-ти към 5-ти бит на резултата при събиране или наличие на заем от 5-ти бит на резултата при изваждане

бит 0 **C**: **Бит за пренос/**$\overline{\text{заем}}$ (при инструкции *addwf, addlw, sublw, subwf*; влияе се и от инструкции *rrf* и *rlf*).
1= Настъпил е пренос от 7-ми бит на резултата при събиране или липсва заем при изваждане
0 = Липсва пренос от 7-ми бит на резултата или наличие на заем при изваждане

Специални функционални регистри

Специалните функционални регистри се използват за конфигуриране и задаване на желаните режими на работа на периферните блокове. Те заемат 20 байта в областта след регистрите на ядрото във всяка банка памет за данни. Тяхната структура и предназначение на битовете са дадени в [7]. Част от тях ще бъдат описани също по-нататък при разглеждането на съответните периферни блокове.

RAM памет с общо предназначение

Това е област от до 80 байта от GPR регисти във всяка банка памет за данни.

Достъпът до RAM с общо предназначение може да се осъществява също по метод без избор на банка посредством регистрите FSRx.

Обща RAM памет

Общата RAM памет са 16 байта, достъпни от всички банки

Подробната карта на паметта за данни е дадена в [7].

EEPROM памет за данни

Тя е удобна за съхраняване на променливи данни, които е необходимо да бъдат запазени за продължителен период от време и при изключено захранване, например параметри за настройка на телевизора, калибровъчни настройки на измервателен уред и др.

При PIC16(L)F18855/75 EEPROM паметта за данни се състои от 256 8-битови клетки.

EEPROM паметта може да се чете/ записва посредством:

- Регистри FSR/INDF - непряко;
- Регистър NVMREG;
- По време на вътрешно-схемно серийно програмиране (ICSP). За разлика от програмната Flash памет, която се записва по редове, EEPROM може да се записва дума по дума.

4.3. Стек

Стековата памет (стекът) при PIC16(L)F18855/75 се състои се от 16 нива от 15-разрядни клетки. Физически той не е част нито от паметта за данни, нито от програмната памет. При изпълнение на инструкции *call* и *callw* или извикване на подпрограмата за обслужване на прекъсвания, съдържанието на програмния брояч PC се записва в стека и се възстановява при изпълнение на инструкции *return, retlw* или *retfie*.

Той е организиран като цикличен буфер, ако конфигурационен бит STVREN е нулиран. Това означава, че ако в него са записвани данни шестнайсет пъти последователно (без междувременно да са извличани), то при седемнайсти запис поредната стойност ще бъде записана в първо ниво (в първата клетка), като ще припокрие съдържанието там. Осемнайстият запис ще припокрие съдържанието на втората клетка и т.н. Подобна е ситуацията при неконтролируемо последователно многократно четене от стека.

При препълване отгоре (запис повече от 16 пъти) или отдолу (четен повече от 16 пъти) се установяват в 1 съответно флагове STKOVF и STKUNF в регистър PCON, независимо от това, дали е разрешено начално установяване при тези условия. Начално установяване при препълване е разрешено, ако конфигурационен бит STVREN е установен в 1.

Тъй като стекът има изключително ограничен обем, той не може да се използва като буферна памет за различни цели и няма инструкции за запис и извличане на данни в/от него.

Достъпът до стека се осъществява чрез три регистъра: TOSH, TOSL и STKPTR. STKPTR е указател в стека. Чрез двойката регистри TOSH:TOSL, които са програмно достъпни, е достъпен върхът на стека – поредната клетка, в която предстои да се записва или първата, от която ще се чете.

По време на нормалното изпълнение на програмата инструкции CALL, CALLW и прекъсванията инкрементират (увеличават с 1) STKPTR, а RETLW, RETURN и RETFIE декрементират (намаляват с 1) STKPTR. STKPTR може да се прочете по всяко време, за да се установи, колко стек е останал свободен. Той винаги сочи текущата използвана клетка. Това означава, че CALL или CALLW инкрементират STKPTR и след това записват в PC, а при връщане първо се зарежда PC, а след това се декрементира STKPTR.

4.4. Непряко адресиране

Непряката адресация се реализира с използване на регистри INDFn и FSRn, като INDFn не са физически съществуващи регистри. Всяка инструкция, в която като операнд участват регистри INDFn, на практика използва регистъра, указван от регистри FSRn, които предварително трябва да съдържат адреса му. Четенето на самите регистри INDFn непряко връща стойност 00h. Записът в INDFn непряко ще доведе до празна операция (като някои битове за състояние може да се променят). FSRn регистрите формират 16-битов адрес, който позволява адресиране на адресно пространство от 65536 клетки. Те са разделени на три области (фиг. 4.3):

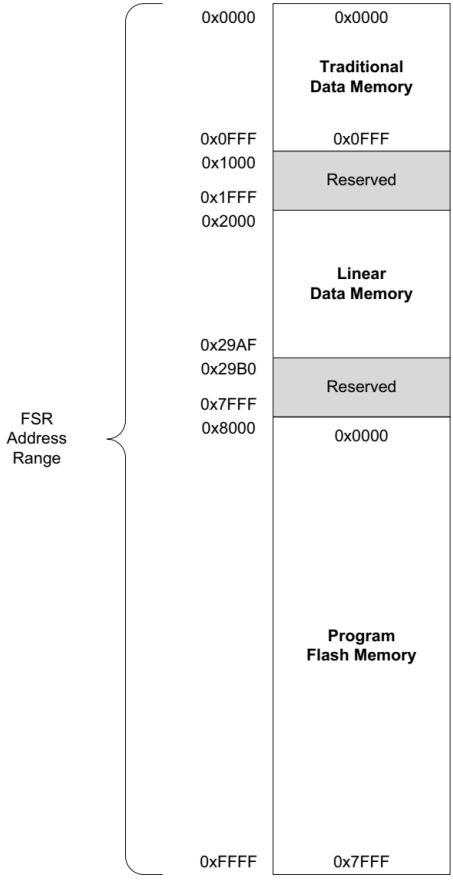
- *Традиционна памет за данни;*
- *Линейна памет за данни;*
- *Програмна Flash памет.*

FSRn могат да служат допълнително и за адресиране на EEPROM паметта за данни при четене.

Традиционната памет за данни включва адресно пространство, адресирано чрез FSR от 0x000 до 0xFFF. Адресите съответстват на абсолютните адреси на всички SFR, GPR и общи регистри.

Линейната памет е областта на FSR адреси от 0x2000 до 0x29AF. Това е виртуална област, която сочи обратно към 80-байтовия блок от GPR паметта във всички банки.

Нереализираната памет се чете като 0x00. Използването на линейна памет за данни позволява буфери, по-големи от 80 байта, тъй като инкрементирането на FSR отвъд дадена банка изпраща директно към GPR паметта в следващата банка.



Фиг. 4.3. Непряко адресиране

EEPROM паметта може да се чете и записва посредством регистри NVMCONx /NVMADRx/ NVMDATx - съответно управляващи, за адрес в паметта и за данни. С цел да се улесни достъпа до EEPROM е възможно също да се осъществява само четене на клетки с непряко адресиране с използване на регистри FSR. Когато старшият байт на FSR (FSRxH) съдържа 0x70, младшата половина на адреса (в FSRxL) определя клетката в EEPROM, от която може да се чете (през регистър INDF). С други думи диапазонът от EEPROM адреси 0x00-0xFF е в FSR-адресируемото пространство 0x7000-0x70FF. Запис в EEPROM не може да се извършва чрез FSR/INDF. Четене от EEPROM чрез FSR/INDF изисква един допълнителен цикъл на инструкция.

За по-лесен достъп до константи в **програмната Flash памет**, тя цялата е разположена в горната половина на FSR адресното пространство. Когато най-старшият бит на FSRnH е установен в 1, младшите 15 бита са адреса в програмната памет, който е достъпен чрез INDF. Само младшите 8 бита на всеки адрес на клетка са достъпни чрез INDF. Запис в програмната Flash памет не може да се изпълни посредством FSR/INDF. Всички инструкции, които осъществяват достъп до програмната Flash памет чрез FSR/INDF изискват един допълнителен цикъл на инструкция, за да завърши изпълнението ѝ.

Въпроси за самоконтрол

1. Колко разрядна е програмната памет и кой регистър се използва за адресирането ѝ?
2. Какво представляват векторите за начално установяване и за прекъсвания и къде се намират?

3. Какво представлява и за какво се използва стекът? Какъв е обемът му? Какво се случва, ако се препълни?
4. Какъв е механизмът на адресиране на програмната памет?
5. Опишете структурата на паметта за данни – колко разрядна е, от какви области се състои, какво е тяхното предназначение?
6. Опишете начините за адресиране на отделните области в паметта за данни.
7. Как се извършва достъпът до EEPROM паметта за данни? Какви регистри се използват?

Раздел II. Блокове за генериране, измерване и управление на сигнали на МК PIC16(L)F18855/75

5. Таймерни блокове

5.1. Приложение на таймерните блокове

Функциите за броене и измерване на времеви интервали са изключително полезни в различни области на електрониката – от преброяване на предмети върху конвейерна лента, хора преминаващи през определена точка и др. до проектиране на измервателна апаратура.

Задължителни компоненти в структурата на МК са един или повече таймера. Таймерните блокове съдържат броячи с различна разрядност, поради което често се среща терминът „таймер/брояч". Освен това броячът може да е сумиращ (напр. при МК PIC16) или изваждащ (напр. при ATmega128 може да се управлява посоката на броене). Микроконтролерите PIC16 съдържат таймери с 8 и 16-разредни броячи.

Да предположим, че периодът на тактовия сигнал на брояча е 1 μs и нямаме възможност да го променяме програмно. Това означава, че към съдържанието му на всяка μs се прибавя единица. Ако стартираме подаването на тактова честота в определен момент, ще можем да измерим времето от този момент нататък, отчитайки стойността в брояча, при това с точност, не по-голяма в нашия пример от 1 μs. Максималният времеви интервал, който ще можем да измерим, ще зависи от разрядността на брояча – ако той е 8-разреден, ще е в състояние да брои до 255, след което ще се препълни.

На този принцип броячът може да се използва за измерване на времеви интервали между две събития. Тези събития могат да са външни за МК, възможно е също първото да е генерирано от МК, а второто да настъпи известно време след първото като реакция на него или пък обратно. Може също да се налага измерване на времето между два импулса или на продължителност на един импулс. В общия случай е необходимо работата на таймер/брояча да се стартира при настъпване на първото събитие и да се спре при настъпване на второто (макар че са възможни и други варианти).

5.2. Блок Таймер 0

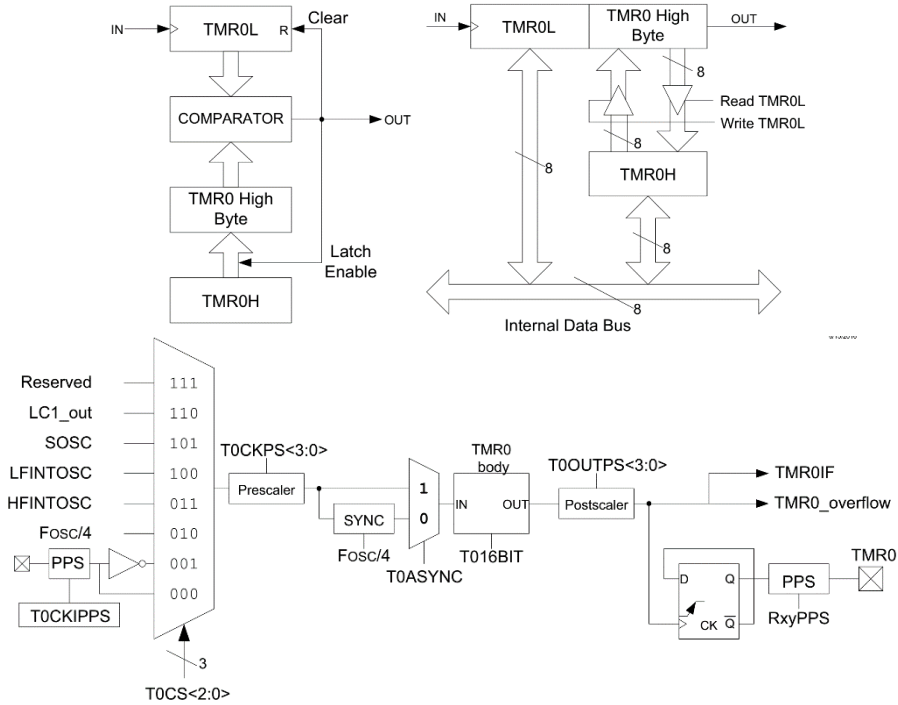
5.2.1. Блокова схема и функционални възможности

Блок Таймер 0 представлява 8/16-битов таймер/брояч със следните функционални възможности:

- Режим на 16-битов таймер/брояч;
- Режим на 8-битов таймер/брояч с програмируем период;
- Синхронна или асинхронна работа;
- Избираем източник на тактов сигнал;

- Програмируем входен делител (независим от този на следящия таймер);
- Програмируем изходен делител;
- Работа по време на режим Sleep;
- Прекъсване при съвпадение или препълване;
- Изход към I/O извод (чрез PPS) или към други периферни блокове.

Блоковата схема на блок Таймер 0 е показана на фиг. 5.1.



Фиг. 5.1. Блокова схема на Таймер 0

5.2.2. Режими на работа

Таймер 0 може да работи като 8-битов или като 16-битов таймер/брояч. Режимът се избира с бит T016BIT на регистър T0CON. Когато се използва с вътрешен източник на тактов сигнал, модулът е в таймерен режим, като увеличава съдържанието на брояча на всеки цикъл на инструкциите. Когато се използва с външен източник на тактов сигнал, модулът може да се използва като таймер или брояч, като увеличава съдържанието на брояча на всеки нарастващ фронт на външния източник.

Режим на 16-битов таймер/брояч

При нормална работа TMR0 се инкрементира по нарастващ фронт на тактовия сигнал. Чрез битове T0CKPS<3:0> в регистър T0CON1 могат да се задават различни стойности за делене на тактовата честота на 15-битовия входен делител.

TMR0H в действителност не е старшият байт на Таймер 0 в 16-битов режим, а негов буферен регистър, който не може да се чете или записва. TMR0H се актуализира със съдържанието на старшия байт по време на четене на TMR0L. Това осигурява възможност за четене на всички 16 бита на брояча, без да се налага да се проверява дали четенето на старшия и младшия байт е валидно, в случай че възникне пренос между последователните четения на старшия и младшия байт.

По аналогичен начин записът в стария байт на брояча също се извършва през буферния регистър TMR0H. Старшият байт се обновява със съдъжанието на TMR0H по време на запис в TMR0L. Това позволява всички 16 бита на брояча да се обновят наведнъж.

Режим на 8-битов таймер/брояч

При нормална работа TMR0 се инкрементира по преден фронт на тактовия сигнал. Чрез битове T0CKPS<3:0> в регистър T0CON1 могат да се задават различни сойности за делене на тактовата честота на 15-битовия входен делител.

Стойността в TMR0L се сравнява с тази в периодния буфер, копие на TMR0H, на всеки тактов цикъл. При съвпадение на двете стойности се случва следното:

- Изходът TMR0_out се установява във високо ниво за един тактов период (зависещ също от стойността на входния делител);
- TMR0L се ресетира;
- Съдържанието на TMR0H се копира в периодния буфер.

В 8-битов режим и двата регистъра TMR0L и TMR0H могат да се четат и записват. Регистърът TMR0L се нулира при всяка причина за НУ, докато регистър TMR0H се инициализира с FFh.

Броячите на входния и изходния делители се нулират при следните събития:

- Запис в регистър TMR0L;
- Запис в регистри T0CON0 или T0CON1;
- Коя да е причина за НУ – при включване на захранването (Power-on Reset - POR), от входа $\overline{\text{MCLR}}$, от следящия таймер (WDTR);
- От смущения в захранването (Brown-out Reset - BOR).

Броячен режим

В режим Брояч, входният делител обикновено е забранен чрез задаване на стойности „0000“битове в T0CKPS в регистър T0CON1. Броячът се инкрементира на всеки преден фронт на тактовия вход (или изхода на входния делител, ако той се използва).

Таймерен режим

В режим Таймер броячът се инкрементира на всеки цикъл на инструкция, докато има валиден тактов сигнал и битове T0CKPS в регистър T0CON1 са нулирани. Когато се използва входният делител, скоростта на инкрементиране на брояча зависи от неговите настройки.

Асинхронен режим

Когато бит T0ASYNC в регистър T0CON1 е установен в 1, броячът се увеличава на всеки преден фронт на тактовия сигнал (или на изхода на входния делител, ако той се използва). Асинхронният режим позволява на брояча да продължи да работи по време на режим Sleep, при условие че тактовият сигнал също е активен.

Синхронен режим

Когато бит T0ASYNC в регистър T0CON1 е нулиран, тактовият сигнал на брояча се синхронизира със системния такт ($F_{OSC}/4$). Когато работи в синхронен режим, тактовата честота на брояча не може да надвишава F_{OSC} / 4.

5.2.3. Източници на тактов сигнал

Източникът на тактов сигнал за брояча чрез битове се задава T0CS<2:0> в регистър T0CON1.

Вътрешен източник на тактов сигнал

Когато е избран вътрешен източник на тактов сигнал, Таймер 0 работи в таймерен режим и се инкрементира в зависимост от тактовия сигнал и настройката на входния делител.

Външен източник на тактов сигнал

Когато е избран външен източник на тактов сигнал, Таймер 0 може да работи в таймерен или броячен режим. Броячът се инкрементира с честота, зависеща от външния източник на тактов сигнал (по преден фронт) и настройката на входния делител.

Програмируем входен делител

Таймер 0 има входен делител, който се конфигурира за 16 опции на коефициент на делене, които са степени на 2: 1: 1 до 1: 32768. Те се задават чрез

битове T0CKPS <3: 0> в регистър T0CON1. Входният делител не може директно да се чете или за записва. Нулиране на регистъра на входния делител може да се извърши чрез запис в регистри TMR0L или T0CON1.

Програмируем изходен делител

Таймер 0 има също и изходен делител с възможност за задаване на 16 опции на коефициент на делене, вариращи от 1: 1 до 1:16. Те се избират чрез битове T0OUTPS <3: 0> в регистъра T0CON0. Средството за сканиране на публикации не може да бъде четено директно или за запис. Нулиране на регистъра на входния делител може да се извърши чрез запис в регистри TMR0L или T0CON0.

5.2.4. Други характеристики на Таймер 0

Работа по време на режим Sleep

В синхронен режим Таймер 0 ще спре да работи. В асинхронен режим той ще продължи да се инкрементира и може да събуди устройството от режим Sleep (ако са разрешени прекъсванията от него), при условие че тактовият сигнал е активен.

Прекъсвания от Таймер 0

Флагът за прекъсване от Таймер 0 (TMR0IF) се установява в 1, когато възникне някое от следните условия:

- Стойността в 8-битовия TMR0L съвпадне с тази, в TMR0H;
- 16-битовият регистър TMR0 се препълни (премине от FFFFh в 0000h).

Когато битовете на изходния делител (T0OUTPS <3: 0>) са настроени за коефициент 1: 1 (няма деление на честотата), флагът T0IF ще се установява в 1 при всяко съвпадение или препълване на TMR0. Най-общо флагът TMR0IF ще се вдига на всяко T0OUTPS +1 съвпадение или препълване.

Ако прекъсванията от Таймер 0 са разрешени (бит TMR0IE в регистър PIE0 = 1), работата на процесора ще бъде прекъсната и устройството може да се събуди от режим на намалена консумация (Sleep).

Изходен сигнал от Таймер 0

Изходът на Таймер 0 може да бъде насочен към всеки I/O извод чрез регистъра за избор на изход RxyPPS (вж. т. 3.2). Той може да се използва и от други периферни устройства, като например за тригер за автоматично преобразуване на аналогово-цифровия преобразувател. Може също и да бъде следен софтуерно чрез бит T0OUT в регистър T0CON0.

TMR0_out ще бъде във високо ниво един тактов период (с отчитане също на коефициента на делене на изходния делител), когато възникне съвпадение между TMR0L и TMR0H в 8-битов режим или когато TMR0 се препълни в 16-битов режим. Изходният сигнал от Таймер 0 е с коефициент на запълване (к.з.) 50%, който се превключва на всеки преден фронт на тактовия сигнал TMR0_out.

5.3. Блокове Таймери 1/3/5

5.3.1. Блокова схема, функционални възможности и принцип на работа

Таймерните блокове 1, 3 и 5 представляват 16-битови таймер/броячи със следните характеристики:

- Два сдвоени регистъра TMR1H:TMR1L, формиращи 16-битов таймер/брояч;
- Програмируем външен и вътрешен източник на тактов сигнал;
- 2-битов входен делител;
- Опционен синхронизиран компараторен изход;
- Множество източници на разрешаващ броенето сигнал;
- Прекъсване при препълване;
- Събуждане при препълване (с външен източник на тактов сигнал, само в асинхронен режим);
- Времева база за функция Буфериране/Сравнение;
- Стартиране на автоматично преобразуване (с използване на CCP);
- Конфигурируема полярност на разрешаващия броенето сигнал;
- Режим на превключване на разрешаващия сигнал за броене;
- Режим на единичен разрешаващ броенето импулс;
- Запазване състоянието на разрешаващия сигнал;
- Прекъсване при броенето в режимите с разрешаващ сигнал.

Блоковата схема на Таймери 1/3/5 е показана на фиг. 5.2. За краткост в описанието на работата по-нататък ще се споменава Таймер 1, като то се отнася и за другите два таймер – 3 и 5.

Блок Таймер 1 съдържа 16-битов сумиращ брояч, до който се осъществява достъп чрез регистровата двойка TMR1H:TMR1L. Записът в TMR1H или TMR1L директно обновява брояча.

Таймерът се разрешава чрез конфигуриране на битовете TMR1ON и GE съответно в регистри T1CON и T1GCON.

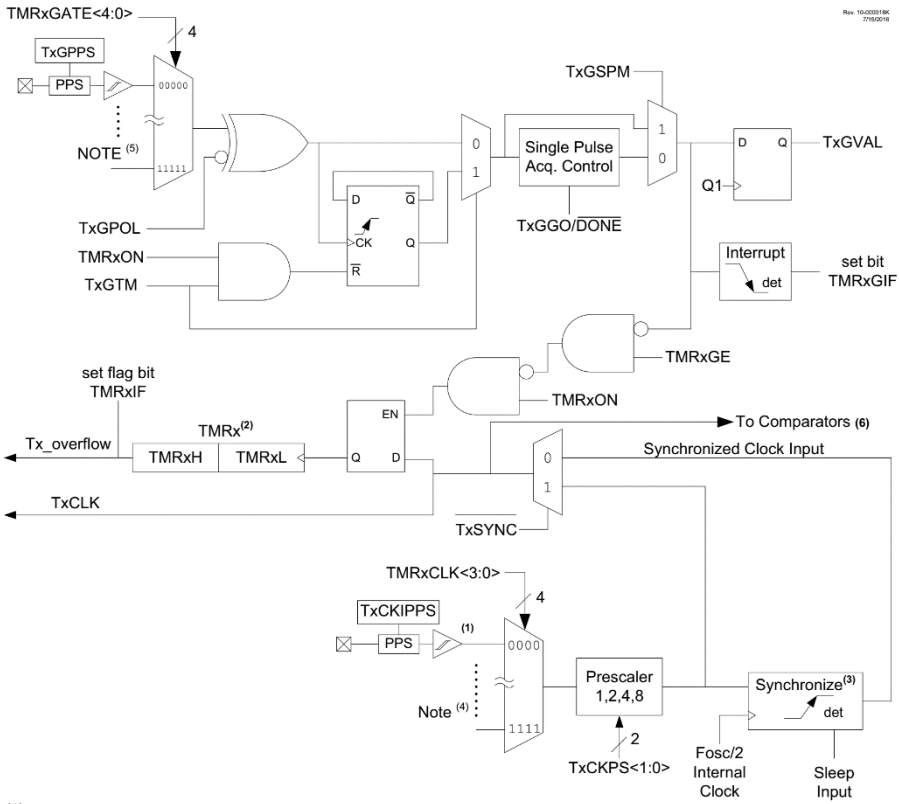
Източници на тактов сигнал

Таймер 1 може да работи с външен и вътрешен източник на тактов сигнал, който се избира от регистър T1CLK.

Когато се използва с вътрешен източник на тактов сигнал, блокът е таймер и се инкрементира на всеки цикъл на инструкция. Когато се използва с външен източник на тактов сигнал, блокът може да се използва или като таймер, или като брояч и се инкрементира на всеки избран фронт на външния източник.

- *Вътрешен източник на тактов сигнал*

Когато е избран вътрешен такт FOSC, двойката регистри TMR1H:TMR1L



(1) При използване на TxCKIPPS ST буферът е високо-скоростен.
(2) Регистър TMRx се инкрементира по преден фронт..
(3) Не работи по време на Sleep.
(4) Възможен е избор от множество източници, избирани чрез регистър TxCLK […].
(5) Към различни изходи, избирани чрез регистър TxGATE […].
(6) Синхронизиран компараторен изход не трябва да се използва заедно със синхронизиран входен такт.

Фиг. 5.2. Блокова схема на Таймери 1/3/5

ще увеличава съдържанието си на кратни фронтове на FOSC, в зависимост от конфигурацията на входния делител. При това то ще нараства с 4 отброявания на всеки цикъл на инструкция. Поради това при четене на стойността в TMR1H:TMR1L ще възникне грешка от 2 LSB (най-младши бита) в раздели-

телната способност. За да се използва пълната разделителна способност на таймера в този режим, трябва да се използва асинхронен входен сигнал към тактовия вход на таймера.

От всички източници на сигнал за таймера, следните могат да бъдат асинхронни и да се използват за тази цел: изходите CLC4, CLC3, CLC2, CLC1, Zero-Cross Detect, Comparator2, Comparator1 и преместваемият чрез блок PPS вход TxG.

- *Външен източник на тактов сигнал*

Когато таймерът е разрешен и е избран съответният вход за външен източник на тактов сигнал (напр. преместваемият T1CKI), таймерът ще се увеличава по преден фронт на сигнала, постъпващ от него. Таймерът може да бъде конфигуриран да работи синхронно или асинхронно.

В режим таймер може да се използва външен кварцов резонатор с честота 32.768 kHz, свързан към изводи SOSCI/SOSCO.

Входният делител на Таймер 1 може да се конфигурира за четири различни коефициента на делене на тактовата честота: на 1, 2, 4 или 8, които се задават чрез битове CKPS в регистър T1CON. Броячът на пред-делителя не може да се чете или записва директно. Той се нулира при запис в TMR1H или TMR1L.

Таймер 1 в режим на 16-битово четене/запис

Таймер 1 може да бъде конфигуриран за 16-битово четене и запис. Когато управляващ бит RD16 (T1CON <1>) е установен в 1, адресът за TMR1H насочва към буферния регистър на старшия байт на таймера. Четенето от TMR1L зарежда съдържанието на старшия байт на Таймер 1 в старшия байт на буферния му регистър. Това предоставя на потребителя възможността да прочете точно всичките 16 бита на брояча, без да се налага да определя дали четенето на старшия байт, последвано от четенето на младшия байт, е невалидно поради препълване на брояча между двете четения, както това се налага при по-стари членове на фамилията PIC16. Записът в старшия байт на брояча също трябва да се извърши през буферния регистър TMR1H. Старшият байт на брояча се актуализира със съдържанието на TMR1H, когато се извърши запис в TMR1L. Това позволява на потребителя да записва всичките 16 бита едновременно в старшите и младши байтове на Таймер 1. Старшият байт на брояча не може да се чете или записва директно в този режим. Всички четения и запис трябва да се извършват през буферния регистър на старшия байт на брояча. Записът в TMR1H не нулира предварителния делител. Това става само само при запис в TMR1L.

Вторичен тактов генератор

Между изводите SOSCI (вход) и SOSCO (изход на усилвателя) има специален ниско-честотен тактов генератор, проектиран да се използва заедно с външен кварцов резонатор с честота 32.768 kHz. Разрешава се чрез установяване в

1 на бит SOSCEN в регистър OSCEN и може да работи в режим Sleep на микроконтролера. Генераторът се нуждае от стартово време за стабилизация на честотата от 1024 такта (вж. т. 28.5 [7]).

Работа на таймера в асинхронен броячен режим

Ако управляващият бит SYNC в регистър T1CON е установен в 1, външният източник на тактов сигнал не е синхронизиран. Таймерът се инкрементира асинхронно спрямо вътрешната тактова поредица. В този режим таймерът ще продължи да работи в режим Sleep и може да генерира прекъсване при препълване, което ще събуди процесора.

Четенето на TMR1H или TMR1L, докато таймерът работи с външен асинхронен тактов сигнал, ще осигури валидно четене (осигурено хардуерно). Потребителят обаче трябва да има предвид, че четенето на 16-битовия таймер като две 8-битови стойности създава определени проблеми, тъй като таймерът може да се препълни между четенията. При запис се препоръчва потребителят просто да спре таймера и да запише желаните стойности. Може да възникне проблем при записване в регистрите на таймера, докато броячът се инкрементира. Това може да доведе до непредсказуема стойност в двойката регистри TMR1H:TMR1L.

5.3.2. Режими на задействане на таймера

Таймер 1 може да бъде конфигуриран да брои свободно или броенето може да бъде разрешавано и забранявано чрез специална схема за задействане на таймера, наречена Timer Gate Enable (TGE). Задействането на таймера може също да се управлява от множество избираеми източници.

5.3.2.1. Режим на разрешаване на таймера

Режимът за разрешаване таймера (Timer Gate Enable - TGE) се избира чрез установяване в 1 на бит GE в регистър T1GCON. Полярността в този режим се управлява от бит GPOL в регистър T1GCON.

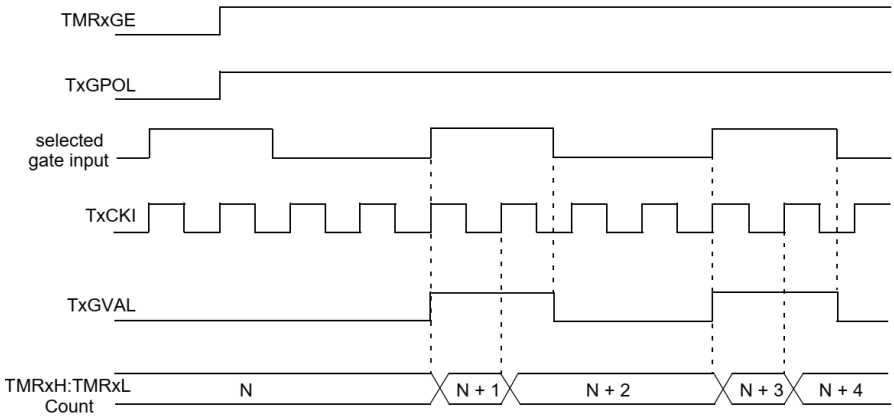
Когато е избран режим TGE, таймерът ще се инкрементира по преден фронт на тактовия сигнал. Когато режимът TGE е забранен, броячът не брои, като се запазва текущата стойност (фиг. 5.3).

Избор на източник на задействащ (разрешаващ) сигнал

За задействане на таймера може да бъде избран един от няколкото различни външни или вътрешни източника на сигнал, който да позволява на таймера да брои. Източникът на входен разрешаващ сигнал може да бъде избран чрез конфигуриране на регистър T1GATE. Полярността за всеки наличен източник също може да се избира, което става чрез бит GPOL в регистър T1GCON.

- *Извод T1G*

Като външен източник на разрешаващ сигнал може да се използва извод T1G.



Фиг. 5.3. Режим TGE на Таймер 1

- *Работа на Таймер 1 при препълване*

Когато Таймер 1 се препълни или възникне съвпадение със стойността в периодния регистър (в 8-битов режим), автоматично се генерира положителен импулс и ще се подаде вътрешно към разрешаващата схема.

- *Изходен сигнал от компаратор 1 или 2 като разрешаващ сигнал*

Изходен сигнал от компаратор 1 или 2 също може да бъде използван като такъв за управление на Таймер 1. Той може да бъде синхронизиран с тактовия сигнал на таймера или да остане асинхронен.

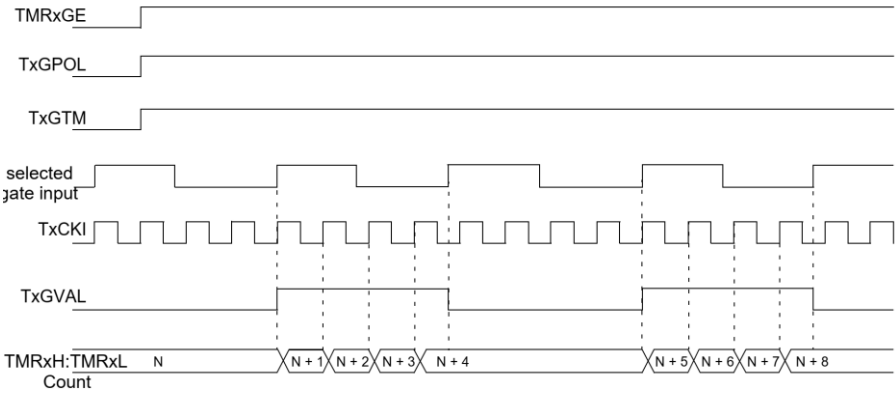
5.3.2.2. Режим на превключване на действието на разрешаващия сигнал за броене

Когато е разрешен този режим (Gate Toggle – GT), е възможно да се измери продължителността на целия период на задействащия сигнал на таймера, за разлика режима на броене само по времето на положителен импулс (фиг. 5.4). Разрешаващият сигнал се насочва през тригер, който променя състоянието на всеки преден фронт на сигнала.

Режим GT се избира чрез установяване в 1 на бит GTM в регистър T1GCON. Когато бит GTM се нулира, тригерът се нулира и остава нулиран. Това е необходимо, за да се управлява по кой фронт се измерва.

5.3.2.3. Режим на броене на Таймер 1 по време на единичен импулс

Когато е избран режим на броене на таймера по време на единичен импулс (Gate Single-Pulse – GSP), е възможно да се буферира събитие във времетраенето на един импулс. Режимът GSP първо се активира чрез установяване на бит GSPM в регистъра T1GCON в 1. След това трябва да бъде установен в 1



Фиг. 5.4. Режим GT на Таймер 1

бит GGO/$\overline{\text{DONE}}$ в регистър T1GCON. Таймерът ще бъде напълно разрешен при следващия преден фронт. При следващия заден фронт на импулса битът GGO/$\overline{\text{DONE}}$ ще се нулира автоматично. Никакви следващи промени на входа няма да мота да инкрементират брояча, докато бит GGO/$\overline{\text{DONE}}$ отново не бъде програмно установен в 1 (фиг. 5.5).

Ако режим GSP се забрани чрез нулиране на бит GSPM в регистър T1GCON, бит GGO/$\overline{\text{DONE}}$ също трябва да се нулира.

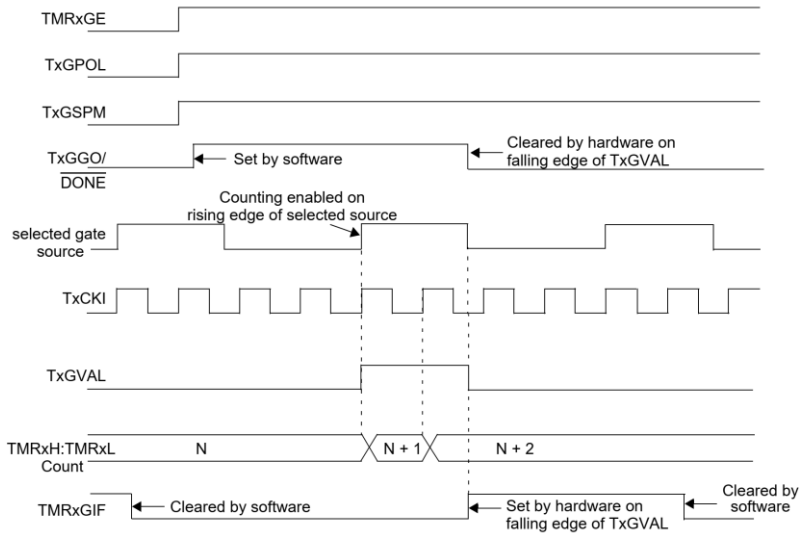
Ако режимите на превключване и на единичен импулс бъдат разрешени едновременно, двете секции ще работят заедно. Това позволява да се измерват времената на цикъла на източника на разрешаващ сигнал за броене на таймера (фиг. 5.6).

5.3.3. Прекъсвания от Таймер 1

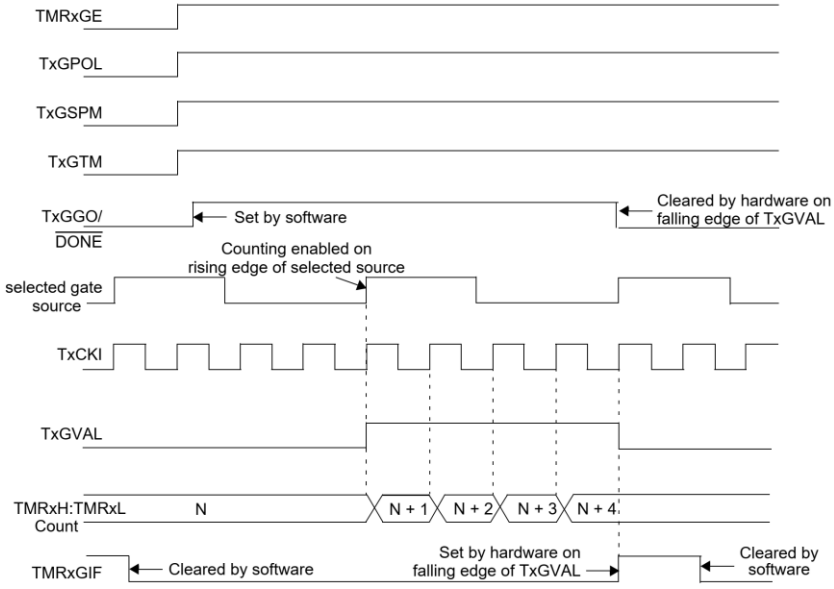
При препълване на брояча на таймера

Двойката регистри на таймера TMR1H:TMR1L се инкрементира до FFFFh и след това преминава към 0000h (броячът се препълва). В този момент се установява в 1 съответният флаг за прекъсване в регистър PIR4. За да се разреши обслужване на прекъсването при препълване, трябва да се установят в 1 следните битове:

- ON в регистър T1CON;
- TMR1IE в регистър PIE4;
- PEIE в регистър INTCON;
- GIE в регистър INTCON.



Фиг. 5.5. Режим GSP на Таймер 1



Фиг. 5.6. Режим GSP на Таймер 1

В подпрограмата за обслужване на прекъсването флагът TMR1IF трябва да се нулира програмно.

При режимите на разрешаване на броенето чрез TGE

Когато е разрешено прекъсването в някой от TGE режимите, е възможно да се генерира прекъсване при възникване на събитие от разрешаващия сигнал. При заден фронт на T1GVAL флагът TMR1GIF в регистър PIR5 се установява в 1. Ако бит TMR1GIE в регистър PIE5 е установен в 1, прекъсването ще се обслужи. Флагът TMR1GIF е активен, дори когато задействащият сигнал не е активен (бит TMR1GE е нулиран).

5.3.4. Работа на Таймер 1 по време на режим Sleep

Таймер 1 може да работи в режим Sleep, само когато е конфигуриран за режим на асинхронен брояч. За да се конфигурира таймерът да може да събужда микроконтролера е необходимо:

- Бит ON в регистър T1CON трябва да е установен в 1;
- Бит TMR1IE в регистър PIE4 трябва да е установен в 1;
- Бит PEIE в регистър INTCON трябва да е установен в 1;
- Бит $\overline{\text{SYNC}}$ в регистър T1CON трябва да е установен в 1;
- Битове CLK в регистър T1CLK трябва да бъдат конфигурирани.
- Тактовият сигнал за таймера трябва да е активен и да продължи да работи по време на режим Sleep. Когато за тази цел се използва SOSC, бит SOSCEN в регистър OSCEN трябва да бъде установен в 1.

МК ще се събуди при препълване на брояча и ще изпълни следващите инструкции. Ако бит GIE в регистър INTCON е установен в 1, ще се изпълни ПОП.

Вторичният осцилатор ще продължи да работи в режим Sleep, независимо от настройката на бит SYNC.

5.3.5. Съвместна работа с блокове CCP

5.3.5.1. Времева база за режими буфериране и сравнение на блокове CCP

CCP модулите използват двойката регистри TMR1H: TMR1L като времева база при работа в режим на буфериране или сравнение.

В режим на буфериране стойността в регистри TMR1H: TMR1L се копира в двойката регистри CCPRxH: CCPRxL при зададено събитие.

В режим Сравнение се задейства събитие, когато стойността двойката регистри CCPRxH:CCPRxL съвпадне със стойността в TMR1H:TMR1L. Това събитие може да стартира автоматично преобразуване.

Таймер 1 не е фиксиран към конкретен CCP1/2/3/4/5 блок. Всички CCP блокове могат да бъдат конфигурирани да споделят един таймер (1, или 3, или 5),

или различни CCP модули могат да бъдат конфигурирани да използват различни ресурси на Таймер 1. Това конфигуриране се извършва чрез регистри CCPTMRS0 и CCPTMRS1.

5.3.5.2. Стартиране на автоматично преобразуване при CCP

Когато някой от CCP блоковете е конфигуриран да задейства автоматично преобразуване, в този момент ще се нулира изчисти двойката регистри TMR1H:TMR1L. Това автоматично преобразуване не причинява прекъсване на таймера. Модулът CCP все още може да бъде конфигуриран да генерира CCP прекъсване.

В този режим на работа двойката регистър CCPRxH:CCPRxL се превръща в периоден регистър за Таймер 1.

За да се използва стартирането на автоматично преобразуване, таймерът трябва да бъде синхронизиран и като източник на тактов сигнал трябва да бъде избран FOSC/4. Асинхронната работа на таймера може да доведе до пропускане на тригера за автоматично преобразуване.

В случай, че запис в TMR1H или TMR1L съвпада с тригер за автоматично преобразуване от CCP, записът ще има предимство.

5.4. Блокове Таймери 2/4/6

5.4.1. Блокова схема и принцип на работа

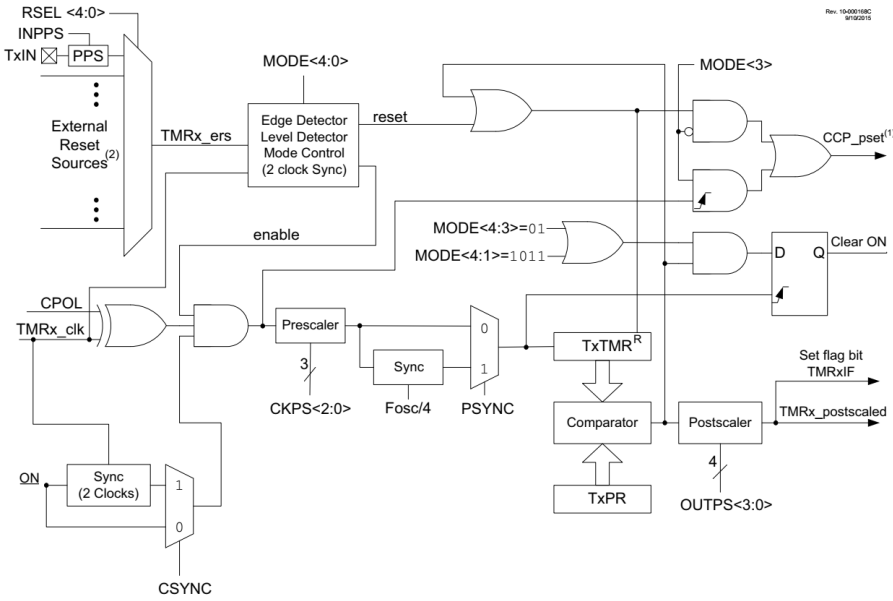
Блоковете Таймер 2/4/6 са 8-битови таймери, които могат да работят като броячи или заедно с външни сигнали, които управляват стартирането, работата, спирането и ресетирането в еднократен цикъл на броене и моностабилен режим на работа. И трите таймера са идентични. Възможно е сложно управление на сигналите, като модулация на импулсна плътност, чрез комбиниране на работата на тези таймери с други вградени периферни устройства, като например компараторите и CCP блоковете.

Основните характеристики на таймерите включват:

- 8-битов регистър на таймера;
- 8-битов периоден регистър;
- Избираеми външни източници за ресет на таймера; Selectable external hardware timer Resets
- Програмируем входен делител (1:1 до 1:128);
- Програмируем изходен делител (1:1 до 1:16);
- Избор на режим на синхронна или асинхронна работа;
- Алтернативни източници на тактов сигнал;
- Прекъсване в края на периода;

- Три режима на работа:
  - Свободно броене;
  - Еднократен цикъл на броене;
  - Моностабилен.

На фиг. 5.7 е показана блоковата схема на Таймерите 2/4/6, като по-нататък за краткост ще се споменава само Таймер 2.



(1) Сигнал към CCP за стартиране на ШИМ импулс.
(2) и (3) По-подробно взаимодействието на блоковете CCP и таймера, както външните източници за НУ, са дадени в [7].

Фиг. 5.7. Блокова схема на Таймери 2/4/6

Във всички режими броячът на Таймер 2 TMR2 брои по преден фронта на тактовия сигнал, постъпващ от програмируемия входен делител. Когато стойността в TMR2 съвпадне с тази в регистър T2PR, изходът се установява във високо ниво, постъпващо към изходния делител. TMR2 се нулира на следващия тактов сигнал.

Възможно е също таймерът да се конфигурира така, че външен сигнал, постъпващ от външно устройство да стартира работата на брояча или да го ресе-

тира. В пусковите режими броячът спира, когато разрешаващият сигнал бъде забранен, и възстановява броенето си, когато бъде разрешен. В ресетиращите режими броячът TMR2 се ресетира по ниво или фронт от външен източник.

Регистърът TMR2 е програмно достъпен за четене и запис. Той се нулира при всяко условие за НУ на МК. Регистър T2PR е двойно-буфериран и се инициализира с 0xFF при всяко НУ. Регистър SFR е програмно достъпен за четене и запис, но само периодният буфер се обновява със стойността в SFR при настъпване на следното условие:

- запис в регистър TMR2;
- запис в регистър T2CON;
- запис в регистър T2HLT;
- при TMR2 = T2PR пълен входен делител;
- събитие за външен ресет, който ресетира таймера.

Броячите на входния и изходния делител се нулират при:

- запис в регистър TMR2;
- запис в регистър T2CON;
- НУ на МК;
- събитие за външен ресет, който ресетира таймера.

5.4.2. Режими на работа

Таймер 2 може да работи в три режима:

- Свободно броене;
- Еднократен цикъл на броене;
- Моностабилен.

Във всеки от тях има различни опции за стартиране, спиране и ресетиране, така че се получават множество подрежими, описани подробно в [7].

Режим на свободно броене

В този режим стойността на TMR2 на всеки тактов цикъл се сравнява с тази в периодния регистър T2PR. Когато двете стойности съвпаднат, на следващия такт компараторът ресетира стойността в TMR2 до 00h и инкрементира брояча на изходния делител. Когато стойността в изходния делител стане равна на тази в битове OUTPS<4:0> в регистър TMRxCON1, на изхода на изходния делител на TMR2 се генерира импулс с продължителност един тактов период и стойността в делителя се нулира.

Еднократен цикъл на броене

Този режим е идентичен на този на свободно броене, с изключение на това, че бит ON се нулира и таймерът спира, когато стойността в регистър TMR2

съвпадне с тази в T2PR. При това той няма да се рестартира, докато бит T2ON не се установи в 0 и след това в 1. Стойности в битове OUTPS<4:0> на изходния делител, различни от 0, са безсмислени в този режим, тъй като таймерът е спрян при първия цикъл на броене и изходният делител се ресетира, когато таймерът се рестартира.

Моностабилен режим

Моностабилните режими са подобни на тези с еднократен цикъл на броене, с изключение на това, че бит ON не се нулира и таймерът може да се рестартира от външно събитие за НУ.

5.4.3. Други характеристики

Изходен сигнал от Таймер 2

Първичният изход от Таймер 2 е TMR2_postscaled (фиг. 5.7), на който се генерират импулси за един период TMR2_clk, когато броячът на изходния делител съвпадне със стойността на битове OUTPS в регистър TMR2CON. Изходният делител T2PR се инкрементира всеки път, когато стойността в TMR2 съвпадне с тази в T2PR. Този сигнал може да се избере като входен за няколко други блока:

- За блок АЦП, като стартиращ автоматичното преобразуване;
- За CWG, като събитие за автоматично изключване;
- За блока за сканиране на паметта, като събитие за стартиране на сканирането;
- За Таймери 1/3/5, като разрешаващ входен сигнал;
- За Таймери 2/4/6, като външен сигнал за НУ;
- За SMT, като входен сигнал за SMT_window и за SMT_signal (фиг. 7.5).

Освен това Таймер 2 може да се използва от блокове CCP при генериране на импулсни поредици в режим ШИМ (вж. тема 6).

Външни източници за начално установяване

За Таймер 2/4/6 могат да се избират външни източници за начално установяване. Това става съответно чрез регистри T2RST, T4RST и T6RST. Тези източници могат да управляват стартирането и спирането на таймера, както и ресетирането му, в зависимост от това, в кой режим работи. Режимът се управлява от битове MODE<4:0> в регистър TMRxHLT. Режимите с управление по фронт изискват шест тактови периода на таймера между външните събития. Режимите с управление по ниво изискват управляващото ниво да е с продължителност поне три тактови периода на таймера. Външните управляващи събития са забранени, по време на режима на настройка (Debug Freeze).

Прекъсвания от Таймер 2

Прекъсване от Таймер 2 се генерира, когато съдържанието на брояча на изходния делител съвпадне с една от 16 стойности за коефициент на делене (от 1:1 до 1:16), които се избират чрез управляващи битове OUTPS<3:0> в регистър T2CON. Прекъсванията са разрешени, когато бит TMR2IE в регистър PIE4 е установен в 1.

Работа на Таймер 2 по време на режим Sleep

Когато бит PSYNC = 1 (изходът на входния делител се синхронизира с Fosc/4), Таймер 2 не може да работи, докато процесорът е в режим Sleep. След него съдържанието на регистри TMR2 и T2PR ще остане непроменено. Когато PSYNC = 0, в режим Sleep таймерът ще работи до тогава, докато има активен тактов сигнал. Това може да е FINTOSC, MFINTOSC или HFINTOSC.

Въпроси за самоконтрол

1. С какво предназначение могат да се използват таймерите?

За всеки от таймерните блокове на микроконтролера PIC16F87X отговорете на средните въпроси:

2. Какво е предназначението на компонентите в блоковата му схема и какъв е принципът му на работа?
3. В какви режими може да работи, с какви източници на тактова честота и може ли тя да се мащабира?
4. Може ли да се стартира и спира работата му?
5. Кога възникват заявки за прекъсване при работата му?
6. Може ли да извежда микроконтролерът от режим с намалена консумация и при какви условия?

6. Блокове за буфериране, сравнение, ШИМ на МК PIC16(L)F18855/75

Блок Буфериране (също и Хронометър, Прихващане)/Сравнение/ШИМ (Capture/ Compare/PWM – CCP) позволява на потребителя да измерва времеви интервали, да управлява различни събития и да генерира сигнали с широчинно-импулсна модулация. Микроконтролерите PIC16(L)F18855/75 имат 5 такива блока, които могат да работят в следните три режима:

- *Буфериране (хронометър, прихващане);*
- *Сравнение;*
- *Генериране на широчинно-импулсно модулирана импулсна поредица (ШИМ).*

В режим Буфериране периферното устройство позволява определяна на продължителността на дадено събитие. Режим Сравнение позволява на потребителя да задейства външно събитие, когато е изтекъл предварително определен период от време. Режим ШИМ може да генерира широчинно-импулсно модулирани сигнали с различна честота и период.

И трите режима са идентични за всички CCP модули.

6.1 Режим „Буфериране“

В режим буфериране, при настъпване на програмирано събитие чрез битове CCPxMODE <3:0> в регистър CCPxCON, 16-битовото съдържание на регистър TMR1 се буферира в регистри CCPR1H:CCPR1L. Това събитие може да бъде някое от следните:

- Всеки заден фронт на сигнала, постъпващ по определен вход;
- Всеки преден фронт;
- Всеки 4-ти преден фронт;
- Всеки 16-ти преден фронт.

Кой точно източник на сигнал ще се използва, се избира чрез конфигуриране на битове CCPxCTS <2:0> в регистър CCPxCAP. Могат да бъдат избрани следните източници:

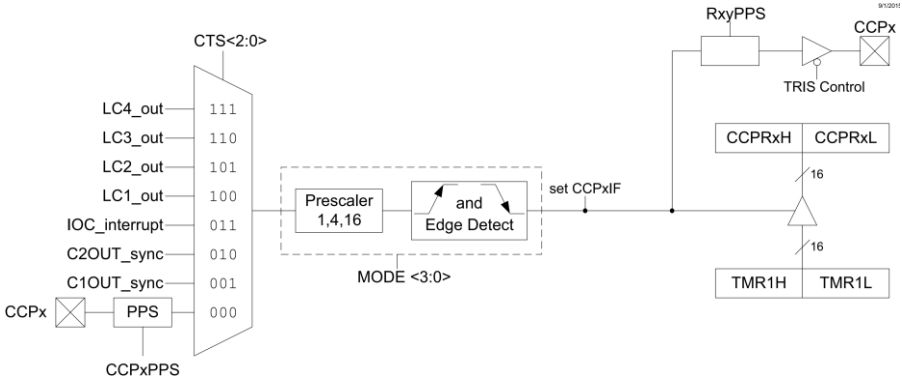
- Вход CCPxPPS;
- C1OUT_sync;
- C2OUT_sync;
- IOC_interrupt;
- LC1_out;
- LC2_out;

- LC3_out;
- LC4_out.

Изводът, по който постъпва входният сигнал за CCPx, трябва да бъде конфигуриран като вход чрез съответния бит в регистър TRIS.

При буферирането се установява в единица флагът за прекъсване CCP1IF в регистър PIR6, като той трябва да бъде нулиран програмно. Ако настъпи ново буфериране, преди стойността в регистър CCPR1 да е прочетена, старата буферирана стойност се губи.

На фиг. 6.1 е показана блоковата схема на CCP блок в режим буфериране.



Фиг. 6.1. Блокова схема на блок CCP в режим буфериране

За да се гарантира правилната работа на CCP модулите в режим „буфериране", Таймер 1, който се използва съвместно с тях, трябва да бъде конфигуриран да работи в *таймерен* или *синхронизиран бройчен режим*.

При промяна на режима е възможно да се генерира фалшива заявка за прекъсване. Ето защо в този момент бит CCP1IE в регистър PIE6 трябва да е нулиран, както и да се нулира флагът CCP1IF в регистър PIR6, ако се е установил в единица при всяка промяна в режима на работа.

Системният такт (F_{OSC}) не трябва да се използва в режим буфериране. За да може в този режим да се разпознае задействащото събитие на извод CCPx, трябва да се използва тактовият сигнал за инструкциите ($F_{OSC}/4$) или от външен източник на тактов сигнал.

Входен делител на CCP

CCP блоковете имат входен делител на честота, за който могат да се задават четири коефициента на делене. Това става чрез битове CCPxMODE <3:0> в регистър CCPxCON. Когато блокът CCP е изключен или не е в режим на бу-

фериране, броячът на входния делител е нулиран. Той се нулира и при НУ на МК.

Превключването от един коефициент на делене към друг не нулира брояча му, като при това може да се генерира фалшиво прекъсване. За да се избегнете това неочаквана операция, преди да се смени стойността в предварителния делител, блокът трябва да се изключи, като се нулира регистър CCPxCON.

Буфериране в режим Sleep

Тъй като, когато Таймер 1 се тактува от $F_{OSC}/4$, няма да работи в режим Sleep, то CCP блокът може да продължи да работи в режим буфериране, само ако Таймер 1 работи с външен източник на тактов сигнал.

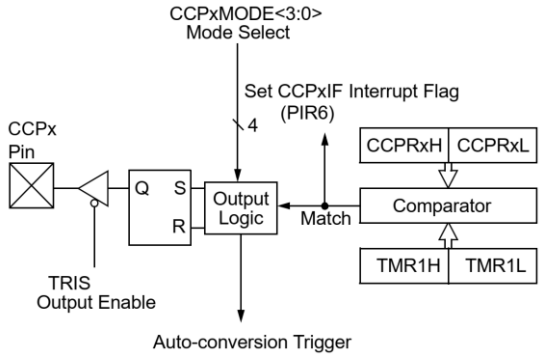
6.2. Режим „Сравнение“

В режим „Сравнение” 16-битовата стойност в регистри CCPRxH:CCPRxL се сравнява непрекъснато със стойността в регистри TMR1H:TMR1L. При възникване на съвпадение, в зависимост от стойността в битове CCPxMODE<3:0> в регистър CCPxCON, става следното:

- Изходът CCPx превключва състоянието си;
- Изходът CCPx преминава във високо ниво;
- Изходът CCPx преминава в високо ниво;
- Генерира се сигнал за автоматично задействане;
- Генерира се програмно прекъсване.

В същото време флагът за заявка да прекъсване CCPxIF се установява в единица и се стартира аналогово-цифровото преобразуване, ако това е зададено.

На фиг. 6.2 е показана блоковата схема на CCP блок в режим сравнение.



Фиг. 6.2. Блокова схема на блок CCP в режим сравнение

Изводът CCPx, който ще се използва, трябва да се конфигурира като изход чрез нулиране на съответния бит в регистър TRIS и да се дефинира чрез регистър RxyPPS.

Изходът на CCP може да се използва също и като вход за други периферни устройства.

Трябва да се има предвид, че нулирането на регистър CCPxCON ще нулира изходния буфер за сравнение на CCPx блока, като това не е буферът за данни на входно-изходния порт.

Както и при режим „Буфериране", режим Таймер 1 трябва да бъде конфигуриран да работи в *таймерен* или *синхронизиран броячен режим*, тъй като в противен случай CCP модулът може да не работи правилно.

Системният такт (F_{OSC}), както в режим „Буфериране", не трябва да се използва в режим сравнение, а тактовият сигнал за инструкциите ($F_{OSC}/4$) или външен източник на тактов сигнал.

Задействащ сигнал за автоматично преобразуване

Всички режими на CCPx установяват флага за прекъсване на CCP (CCPxIF) в 1. Когато този флаг това стане и настъпи съвпадение, може да се осъществи стартиране на автоматично преобразуване, ако модулът CCP е избран като източник за това.

Премахването на условието за съвпадение чрез промяна на съдържанието на регистровата двойка CCPRxH и CCPRxL, между фронта на сигнала, който стартира автоматично преобразуване, и този, който ресетира Таймер 1, ще предотврати НУ.

Сравнение в режим Sleep

Тъй като F_{OSC} е изключен по време на режим Sleep, режимът Сравнение няма да работи правилно, тъй като таймерът не работи. МК ще се събуди при прекъсване (ако е разрешено).

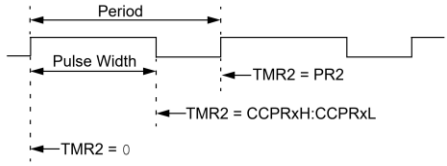
6.3 Режим ШИМ

В режим ШИМ на извод CCPx се генерира широчинно-импулсно модулирана импулсна поредица с до 10-битова разрешаваща способност. Периодът, коефициентът на запълване и разделителната способност се управляват от регистри PR2, T2CON, CCPRxL и CCPxCON.

Времедиаграма на генерирания ШИМ сигнал е показана на фиг. 6.3, а блоковата схема на блок CCP в режим ШИМ на фиг. 6.4.

Честотата на ШИМ сигнала е производна на системната тактова честота. Всякакви промени в системната тактова честота ще доведат до промени в честотата на ШИМ сигнала.

За да се конфигурира извод CCPx като изход, съответният бит TRIS трябва



Фиг. 6.3. Времедиаграма на сигнала, генериран в режим ШИМ

да бъде нулиран.

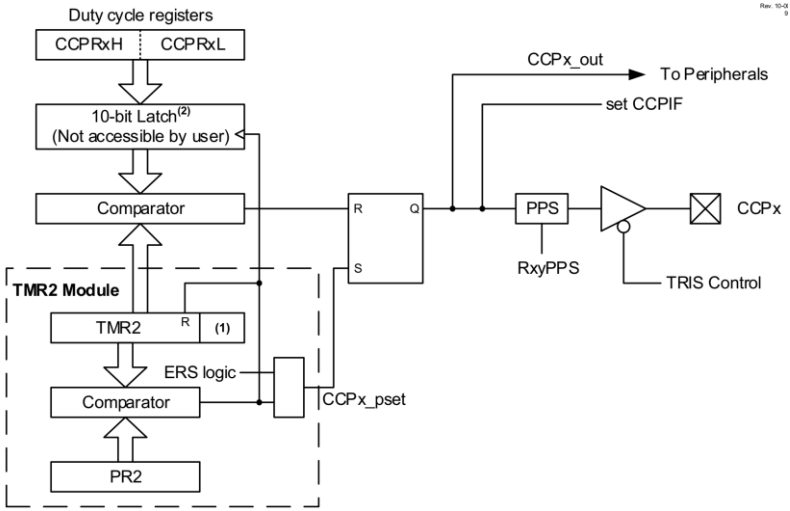
Избор на тактов сигнал за CCP в режим ШИМ

При PIC16F18855/75 е възможно на всеки отделен CCP и ШИМ блок да се избира таймер, който да го управлява. Изборът за всеки блок се прави индивидуално.

Тъй като има до три 8-битови таймера с автоматично презареждане (Таймери 2/4/6), режимът ШИМ на CCP и ШИМ блоковете може да използва всеки от тези таймери.

Регистрите CCPTMRS0 и CCPTMRS1 се използват, за да се избере кой таймер да се използва.

Микроконтролерите PIC16(L)F18855/75 имат по-нова версия на блок TMR2. Той има допълнителни нови режими, които позволяват по-голямо персонализиране и управление на ШИМ сигналите, отколкото при по-старите МК. Работата на CCP изисква таймерът, използван като времева база за ШИМ, да има избран като източник на тактов сигнал $F_{OSC}/4$.



Фиг. 6.4. Блокова схема на блок CCP в режим ШИМ

Период на ШИМ сигнала

Периодът на генерирания ШИМ сигнал се определя чрез запис на константа в регистър PR2/4/6 и може да се изчисли по формулата:

$$T_{ШИМ} = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (к.дел._{TMR2}), \qquad (6.1)$$

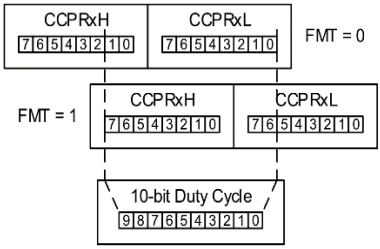
където к.дел.$_{TMR2}$ е коефициентът на делене на предварителния делител. В този режим на работа на CCP изходният делител на Таймер 2/4/6 не се използва.

Когато стойността в TMR2/4/6 се изравни с тази в PR2, на следващия цикъл се случва едно от следните три събития:

- TMR2/4/6 се нулира;
- Извод CCPx се установява в 1. (с изключение на случая, когато к.з. на ШИМ = 0%);
- Стойността за к.з. се прехвърля от регистровата двойка CCPRxL/H в 10-битов буфер.

Коефициент на запълване за ШИМ

Коефициентът на запълване на ШИМ се задава чрез запис на 10-битова константа в двойката регистър CCPRxH:CCPRxL. Подравняването ѝ се определя от бит CCPRxFMT в регистър CCPxCON (фиг. 6.5). В регистровата двойка CCPRxH:CCPRxL може да се записва по всяко време, но стойността за к.з. не се прехвърля в 10-битовия буфер, докато не възникне съвпадение на стойностите в PR2 и TMR2.



Фиг. 6.5.

Продължителността на импулса се изчислява по формула (6.2), а к.з. по формула (6.3):

$$\tau_{ШИМ} = (CCPRxH:CCPRxL) \bullet TOSC \bullet (к.дел._{TMR2}) \qquad (6.2)$$

$$к.з. = \frac{(CCPRxH:CCPRxL)}{4(PR2+1)} \qquad (6.3)$$

Двойката регистри CCPRxH:CPRxL се използва за двойно буфериране на к.з. на ШИМ. Това е от съществено значение за безпроблемната работата в режим ШИМ.

Регистърът на 8-битовия таймер TMR2 се обединява или с 2-битовия вътрешен системен такт (FOSC), или с двата бита на входния делител, за да се

получи 10-битова времева база. Системният такт се използва, ако входният делител на Таймер 2 е конфигуриран за коефициент на делене 1:1.

Когато 10-битовата времева база съвпадне с двойката регистри CCPRxH:CCPRxL, изводът на CCPx се нулира (фиг. 6.4).

Разрешаваща способност в режим ШИМ

Разрешаващата способност определя диапазона от стойности за к.з. за даден период. Например, 10-битова разделителна способност ще доведе до 1024 възможни стойности за к.з., докато 8-битова разделителна способност определя 256 различни възможни стойности. Максималната разделителна способност е десет бита при PR2, инициализиран с константа 255. Тя е функция на стойността в регистър PR2:.

$$R = \frac{\log[4(PR2+1)]}{\log(2)} \tag{6.4}$$

Ако стойността на широчината на импулса е по-голяма от периода, избраният за ШИМ извод ще остане непроменен.

ШИМ в режим Sleep

В режим Sleep регистър TMR2 няма да се инкрементира и състоянието на блока няма да се променя. Състоянието на извод CCPx ще се задържи същото. Когато МК се събуди, TMR2 ще продължи да брои от предишното си състояние.

Какво се случва при НУ

При всяко условие за НУ ще I/O изводи ще се конфигурират като входове, а регистрите на CCP блоковете ще се инициализират с началните си стойности.

Въпроси за самоконтрол

1. В колко различни режима могат да работят модулите за буфериране/ сравнение/ ШИМ?
2. За всеки от режимите отговорете на следните въпроси, като ползвате блоковите схеми:
   - Какъв е механизмът на работа на CCP модула в съответния режим?
   - В какви подрежими може да работи и каква е същността им?
   - Каква е разрешаващата способност на модулите в различните режими, от какво зависи и как се определя?
   - Кога възникват заявки за прекъсване във всеки от режимите и при какви условия се обслужват?
   - Може ли блокът да работи в режим Sleep?

7. Други блокове за генериране, измерване и управление на сигнали

Микроконтролерите PIC16(L)F18855/75 притежават и други блокове за генериране на сигнали с различни параметри, предназначени за различни цели. Ще се спрем накратко на тяхната същност и основни функции, без да навлизаме в детайли на тяхната архитектура, режими на работа, основи изисквания и др., с малки изключения.

7.1. Допълнителни ШИМ блокове

В допълнение на блоковете CCP, микроконтролерите PIC16(L)F18855/75 имат още два блока ШИМ (PWM6 и PWM7). По същество те са същите, като блоковете CCP, но без режимите за буфериране и сравнение.

7.2. Блок Генератор на комплементарни сигнали

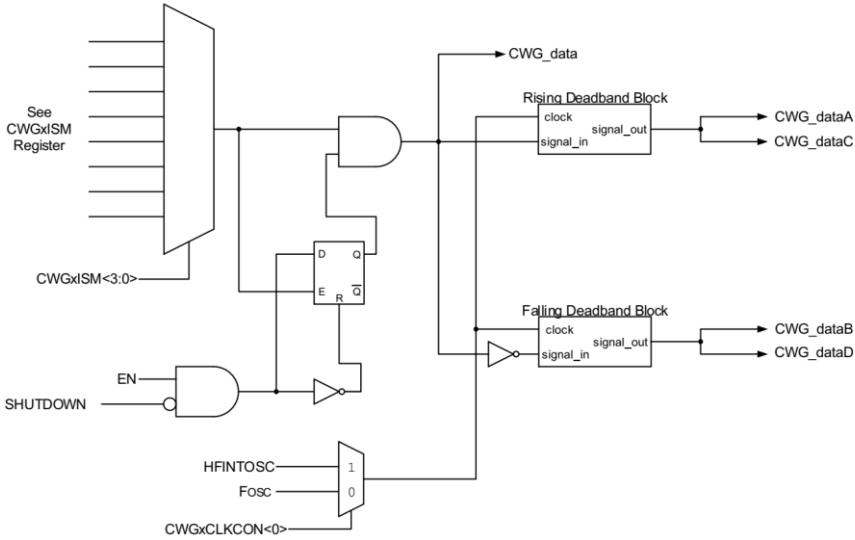
Генераторът на комплементарни сигнали (Complementary Waveform Generator - CWG) генерира правоъгълни импулсни поредици в полумостов, напълно мостов и управляващ режим с ШИМ. Той е съвместим с функциите на ECCP при по-стари фамилии микроконтролери.

Микроконтролерите PIC16(L)F18855/75 имат три CWG блока.

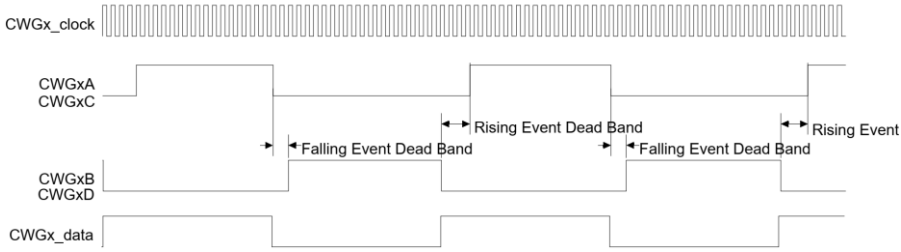
CWG има следните характеристики:

- Шест режима на работа:
  - Режим на синхронно управление;
  - Режим на асинхронно управление;
  - Прав напълно-мостов режим;
  - Обратен напълно-мостов режим;
  - Полумостов режим;
  - Режим Push-Pull.
- Контрол на полярността на изхода;
- Управление на изхода:
  - Синхронизация по преден фронт;
  - Незабавен ефект.
- Независими 6-битови таймери за пауза при преден и заден фронт:
  - Тактуване по време на пауза;
  - Независимо разрешаване на пауза при преден и заден фронт.
- Управление на автоматично изключване с:
  - Избираеми източници за изключване;
  - Разрешаване на автоматичното рестартиране;
  - Управление на извода за автоматично изключване.

В **полумостов режим** (фиг. 7.1) се генерират два изходни сигнала като права и обратна правоъгълна поредица на входа, както е показано на фиг. 7.2. Между двете изходни поредици е осигурена пауза, с цел предотвратяване на свързването на късо при различни приложения за реализация на захранване.



Фиг. 7.1. Блокова схема на блок CWG в полумостов режим на работа



Фиг. 7.2. Времедиаграми при полу-мостов режим на работа на CWG

В режим **Push-Pull** се генерират два изходни сигнала, редуващи се копия на входния сигнал. Това редуване създава push-pull ефект, необходим при управлението на някои типове захранващи източници на базата на трансформатор.

При прав и обратен **напълно-мостов режим** три изхода поддържат постоянно ниво, докато четвъртият се модулира от входния сигнал за данни. В прав напълно-мостов режим CWGxA се привежда в активно състояние, CWGxB и CWGxC се привеждат в неактивно състояние и CWGxD се модулира от входния сигнал. В обратен напълно-мостов режим CWGxC се привежда в активно

състояние, CWGxA и CWGxD се привеждат в неактивните им състояния, а CWGxB се модулира от входния сигнал. В напълно-мостов режим продължителността на паузата се използва, когато има превключвател от прав към обратен режим или обратно.

В **режим на управление** входните данни могат да бъдат насочени към кой да е или всичките четири изхода на блок CWG. В синхронен режим на управление, промени в регистрите за избор на управление оказват ефект на следващия преден фронт на входния сигнал, в асинхронен режим – на следващия цикъл на инструкция.

Генераторът CWG може да генерира изходни сигнали на базата на разнообразие от входни такива, като вход на МК, определен от блок PPS, изходите на CCP блоковете, на компараторите, на NCO генератора и др.

Всеки изход на CWG има индивидуално разрешаване на извеждането, както и избор на полярността на изходния сигнал.

Възможността за генериране на пауза осигурява неприпокриване на генерираните ШИМ сигнали, за да се предотврати късо съединение на ключовите елементи. Използва се в полумостов и напълно-мостов режим. За целта CWG генераторът съдържа два 6-битови брояча. Единият от тях се използва за паузата при нарастващия фронт на входния управляващ сигнал в реверсивен полумостов режим или за пауза при напълно мостов режим. Другият се използва за паузата при падащия фронт на входния управляващ сигнал в прав полумостов режим или за пауза при напълно мостов режим.

Управляваната схема трябва да бъде предпазена от възможността за неизправност или сигнал от обратна връзка, който да пристигне твърде късно или изобщо да не пристигне. В този случай активното управление трябва да бъде прекратено преди състоянието на неизправност да причини повреда. Поради това всички изходни режими поддържат опцията за автоматично изключване.

7.3. Цифрово управляем генератор

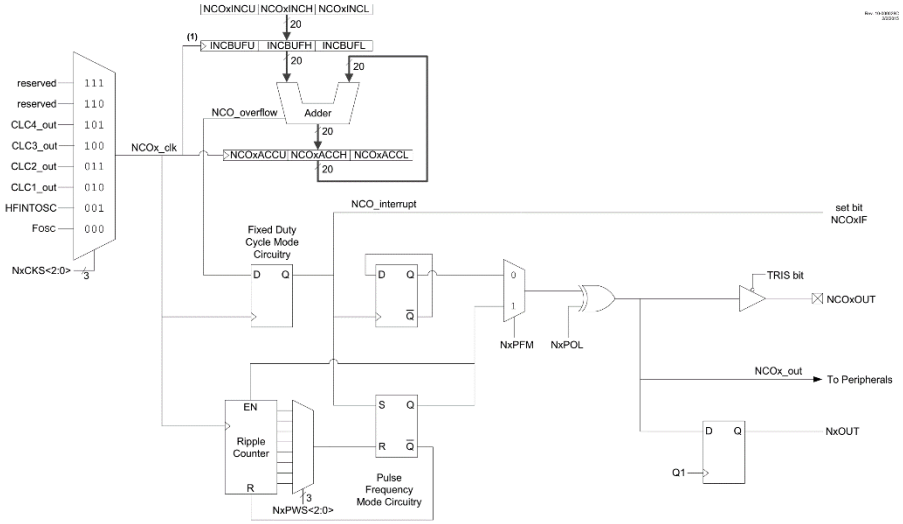
7.3.1. Структура и функционални възможности

Цифрово управляемият генератор (The Numerically Controlled Oscillator - NCO) представлява таймер, който използва препълването при събиране към натрупваща се стойност за разделяне на входната честота. Предимството на метода с прибавяне пред таймер, базиран на обикновен брояч, е, че разделителната способност на изходната честота не се променя в зависимост от стойността на делителя. NCO генераторът е най-полезен за приложение, което изисква точност на честотата и фина разделителна способност при фиксиран коефициент на запълване. Характеристиките на NCO генератора включват:

- 20-битова функция за инкрементиране;

- Режим на фиксиран коефициент на запълване (к.з.) (Fixed Duty Cycle mode - FDC);
- Режим на честотни импулси (Pulse Frequency - PF);
- Управление на широчината на изходния импулс;
- Множество източници на входен тактов сигнал;
- Управление на полярността на изходния сигнал;
- Възможност за прекъсвания.

Опростената блокова диаграма на NCO генератора е показана на фиг. 7.3.



Фиг. 7.3. Блокова схема на NCO генератор

NCO работи чрез многократно прибавяне на фиксирана стойност (N) към съдържанието на акумулатора. Прибавянията се извършват с входната тактова честота (FTГ). Акумулаторът периодично ще се препълва с пренос, който се явява първичната изходна честота от NCO (NCO_overflow). Това ефективно намалява входната тактова честота със съотношението на добавената стойност към максималната стойност на акумулатора (Уравнение 7.1).

$$F_{\text{ПРЕПЪЛВ.}} = \frac{F_{\text{ТГ}}.N}{2^{20}} \qquad (7.1)$$

Изходът на NCO може да бъде допълнително модифициран чрез разширяване на импулса или превключване на тригер. След това модифицираният NCO

изход се разпределя вътрешно към други периферни устройства и по желание може да бъде изведен към извод. Препълването на акумулатора също така генерира прекъсване (NCO_overflow).

Периодът на NCO се променя с дискретни стъпки, за да генерира средна честота. Този изход зависи от способността на приемащата верига (т.е. CWG или външна резонансна преобразувателна верига) да усредни изхода на NCO, за да намали нестабилността.

Източници на тактов сигнал за NCO могат да са: HFINTOSC, FOSC, LC1_out, LC2_out, LC3_out, LC4_out и се избират чрез конфигуриране на битовете N1CKS <2: 0> в регистъра NCO1CLK.

Акумулаторът е 20-битов регистър. Достъпът за четене и запис до него е възможен чрез три регистъра: NCO1ACC, NCO1ACCH и NCO1ACCU.

NCO Adder е пълен суматор, който работи независимо от източника на тактов сигнал. Сумата от предходния резултат и стойността на нарастването замества стойността в акумулатора по нарастващ фронт на всеки тактов сигнал.

Стойността за прибавяне се съхранява в три регистри, съставляващи **20-битов инкрементален регистър**. По ред от най-младши към най-старшия байт те са: NCO1INCL, NCO1INCH и NCO1INCU. Когато NCO модулът е разрешен, първо трябва да се запишат регистрите NCO1INCU и NCO1INCH, а след това регистърът NCO1INCL. Записът в регистър NCO1INCL инициира инкременталните буферни регистри да бъдат заредени едновременно на втория преден фронт на сигнала NCO_clk. Регистрите могат да се четата и записват. Инкременталните регистри са двойно буферирани, за да позволят да се правят промени на стойностите, без първо да се забранява NCO блокът.

Когато NCO модулът е забранен, инкременталните буфери се зареждат веднага след запис в инкременталните регистри. Те не са достъпни за потребителя.

Последното стъпало в NCO блока е това за **полярността на изходния сигнал**. Тя се задава чрез бит N1POL в регистър NCO1CON. Смяната на полярността, докато прекъсванията са разрешени, ще доведе до прекъсване на получения изходен преход.

Изходният сигнал от NCO е достъпен за следните периферни устройства: CLC, CWG, Таймер 2/4/6, SMT, DSM, блок източник на опорен тактов сигнал.

7.3.2. Режими на работа

Режим с фиксиран коефициент на запълване

В режим с фиксиран коефициент на запълване (Fixed Duty Cycle - FDC), всеки път, когато акумулаторът се препълни (NCO_overflow), изходът се превключва. Това осигурява к.з. 50%, при условие че стойността на нарастването остава постоянна.

Режим FDC се избира чрез нулиране на бит N1PFM в регистър NCO1CON.

Импулсно-честотен режим

В импулсно-честотен режим на (Pulse Frequency - PF), всеки път, когато акумулаторът се препълни, изходът става активен за един или повече тактови периоди. След като периодът на тактовия сигнал изтече, изходът се връща в неактивно състояние. Това осигурява импулсен изход. Изходът става активен по преден фронт на тактовия сигнал непосредствено след препълване. Нивото на активните и неактивните състояния зависи от бита за полярност, бит N1POL в регистър NCO1CON.

Режим PF се избира чрез задаване на бита N1PFM в регистъра NCO1CON.

При работа в PF режим, активното състояние на изхода може да варира по продължителност с множество тактови периоди. Различните широчини на импулсите се избират с битове N1PWS <2: 0> в регистър NCO1CLK. Когато избраната широчина на импулса е по-голяма от периода на препълване на акумулатора, тогава DDS операцията е недефинирана.

7.3.3. Други характеристики

Прекъсвания

Когато акумулаторът се препълни (NCO_overflow), флагът за прекъсване от NCO, NCO1IF в регистър PIR7, се установява в 1. За да се разреши прекъсването, трябва да се установят в 1 следните битове:

- бит N1EN в регистър NCO1CON;
- бит NCO1IE в регистър PIE7;
- бит PEIE в регистър INTCON;
- бит GIE в регистър INTCON.

Флагът за прекъсване NCO1IF трябва да се нулира в подпрограмата за обслужването му.

Начално установяване

При НУ всички регистри на блок NCO се нулират.

Работа в режим Sleep

Блокът NCO работи независимо от системния тактов сигнал и ще продължи да работи по време на режим Sleep, при условие че избраният източник на тактов сигнал остава активен.

HFINTOSC остава активен по време на режим Sleep, когато блок NCO е разрешен и като източник на тактов сигнал е избран HFINTOSC, независимо от източника на системен тактов сигнал. С други думи, ако HFINTOSC е избран едновременно като системен такт и тактов сигнал за NCO, когато NCO е разрешен, процесорът няма да работи по време на Sleep, но NCO ще продължи да работи и HFINTOSC ще остане активен. Това ще окаже влияние върху консумацията в режим Sleep.

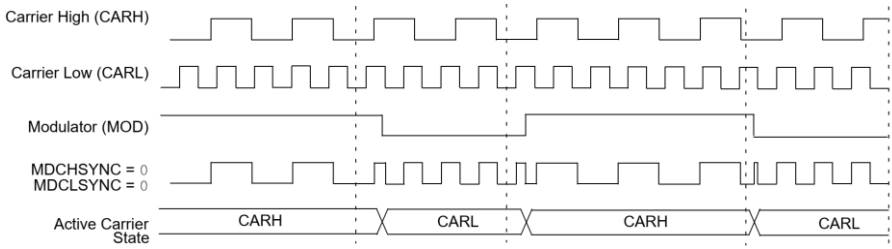
7.4. Блок за модулиране на цифрови сигнали

Блокът за модулиране на цифрови сигнали (сигнали за пренос на данни) (Data Signal Modulator - DSM) е периферен блок, който позволява на потребителя да смесва поток от данни, известен също като модулиращ сигнал, с носещ сигнал, за да генерира модулиран изходен сигнал.

Сигналите на носителя и на модулатора се подават към DSM блока или вътрешно, от изхода на периферно устройство, или външно през входен извод.

Модулираният изходен сигнал се генерира чрез извършване на логическа операция „И“ на носещия и на модулирания сигнали и след това се подава към изхода MDOUT.

Носещият сигнал се състои от два отделни сигнала - за високо ниво (CARH) и за ниско ниво (CARL). През времето, през което модулиращият сигнал (MOD) е с високо логическо ниво, DSM смесва носещия сигнал за високо ниво с модулиращия сигнал. Когато модулиращият сигнал е с ниско логическо ниво, DSM смесва носещия сигнал за ниско ниво с модулиращия сигнал (фиг. 7.4).



Фиг. 7.4. Пример за модулиране на цифрови сигнали без синхронизация на носещите честоти

Използвайки този метод, DSM може да генерира следните типове модулация:

- Честотна манипулация - Frequency-Shift Keying (FSK);
- Фазова манипулация - Phase-Shift Keying (PSK);
- Амплитудна манипулация - Amplitude-shift keying (ASK) или On-Off Keying (OOK);

Освен това DSM блокът има още следните функции:

- Синхронизация на носещата честота;
- Избор на полярност за източника на носещата честота;
- Забрана на извода за източник на носеща честота;
- Програмируеми данни за модулатора;

- Забрана на извода за източник на модулиращата честота;
- Избор на полярност за модулирания изходен сигнал;
- Контрол на стръмността на фронтовете.

Източници за модулиращ и носещ сигнал могат да бъдат изводи на МК (външен източник), изходи от CCP блоковете, от ШИМ блоковете от NCO генератора и др.

7.5. Блок таймер за измерване на параметрите на цифрови сигнали

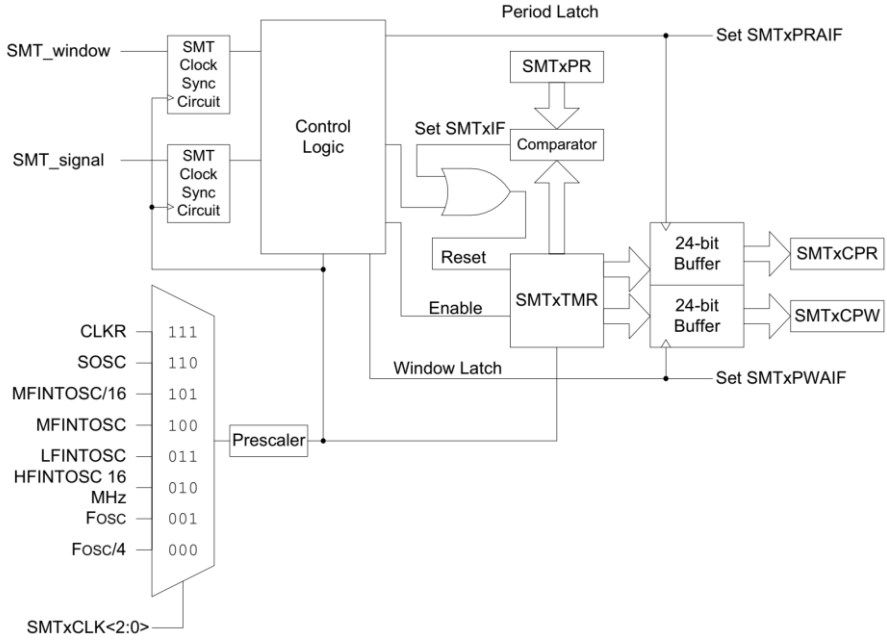
Блокът таймер за измерване на цифрови сигнали (Signal Measurement Timer – SMT) е 24-битов брояч с разширена тактуваща и разрешаваща логика, който може да бъде конфигуриран за измерване на различни параметри на сигналите, като широчина на импулс, честота и коефициент на запълване, както и фазова разлика на два сигнала.

Микроконтролерите PIC16(L)F18855/75 имат два SMT блока, чиято блокова схема е показана на фиг. 7.5.

Основните характеристики на блок SMT включват:

- 24-битов таймер/брояч:
  - Три 8-битови регистъра (SMTxL/H/U);
  - Програмно достъпен за четене и запис;
  - Опционен 16-битов работен режим.
- Два 24-битови буфериращи регистъра за измерване;
- Един 24-битов периоден регистър за съвпадение;
- Различни режими на работа, включително измерване на относителни врермена
- Прекъсване при съвпадение на периода;
- Множество източници на тактови и разрешаващи сигнали и на измервани сигнали;
- Прекъсване при завършване на събирането на данни;
- Способност за четене на текущите входни стойности.

Ядрото на блока е 24-битов брояч – SMTxTMR, съчетан със сложен възел за събиране на данни. В зависимост от избрания режим на работа, SMT може да извършва различни видове измервания, като например таймерен режим (без събиране на данни), таймерен с разрешаване/забрана на броенето, измерване на период и коефициент на запълване, измерване на продължителност на импулс и пауза, измерване на времевата разлика между предните фронтове на сигналите SMTWINx и SMTx и др.; общо 11 различни режима.



Фиг. 7.5. Блокова схема на таймер за измерване на цифрови сигнали

Различни източници на тактов сигнал могат да бъдат избрани чрез битове CSEL<2:0> в регистър SMTxCLK, като F_{OSC}, $F_{OSC}/4$, HFINTOSC (16 MHz), LFINTOSC, MFINTOSC/16 (31.25 kHz), които също могат да бъдат мащабирани.

Подобно на другите таймер, SMT генерира заявка за прекъсване, когато броячът SMTxTMR се препълни (стойността в него премине към 0). Това се случва, когато SMTxTMR = SMTxPR, независимо от режима. Следователно във всеки режим, при който се използва външен сигнал или времева продължителност за ресетиране на таймера, за да работи правилно той, трябва SMTxPR да бъде инициализиран за период, по-голям от този на очаквания сигнал или продължителност.

7.6. Блок с изход за референтна тактова честота

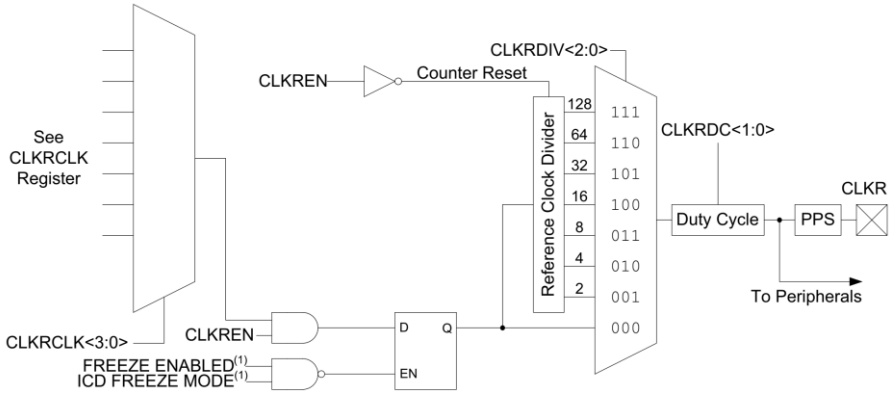
Блокът с изход за референтна тактова честота (Reference Clock Output) дава възможност за извеждане на тактов сигнал към изход CLKR. Той може също да се използва и като сигнал за други периферни устройства, като блок DSM.

Блокът RCO има следните основни характеристики:

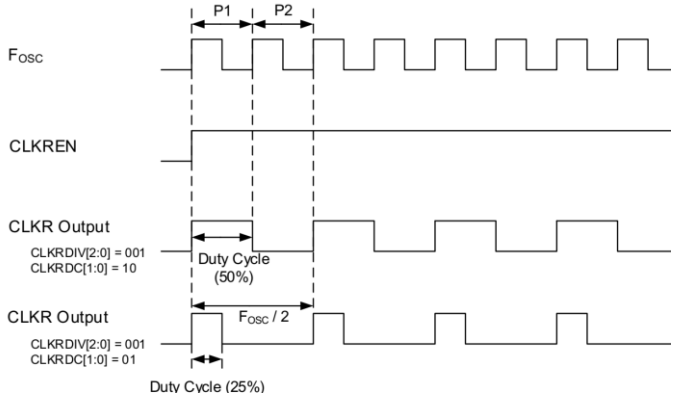
- Избираем входен тактов сигнал;

- Програмируем делител на тактова честота;
- Избираем коефициент на запълване.

На фиг. 7.6 и 7.7 са дадени съответно блоковата схема на блока и времедиаграми на работата му.



Фиг. 7.6. Блокова схема на блока за референтна тактова честота



Фиг. 7.7.

Възможни са различни източници на тактова честота (чрез регистър CLKRCLK), като FOSC, изход от NCO, изходи от конфигурируемите логически клетки и др.

Избраната тактова честота може да бъде разделена, като това се определя от битове CLKRDIV<2:0> в регистър CLKRCON. Различните конфигурации са

следните: основната входна тактова честота; тактовата честота, разделена на: 2, 4, 8, 16, 32, 64, 128.

Коефициентът на запълване също може да се променя, като се задава чрез битове CLKRDC<1:0> в регистър CLKRCON. Могат да бъдат избрани следните варианти: 25%, 50% или 75% от всички стойности на тактови честоти, с изключение на основната FOSC без делене. Подразбиращата се стойност за к.з. при НУ е 50%.

В режим Sleep блокът RCO ще продължи да работи, ако избраният тактов сигнал е активен по време на Sleep.

Въпроси за самоконтрол

За всеки от представените блокове отговорете на следните въпроси:

1. Какво е предназначението на блока?
2. От какви основни блокове се състои как функционира?
3. В какви режими може да работи?
4. Кога (при какви условия) възникват заявки за прекъсване?
5. Може ли да събужда микроконтролера от режим Sleep и при какви условия?

Раздел III. Аналогови блокове в МК PIC16(L)F18855/75

8. Аналогово-цифров преобразувател с допълнителна обработка на резултата

8.1. Структура, принцип на работа и конфигуриране

Блокът Аналогово-цифров преобразувател с изчисление (АЦПИ) (Analog-to-Digital Converter with Computation - ADC2) служи за преобразуване на входен аналогов сигнал в 10-битова двоична стойност. Аналоговите входове, свързани с него, са мултиплексирани, като през избрания аналогов вход се зарежда кондензатор за следене и запомняне. Той е свързан към входа на преобразувателя, който генерира цифров еквивалент на аналоговото ниво по метода на последователните приближения. В резултат се получава 10-битова цифрова стойност, записвана в регистровата двойка ADRESH:ADRESL.

Блоковата схема на АЦПИ е показана на фиг. 8.1.

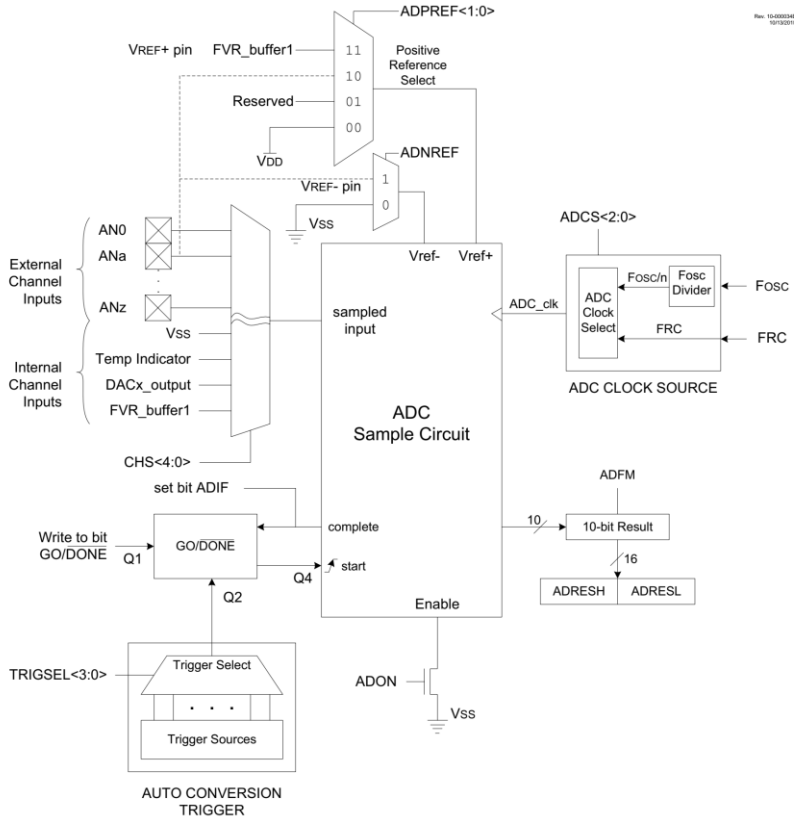
Опорното напрежение за АЦПИ се избира програмно, като може да е от вграден или външен източник. Блокът може да генерира прекъсване при завършване на преобразуването или достигане на зададена гранична стойност на напрежението. Тези прекъсвания могат да събуждат МК от режим SLEEP.

При конфигурирането на АЦПИ може да се задават следните опции и да се избират следните функционални възможности:

- Конфигуриране на портове;
- Избор на канал;
- Избор на опорно напрежение за АЦПИ;
- Избор на източник на тактов сигнал;
- Управление на прекъсванията;
- Форматиране на резултата;
- Избор на източник на събития за стартиране на преобразуването;
- Време за следене;
- Време за предварително зареждане;
- Допълнителен кондензатор за следене-запомняне;
- Преобразуване с единично/двойно стробиране;
- Изходи за предпазен пръстен.

Блокът АЦПИ се разрешава чрез установяване в 1 на бит ADON в регистър ADCON0. Преобразуването се стартира по един от следните начини:

- с установяване в 1 на бит ADGO в регистър ADCON0;
- с външно събитие;



Фиг. 8.1. Блокова схема на блок АЦПИ

- рестартиране при работа в непрекъснат режим.

При завършване на единично преобразуване стойността, която вече е в регистри ADRES, се записва в ADPREV (ако ADPSIS=1), а в ADRES се получава новата получена стойност. Когато преобразуването е завършило, ще се случи следното:

- Бит ADGO се нулира (освен ако бит ADCONT в ADCON0 не е установен в 1);
- Установява се в 1 флагът за прекъсване ADIF;
- Установява се в 1 бит ADMATH;
- Обновява се ADACC.

Ако бит ADDSEN=0, след всяко преобразуване, или ако ADDSEN=1, тогава след всяко второ преобразуване ADERR ще се изчисли и ако получената стойност отговаря на граничните стойности, ADTIF ще се установи в 1. Освен това ще се изчисли ADSTPE и в зависимост от него сравнението с праговата стойност ще доведе до установяване на флага ADTIF в 1.

Изчисляването на филтриращите и прагови стойности става след завършване на преобразуването. Поради това подпрограмата за обслужване на прекъсванията, изпълнявана в резултата на установяването на флага ADIF в 1, трябва първо да провери флага ADTIF, преди да ги прочете.

Ако преобразуването трябва да бъде прекъснато, преди да е завършило, това става с програмно нулиране на бита ADGO. Регистри ADRESH и ADRESL ще се обновят с получената до момента стойност. В този случай се извършва филтриране или сравнение с праговата стойност.

Конфигуриране на I/O портове и избор на аналогов канал

АЦПИ може да се използва за преобразуване както на аналогови, така и на цифрови сигнали. При преобразуване на аналогови сигнали I/O изводи трябва да се конфигурират като аналогови чрез програмиране на свързаните с тях битове TRIS и ANSEL.

Аналоговите нива на входове, конфигурирани като цифрови, не влияят на точността на преобразуване, но могат да доведат до това входният буфер да консумира ток, който е по-голям от допустимия.

Възможен е избор между различни аналогови входни канали: всичките 8 изводи на портове PORTA (RA<7:0>), PORTB (RB<7:0>), PORTC (RC<7:0>), PORTD (RD<7:0>, само при PIC16(L)F18875); трите извода на PORTE (RE<2:0>, само при PIC16(L)F18875); температурният индикатор; изходът от ЦАП; блокът за фиксирано опорно напрежение FVR; аналогова маса AV_{SS}.

Кой аналогов канал ще бъде свързан към схемата за следене-запомняне, се определя чрез регистър ADPCH.

Източници на опорни напрежения

Източник на положително опорно напрежение се избира чрез битове ADPREF в регистър ADREF, като това може да бъде: извод V_{REF+}; V_{DD}; FVR 1.024V; FVR 2.048V; FVR 4.096V.

Чрез бит ADNREF в регистър ADREF се задава източник на отрицателно опорно напрежение, като това може да е извод V_{REF-} или V_{SS}.

Източник на тактов сигнал

Източникът на тактов сигнал за преобразуване се избира програмно чрез регистър ADCLK и бит ADCS в регистър ADCON0. Възможни са два източника на тактов сигнал: $F_{OSC}/(2*(n+1))$ (където n е стойност от 0 до 63) и FRC (специален RC генератор).

Времето за преобразуване на един бит се означава с T_{AD}. Едно пълно 10-битово преобразуване изисква 11.5 T_{AD} периода.

Прекъсвания

При завършване на преобразуването възниква заявка за прекъсване – флагът ADIF в регистър PIR1 се установява в 1, като е необходимо да се нулира програмно. Локалната маска ADIE се намира в регистър PIE1. За да се обслужи то, освен ADIE трябва също и битове PEIE и GIE в регистър INTCON да са установени в 1.

Прекъсването може да събуди МК от режим Sleep, като ако е необходимо процесорът да продължи с изпълнението на прекъснатата програма, без да го обслужва, е необходимо битове ADIE и PEIE да са в 1, но бит GIE да е нулиран. В противен случай, процесорът ще премине към изпълнението на ISR.

АЦПИ ще продължи да работи по време на Sleep, само ако е избран ТГ FRC.

Форматиране на резултата от преобразуването

При завършване на преобразуването резултатът се записва в двойката регистри ADRESH:ADRESL. Тъй като те побират 16-битова стойност, а резултатът от преобразуването е 10-битов, програмистът има възможността да зададе подравняването му вляво (бит ADFRM0 =0) или вдясно (бит ADFRM0=1) в регистровата двойка, като неизползваните битове (съответно най-младшите или най-старшите шест) се четат като нули.

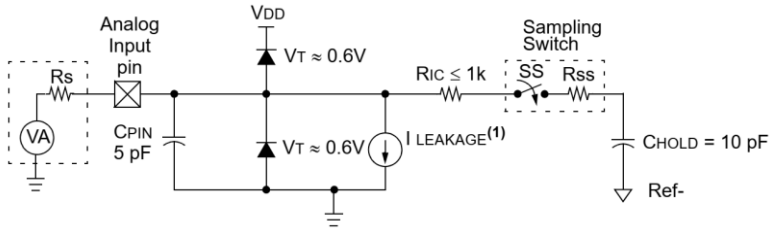
Режим на автоматично преобразуване

Режимът на автоматично преобразуване позволява да се извършват периодични измервания без програмна намеса. Когато бъде открит преден фронт на сигнала от избран източник, бит ADGO ще се установи в 1 по апаратен път. Източник на такова събитие се задава чрез битове ADACT<4:0> в регистър ADACT. Такива могат да бъдат препълванията на таймерите, изходни сигнали от CCP блоковете, компараторите, кофигурируемите логически клетки и др.

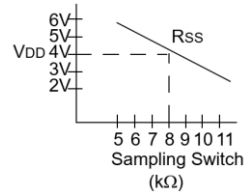
8.2. Осигуряване на необходимото време за следене

За да се постигне определената точност на преобразуване на АЦПИ, е необходимо да се даде възможност на запомнящия кондензатор CHOLD да се зареди напълно до напрежителното ниво на входния канал. Моделът на аналогов вход на АЦПИ е показан на фиг. 8.2.

Импедансът на източника RS и този на вътрешния ключ за следене/запомняне Rss влияят пряко на времето за зареждане на кондензатора CHOLD. Импедансът на ключа за следене/запомняне Rss се променя в зависимост от захранващото напрежение VDD. Препоръчва се **максимален входен импеданс** на аналоговия източник 10 $k\Omega$. С намаляването му намалява и времето за преобразуване.



C_{PIN}	- капацитет на входа
$I_{LEAKAGE}$	- ток на утечка на извода, дължащ се на паразитни връзки
V_T	- прагово напрежение
R_{IC}	- съпротивление на съединенията
SS	- ключ следене/запомняне
R_{SS}	- съпротивление на ключа
C_{HOLD}	- капацитет на кондензатора за следене/запомняне



Фиг. 8.2. Модел на аналогов вход

След като бъде избран (или сменен) аналоговият входен канал, трябва да се изчака да се зареди CHOLD, преди да се стартира преобразуването.

За изчисляване на минималното време за следене може да се използва следното равенство:

$$T_{СЛ} = T_{УУ} + T_{ЗК} + K_T = 2\,\mu s + K_{ЗК} + [(T\text{-}25°C)(0{,}05\,\mu s/°C)], \quad (8.1)$$

където:
$T_{УУ}$ - време за установяване на усилвателя
$T_{ЗК}$ - време за зареждане на кондензатора следене/запомняне
K_T - температурен коефициент
T – температура

Ако приемем, че T=50°C, R_S=10kΩ и VDD=5.0V, то:

$$T_{ЗК} = -C_{С/З}(R_{вр} + R_{С/З} + R_И)\ln(1/2047) =$$
$$= -10\text{ pF}\,(1\,k\Omega + 7\,k\Omega + 10\,k\Omega)\ln(0{,}0004885) =$$
$$= 1{,}37\,\mu s \quad (8.2)$$

$$T_{СЛ} = 2\,\mu s + 892\text{ns} + [(50°C - 25°C)(0{,}05\,\mu s/°C)] =$$
$$= 4{,}62\,\mu s \quad (8.3)$$

Изчисленията са извършени при максималната допустима стойност за грешката от ½ дискрета (1024 стъпки за АЦПИ), определяна от разрешаващата способност на АЦПИ.

Трябва да се имат предвид също и следните особености:

- Опорното напрежение V_{REF} не влияе на изчисленията, тъй като се съкращава;
- C_{HOLD} не се разрежда след всяко преобразуване.

8.3. Работа на АЦПИ по време на режим SLEEP

За да работи АЦПИ, когато микроконтролерът е в режим SLEEP, като източник на тактов сигнал за преобразуване трябва да се използва вграденият генератор FRC. В този случай АЦПИ изчаква един цикъл на инструкция преди стартиране на преобразуването. Това позволява да се изпълни инструкция SLEEP, което ще елиминира шума от цифрово превключване. Ако е разрешено прекъсването от АЦПИ, микроконтролерът ще се „събуди" от режим Sleep. А ако не е, блок АЦПИ ще се изключи, независимо от това, че бит ADON ще остане установен в единица.

Ако възникне външно събитие за стартиране по време на режим Sleep, при избран ТГ FRC, блок АЦПИ ще завърши преобразуването и флагът ADIF ще се установи в 1. Ако е избран друг тип ТГ, събитието ще бъде записано, но преобразуването няма да започне, докато МК не излезе от режим Sleep.

8.4. Капацитивен делител на напрежение

Блок АЦПИ съдържа възел, който способства за измерване на относителен капацитет на всеки канал на АЦПИ с използване на вграден кондензатор за следене/запомняне с опорен капацитет - Capacitive Voltage Divider (CVD). Тази функция може да се използва за реализация на капацитивни тъч бутони и детектиране на близки обекти.

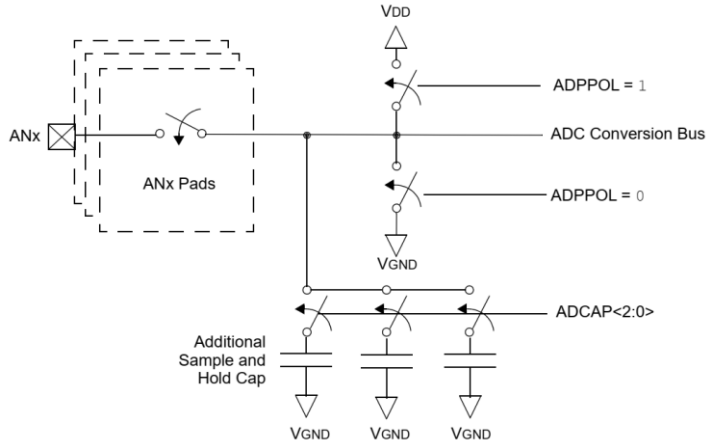
На фиг. 8.3 е показана блоковата схема на CVD възела в АЦПИ.

8.5. Допълнителна обработка на резултата

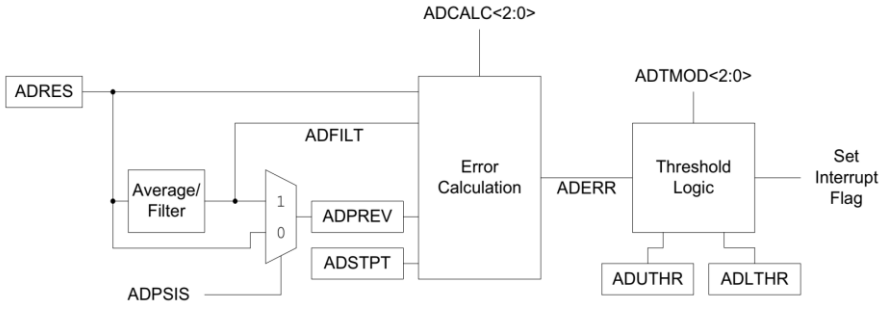
Блок АЦПИ включва също възел за допълнителна обработка на резултата, след завършване на преобразуването, като функции за цифрова филтрация/ средна стойност и сравнение с прагова стойност (фиг. 8.4).

Възелът за допълнителна обработка на резултата се управлява от битове ADMD <2:0> в регистър ADCON2. Той може да работи в един от следните пет режима:

- **Основен**: В този режим блок АЦПИ извършва преобразуване на всяко (ADDSEN=0) или всяко второ (ADDSEN=1) стробиране. Флагът ADIF се установява в след завършване на всяко преобразуване. В този режим са забранени всякакви допълнителни изчислителни обработки.



Фиг. 8.3. Блокова схема на възел за детектиране на относителен капацитет



Фиг. 8.4. Блокова схема на възел за допълнителна обработка на АЦПИ

- **Акумулиращ**: След всяко стартиране на преобразуването резултатът се добавя в акумулатор и броячът ADCNT се инкрементира. Флагът ADIF се установява в 1 след всяко преобразуване. Флагът ADTIF се установява в 1 в зависимост от избрания режим на обработка.

- **Средна стойност**: След всяко преобразуване резултатът се натрупва в акумулатор. Когато се акумулират ADRPT брой стробирания, се изпълнява прагов тест. При стартиране на следващо преобразуване, броячът се ресетира до 1. За следващи прагови тестове се правят нови ADRPT на брой стробирания за натрупване на стойност в акумулатора.

- **Ускорена средна стойност**: При събитие за старт на преобразуването акумулаторът и броячът се нулират. Резултатите от преобразуванията след

това се отчитат последователно, до акумулирането на ADRPT стробирания и накрая се тества за достигане на прагова стойност.

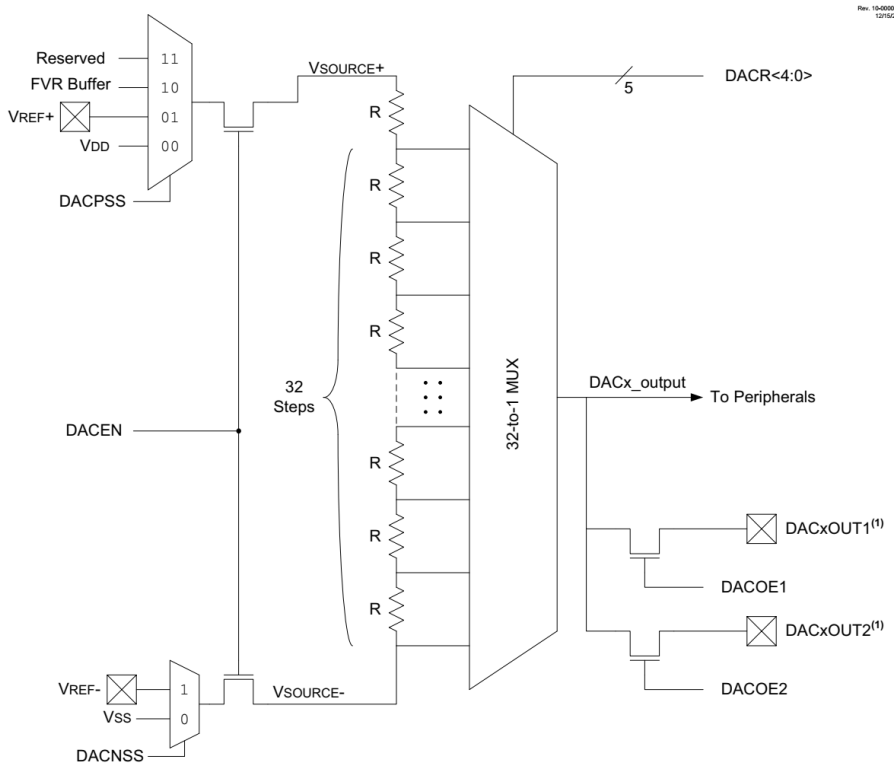
- **Ниско-честотно филтриране**: След всяко преобразуване резултатът се изпраща през филтъра. Когато се извършат ADRPT стробирания, се изпълнява прагов тест.

Въпроси за самоконтрол

1. Какъв е заложеният метод на преобразуване в блок АЦПИ?
2. Колко канален е и кои входно-изходни портове могат да се използват за работа с него?
3. От какво зависи времето за преобразуване? А разрешаващата способност?
4. С какви източници на опорно напрежение може да работи блок АЦПИ?
5. Може ли да работи в режим SLEEP? От какво зависи?
6. Какво е предназначението на капацитивния делител на напрежение?
7. За какво е предназначен възелът за допълнителна обработка и в какви режими може да работи?

9. Цифрово-аналогов преобразувател

Цифрово-аналоговият преобразувател (ЦАП) генерира аналогов сигнал, пропорционален на входен източник с 32 избираеми нива на изхода (фиг. 9.1).



(1) За извод(и) DACxOUT е предвиден сигнал DACx_output

Фиг. 9.1. Блокова схема на блок ЦАП

Входът на ЦАП може да бъде свързан към:

- Външни изводи V_{REF};
- Захранващото напрежение V_{DD};
- От блока за фиксирано опорно напрежение FVR.

Изходният сигнал от ЦАП може да се конфигурира да осигурява опорно напрежение за:

- Положителния вход на компаратора;
- Входен канал за блок АЦП;
- Извод DAC1OUT.

Блок ЦАП може да се разрешава/забранява чрез ит DAC1EN в регистър DAC1CON0.

Нивата на изходното напрежение (32 на брой) се задават чрез битове DAC1R<4:0> в регистър DAC1CON1.

Изходното напрежение от ЦАП се определя по формулата:

$$V_{OUT} = \left((V_{SOURCE+}) - (V_{SOURCE-}).\frac{DAC1R<4:0>}{2^5}\right) + (V_{SOURCE-}) \quad (9.1)$$

$V_{SOURCE+} = V_{DD}$ или V_{REF+} или FVR

$V_{SOURCE-} = V_{SS}$ или V_{REF-}

Изходните нива от ЦАП се получават чрез резисторна матрица, чийто изводи са свързани към източници на положително и отрицателно опорно напрежение Ако напрежението на някой от източниците не е стабилно, то ще окаже съответно влияние на изходния сигнал.

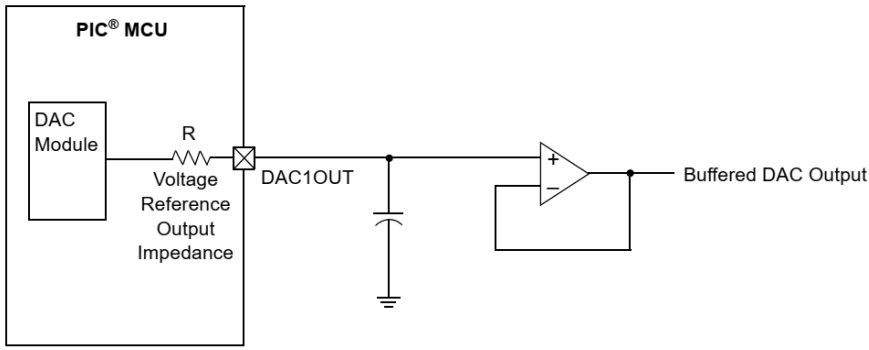
Напрежението от ЦАП може да се изведе към изводи DAC1OUT1/2, като това се определя от битове DAC1OE1/2 в регистър DAC1CON0. Изборът на опорно напрежение за изходи DAC1OUT1/2 автоматично презаписва стойността в буфера за цифров изход, като детекторът за праг на цифровия вход сработва и деактивира изтеглящите към захранване резистори. Четенето на извод DAC1OUT 1/2, когато той е конфигуриран като изход за опорно напрежение, винаги връща като резултат 0.

Поради ограничената възможност за управление на тока, на изхода за опорно напрежение на ЦАП трябва да се използва буфер за външните връзки към изводи DAC1OUT1/2 (фиг. 9.2).

Блокът ЦАП продължава да работи по време на режим Sleep. Когато МК излезе от режима чрез прекъсване или таймаут на следящия таймер, съдържанието на регистър DAC1CON0 остава непроменено.

При начално установяване се случва следното:

- ЦАП е спрян;
- Изходното напрежение на ЦАП към изводи DAC1OUT1/2 се прекъсва;
- Битовете за избор на диапазон DAC1R<4: 0> се нулират.



Фиг. 9.2. Пример за изходен буфер на изход за опорно напрежение DAC1OUT

Въпроси за самоконтрол

1. Опишете работата на блок ЦАП по блоковата му схема.
2. Кои могат да са източниците на опорни напрежения?
3. На къде може да бъде насочен изходният сигнал от АЦП?

10. Блок източник на фиксирано опорно напрежение

Блокът за фиксирано опорно напрежение (Fixed Voltage Reference – FVR осигурява стабилно опорно напрежение, независимо от захранващото VDD, с избираеми изходни нива 1.024V, 2.048V или 4.096V. Изходът от FVR може да се конфигурира да осигурява опорно напрежение за следните блокове:

- Входен канал за АЦП;
- Положително опорно напрежение за АЦП;
- Положителния вход на компаратор;
- Цифрово-аналоговия преобразувател.

FVR се разрешава чрез установяване в 1 на бит FVREN в регистър FVRCON.

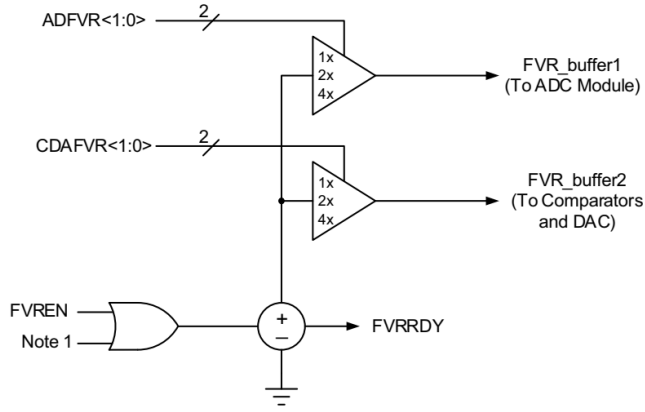
Изходното напрежение от FVR не може да надвишава VDD.

Изходът от FVR, който е свързан към АЦП, компараторите и ЦАП, се рутира през два независими програмируеми усилвателя. Всеки от тях може да бъде програмиран за усилване по 1, 2 или 4 за получаване на трите възможни нива на напрежение.

За разрешаване и настройка на усилвателите при работа с блок АЦП се използват битове ADFVR<1:0> в регистър FVRCON. За съвместна работа с ЦАП и компараторите се използват битове CDAFVR<1:0> в регистър FVRCON.

Когато блок FVR е разрешен, е необходимо време за стабилизиране на схемата, след което ще се установи в 1 бит FVRRDY в регистър FVRCON.

Блоковата схема на блок FVR е показана на фиг. 10.1.



Фиг. 10.1. Блокова схема на блок FVR

Въпроси за самоконтрол

1. Какво е предназначението на блок FVR и какъв е принципът му на работа?
2. За кои други периферни блокове на МК може да осигурява опорно напрежение?

11. Компаратори

Компараторите се използват като интерфейс между аналогови и цифрови схеми и блокове. Те работят, като сравняват две аналогови нива на напрежение и генерират цифров сигнал, индициращ техните относителни величини. Компараторите са много полезни възли в градивни блокове на устройства и системи със смесени сигнали, тъй като оигуряват обработка на аналогови сигнали, независимо от програмното изпълнение.

МК PIC16(L)F18855/75 имат два компаратора със следните основни характеристики:

- Програмируем избор на входни сигнали;
- Програмируема полярност на изходния сигнал;
- Прекъсвания по преден/ заден фронт на изходния сигнал;
- Събуждане на МК от режим Sleep;
- Програмируема оптимизация за бързодействие/ консумация;
- Източник за автоматично изключване за блок CWG1;
- Избираемо опорно напрежение

На фиг. 11.1 е показана функционалната схема на компаратор с взаимовръзката между входните аналогови нива и цифровия изход. Когато аналоговото напрежение на входа V_{IN+} е по-малко, от това на входа V_{IN-}, на изхода на компаратора ще се установи ниско логическо ниво. Когато аналоговото напрежение на входа V_{IN+} е по-високо, от това на входа V_{IN-}, на изхода на компаратора ще се установи високо логическо ниво.

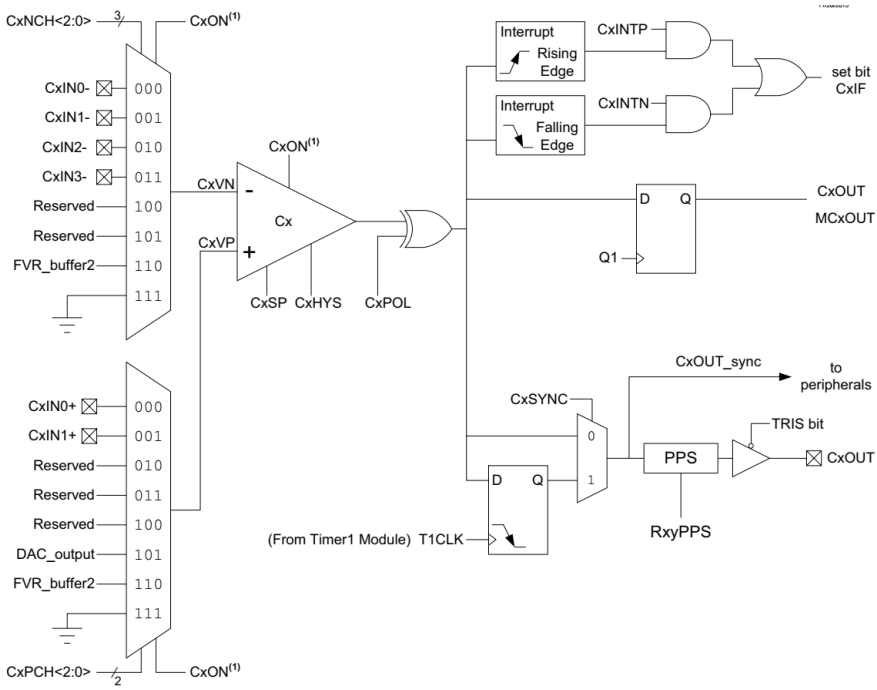
Конфигурирането и управлението на компараторите се осъществява чрез два управляващи регистъра, с каквито разполага всеки от тях: CMxCON0 и CMxCON1.

Регистър CMxCON0 съдържа битове за управление и състояние със следното предназначение:

- *За разрешаване* – чрез установяване в1 на бит CxON в регистър CMxCON0. Ако той е нулиран, работата на компаратора се забранява, което води до намаляване на консумацията.
- *За изходния сигнал*

Изходът на компаратора може да се наблюдава, или чрез четене на бит CxOUT в регистър CMxCON0, или чрез бит MCxOUT в регистър CMOUT.

Изходът от компаратора може също да бъде трасиран до изход на МК посредством регистър RxyPPS. Съответният TRIS бит трябва да бъде нулиран, за да се конфигурира изводът като изход.



(1) При CxON = 0 всички входове на мултиплексора са несвързани и на изхода ще има „0“.

Фиг. 11.1. Функционална схема на компараторен блок

- *За полярността на изходния сигнал*

Инвертирането на изходния сигнал на компаратора е равносилно на размяната на входните сигнали. Задава се чрез установяване в 1 на бит CxPOL в регистър CMxCON0. При CxPOL=0 имаме неинвертиран изход.

- *За разрешаване на хистерезис*

Възможно е да се избере функция за хистерезис при работата на всеки компаратор, с добавяне на известна стойност на разликата в напреженията (типична стойност 25 mV) при двата входа. Разрешава се чрез установяване в 1 на бит CxHYS в регистър CMxCON0.

- *За синхронизация на Таймер 1*

Изходен сигнал от работещ компаратор може да се използва като разрешаващ сигнал за Таймер 1. Тази възможност е полезна за задаване на времеви параметри за аналогови събития.

Препоръчва се компараторният изход да се синхронизира с Таймер 1. Това ще гарантира, че Таймер 1 няма да се инкрементира, докато се извършва промяна в компаратора.

Регистър CMxCON1 съдържа управляващи битове-маски за разрешаване на прекъсвания по преден/ заден фронт.

Прекъсвания от компараторите

Прекъсване може да се генерира при промяна на състоянието на изхода на компаратора (за всеки компаратор), като за целта в структурата на компараторите са включени детектори за преден и заден фронт.

Когато някой детектор сработи и свързаният с него разрешаващ бит е установен в 1 (CxINTP и/ или CxINTN в регистър CMxCON1), съответният флаг за прекъсване (CxIF в регистър PIR2) ще се установи в 1, като трябва програмно да се нулира. За да се обслужи прекъсването, трябва да се установят в 1 още следните битове: CxON, CxPOL и CxSP в регистър CMxCON0; CxIE в регистър PIE2; CxINTP в регистър CMxCON1 (за прекъсване по преден фронт); CxINTN в регистър CMxCON1 (за прекъсване по заден фронт); PEIE и GIE в регистър INTCON.

Избор на входен сигнал към неинвертиращия вход на компаратора

Инициализацията на битове CxPCH<2:0> в регистър CMxCON1 задава източник на сигнал към неинвертиращия вход на компаратора – вътрешно опорно прежение или аналогов вход: аналогов вход CxIN0+; изход от ЦАП; от блока за фиксирано опорно напрежение FVR; V_{SS} (маса).

Избор на входен сигнал към инвертиращия вход на компаратора

Чрез битове CxNCH<2:0> в регистър CMxCON1 се задава източник на сигнал към инвертиращия вход на компаратора - вътрешно опорно напрежение FVR, аналогов вход CxIN или аналогова маса.

Някои опции за избор на сигнал за инвертиращия вход на компаратора споделят извод с изхода на операционния усилвател. Едновременното разрешаване на двете функции ще насочи изходния сигнал от ОУ към инвертиращия вход на компаратора.

За конфигуриране на изводи CxINy+ и CxINy- като аналогови входове, е необходимо да се установят в 1 съответните битове в регистър ANSEL, както и TRIS битовете, за да се забранят изходните драйвери.

Източник за автоматично изключване за блок CWG1

Изходен сигнал от компараторния блок може да се използва за автоматично изключване за блок CWG1. Когато изходът на компаратора е активен и съответният ASxE е разрешен, работата на блок CWG ще бъде незабавно прекратена.

Работа в режим Sleep

Компараторният блок може да работи в режим Sleep. Тактовият му сигнал се базира на тактовия сигнал на Таймер 1. Ако тактовият сигнал на Таймер 1 е системният такт (F_{OSC}) или цикълът на инструкциите ($F_{OSC}/4$), Таймер 1 няма да работи по време на Sleep, и синхронните изходи на компараторите няма да функционират. Прекъсване от компаратор може да събуди МК от режим Sleep. За да се разрешат прекъсванията, битове CxIE в регистър PIE2 трябва да са установени в 1.

Въпроси за самоконтрол

1. Какво е предназначението на компараторните блокове?
2. Кои периферни блокове могат да са източници на входни сигнали за тях?
3. Кои други техни характеристики са програмируеми?
4. При какви условия се генерират заявки за прекъсване?
5. Могат ли да работят в режим Sleep и да „събуждат“ МК и при какви условия?

Раздел IV. Серийни интерфейси на МК PIC16(L)F18855/75

12. Главен синхронен сериен порт (MSSP)

12.1. Същност и видове последователен обмен на данни при PIC МК. Предназначение на блок MSSP

Съществуват различни методи за обмен на информация според разрядността на едновременно предаваните данни, но при PIC микроконтролерите се използват следните:

- ***Паралелен***, при който се предават едновременно по отделни линии всички битове на данната;
- ***Последователен (сериен),*** при който отделните битове се предават по една линия бит по бит;

От тяхната същност следват и техните основни **предимства и недостатъци**. При паралелния обмен на данни са необходими повече линии за връзка, но се постига по-висока скорост на обмен. За реализация на последователен обмен са необходими по-малко проводници, но той е по-бавен и са необходими по-сложни апаратни средства за приемане и предаване на данните. Ето защо в миналото за обмен на къси разстояния е използван предимно паралелен интерфейс. В днешно време, обаче предимствата на паралелния обмен са все повече под въпрос – предимството на последователните интерфейси за значително помалък брой свързващи линии в много случаи е решаващо. Ето защо, при широкото използване на последователния обмен на данни, усилията на разработчиците са насочени към преодоляване на посочените недостатъци.

При обмена на данни бит по бит възникват въпросите - как да се определи кога започва и завършва даден бит, а също и началото и края на предаваната данна?

За идентификация на отделните битове се използват също два от съществуващите няколко метода:

- ***Синхронен последователен обмен*** – тактовият сигнал „съпровожда" данните по отделна линия;
- ***Асинхронен последователен обмен*** – не се предава тактов сигнал, функциите за синхронизация са „внедрени" в самите данни.

За разпознаване на началото и края на предаваната данна тя обикновено се „опакова" в някакъв формат. Пример за такъв формат, използван при асинхронен обмен е показан на фиг. 12.1. При него началото на обмена на n-битовата данна се маркира от стартов бит с ниско ниво (различно от високото неактивно ниво на линията). След информационните битове маже да има контрол по четност /нечетност (P) и се завършва с поне един стопов бит с ниско ниво.

Фиг. 12.1. Примерен формат на съобщение при асинхронен последователен обмен

Синхронизацията и форматирането на данните предполагат използването на правила, които да гарантират съгласуването на връзката. Тези набори от правила се наричат протоколи, някои от които са доста прости, а други значително по-сложни.

Обменът на данни рядко е еднопосочен. Да си представим две устройства, свързани с последователен интерфейс. Ако те искат да предадат информация едно на друго по една единствена линия за обмен на данни, то очевидно това не може да се случи едновременно. Налага се те да си „разделят" използването й във времето, т.е. в определен интервал от време едното да предава, а другото само да приема, а в следващ интервал от време да става обратното. Този режим на работа се нарича **полудуплексен последователен обмен** на данни. Съществува и друг тип реализация на последователен интерфейс, при която са предвидени две линии за връзка – тогава предаването и приемането на данни от страна на двете устройства може да се извършва едновременно – **напълно дуплексен обмен** (пълен дуплекс).

При разглежданите микроконтролери има два периферни блока, които реализират последователен обмен на данни и които са предмет на настоящата и следващата теми – блок MSSP и блок EUSART, а при някои микроконтролери с PIC ядро, също и блокове USB и CAN.

Главният синхронен сериен порт (Host Synchronous Serial Port - MSSP, M от Master при по-старите PIC микроконтролери) се използва за последователен синхронен обмен на данни с други микроконтролери или периферни устройства. Такива са например серийни EEPROM памети, преместващи регистри, драйвери за дисплеи, АЦП и др. Модулът MSSP може да работи в два режима:

- Сериен периферен интерфейс - Serial Peripheral Interface (SPI);
- Интерфейс I2C (Inter-Integrated Circuit).

12.2. Режим SPI

12.2.1. Същност, блокова схема, принцип на работа

През 70-те и началото на 80-те години на миналия век фирми като National Semiconductor, Motorola и др. разработват микропроцесори и микроконтролери, които имат вградени функции за синхронен последователен обмен на данни. При това възниква необходимостта от дефиниране на техните работни характеристики, което да помогне на други фирми при разработката и производството на устройства, които да могат надеждно да се свързват с тях. На базата

на техните разработки са съставени два стандарта – “Microwire” на National Semiconductor и SPI – Motorola. Двата стандарта са подобни и съвместими и се поддържат от много съвременни устройства. Блоковете SSP и MSSP на микроконтролерите PIC16 поддържат стандарта SPI.

Шината SPI е предназначена за сихронен сериен обмен на данни в напълно дуплексен режим. Свързаните устройства могат да са хостове (главни) или клиенти (подчинени). В една система с шина SPI може да има само един хост. Началото на обмена се инициира от хост-устройството. Устройствата-клиенти се адресират чрез сигнал Chip Select, известен също и като Client Select.

За комуникация се използват следните **три извода:**

- За извеждане на серийни данни - Serial Data Out (SDO);
- За въвеждане на серийни данни - Serial Data In (SDI);
- За тактов сигнал (SCK).

Още един, **четвърти извод** може да се използва при подчинен режим при повече от едно устройство-клиент:

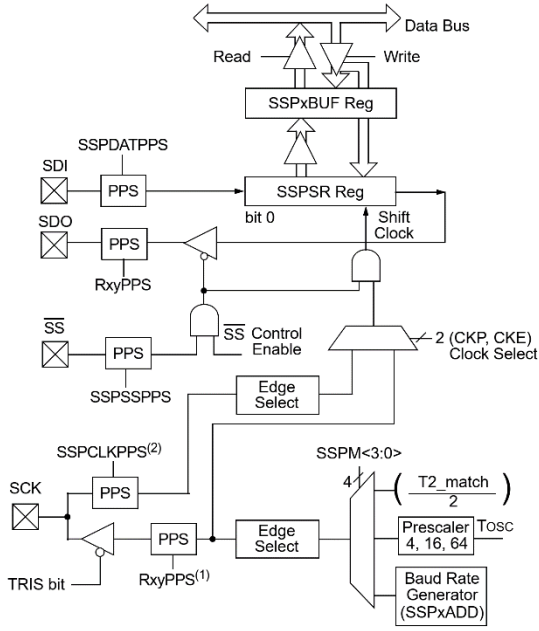
- Избор на подчинен модул - Client Select ($\overline{CS}$).

Блокова схема и принцип на работа

На фиг. 12.2 е показана структурната схема на блок MSSP, когато той работи в режим SPI.

Блок MSSP съдържа преместващ регистър за приемане/ предаване (SSPxSR) и буферен регистър (SSPxBUF). SSPxSR измества данните навън и навътре в устройството, започвайки с най-старшия бит. Регистър SSPxBUF съхранява данните, записани в SSPxSR, докато не са готови новите приети данни. След като се приемат всичките осем бита, целият байт се прехвърля в регистър SSPxBUF. След това битът за пълен буфер (Buffer Full Detect) в регистър SSPxSTAT и флагът за прекъсване SSPxIF се установяват в 1. Това двойно буфериране на приеманите данни (SSPxBUF) позволява да се започне приемане на следващия байт, преди четенето на данната, която е получена току-що. Всеки запис в регистър SSPxBUF по време на приемане/ предаване ще бъде игнориран, като битът за колизия при запис WCOL в регистър SSPxCON1 ще се установи в 1. Потребителската програма трябва да го нулира, за да се разреши следващият запис в SSPxBUF да завърши успешно.

Когато приложният софтуер очаква да пристигнат валидни данни, регистър SSPxBUF трябва да бъде прочетен, преди слдващият байт данни за предаване да бъде записан в SSPxBUF. Бит BF в SSPxSTAT индицира, че SSPxBUF е зареден с приетите данни (предаването е завършено). Когато SSPxBUF бъде прочетен, бит BF се нулира. Тези данни може да са неподходящи, ако SPI е само предавател. Най-общо прекъсванията при MSSP се използват да се определи, кога е завършил процесът на предаване/ приемане. Ако няма да се използват



(1) *Избор за изход за режим хост (главен)*
(2) *Избор за вход за режим клиент (подчинен)*

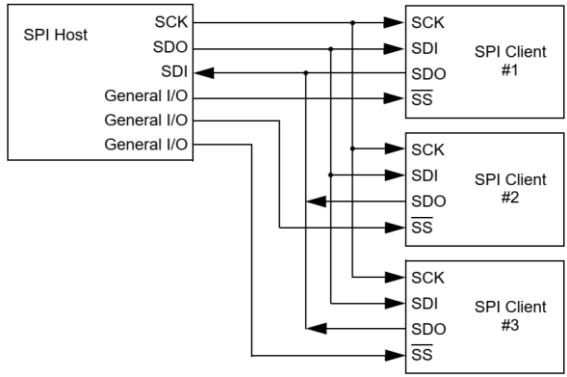
Фиг. 12.2. Блокова схема на блок MSSP в режим SPI

прекъсвания, може да се извърши програмно тестване на флаговете, за да е сигурно, че няма да възникне колизия при запис.

Регистър SSPxSR не е програмно достъпен за четене и запис и до него може да има достъп само през регистър SSPxBUF. Освен това в регистър SSPxSTAT има допълнително флагове за състояние с различно предназначение.

Пример за свързване на хост с няколко клиента е показан на фиг. 12.3.

Главният микроконтролер инициира обмена на данни чрез изпращане на тактов сигнал по линията SCK. Данните преминават през двата 8-битови преместващи регистъра – на хоста и клиента, бит по бит. Извеждането им става по програмирания фронт на тактовия сигнал и се буферират по противоположния фронт на такта. Извеждането започва с най-старшия бит от данната, като едновременно в регистъра започва да постъпва приеманата данна, започвайки от най-младшия бит. И двата микроконтролера трябва да бъдат програмирани за една и съща полярност на тактовия сигнал (CKP), тогава и двата могат да изпращат и приемат данни по едно и също време.



Фиг. 12.3. Свързване на хост с няколко клиента при шина SPI

По време на всеки SPI такт се осъществява напълно дуплексен обмен. Това означава, че докато хостът изпраща най-старшия бит от преместващия си регистър (през своя SDO извод), а клиентът чете този бит и го записва като най-младши в своя преместващ регистър, клиентът също изпраща най-старшия бит от своя преместващ регистър (през своя извод SDO), а хостът го прочита и съхранява като най-младши в своя преместващ регистър. След изпращането по този начин на осем бита, хостът и клиентът са разменили стойностите в своите регистри. Ако има повече данни за обмен, преместващите регистри се зареждат с новите данни и процесът се повтаря.

Дали данните са смислени или не, зависи от приложния софтуер. Следователно може да има три ситуации:

- Хостът изпраща полезни данни, а клиентът - невалидни.
- И хостът, и клиентът изпращат полезни данни.
- Хостът изпраща безсмислени данни, а клиентът - полезни.

Обменът продължава, докато хостът изпраща тактов сигнал и клиентът е избран.

Всяко клиентско устройство, свързано към шината, което не е избрано чрез неговата линия $\overline{CS}$, трябва да игнорира тактовия сигнал и изпращаните данни и самото то не трябва да изпраща данни.

Конфигуриране

Блокът MSSP използва пет регистър при работата си в режим SPI. Това са:

- Регистър за състояние (SSPxSTAT);
- Управляващ регистър 1 (SSPxCON1);
- Управляващ регистър 3 (SSPxCON3);

- Буферен регистър за данни (SSPxBUF);
- Регистър за адрес (SSPxADD);
- Преместващ регистър (SSPxSR) (Не е пряко достъпен).

При инициализацията на SPI трябва да се зададат няколко опции, което се прави чрез съответните управляващи битове (SSPxCON1<3:0> и SSPxSTAT<7:6>). Чрез тях се задава:

- Режим хост (SCK е изход за тактов сигнал);
- Режим клиент (SCK е вход за тактов сигнал);
- Полярност на тактовия сигнал (Неактивно състояние на SCK);
- Моментът на стробиране на входните данни (в средата или в края на продължителността на данните);
- Фронт на тактовия сигнал (извеждане на данни по нарастващ/падащ фронт на SCK);
- Скорост на обмен (само за режим хост);
- Избор на режим подчинен (само за подчинено устройство).

За да се разреши серийният порт, бит SSPEN (SSPxCON1) трябва да се установи в единица. А за да се установи в начално състояние или да се реконфигурира за режим SPI, бит SSPEN трябва да се нулира, да се реинициализират управляващите регистри SSPxCONx и след това да се установи в единица бит SSPEN. Тези действия ще конфигурират изводи SDI, SDO, SCK и $\overline{CS}$, като такива за работа с него. Едновременно с това трябва съответните битове в регистри TRISx да бъдат конфигурирани по подходящ начин:

- За извод SDI съответният TRIS бит трябва да е установен в 1;
- За извод SDO съответният TRIS бит трябва да бъде нулиран;
- За извод SCK (режим главен) съответният TRIS бит трябва да бъде нулиран;
- За извод SCK (режим подчинен) съответният TRIS бит трябва да бъде установен в единица;
- За извод $\overline{CS}$, съответният TRIS бит трябва да бъде установен в единица.

12.2.2. Режим „Главно устройство”

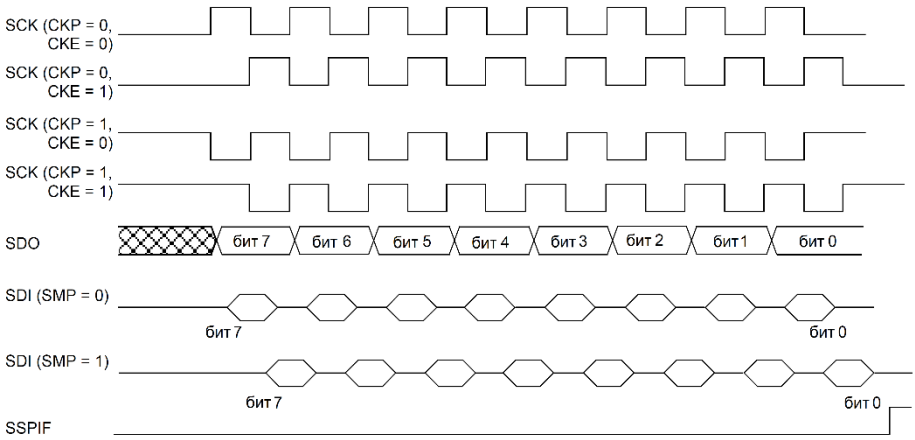
Блок, конфигуриран да работи като главен, е инициатор на обмена, тъй като генерира тактовия сигнал SCK. Той определя кога подчиненият да извършва обмен чрез програмен протокол.

Данните се предават или приемат, когато регистър SSPxBUF бъде записан. Данните в регистър SSPSR ще продължават да се преместват при налични та-

кива на извод SDI с програмираната честота на обмен. При приемането на всеки байт, той ще се прехвърля в регистър SSPxBUF (флагът за прекъсване и тези за състояние ще се установят).

Полярността на тактовия сигнал може да се програмира чрез бит CKP (SSPxCON1), както е показано на фиг. 12.4, където най-старшия бит се предава първи. В главен режим честотата на обмен на SPI може да се програмира:

- $F_{OSC}/4$ (или T_{CY});
- $F_{OSC}/16$ (или $4 \cdot T_{CY}$);
- $F_{OSC}/64$ (или $16 \cdot T_{CY}$);
- Изходната честота от Таймер 2, разделена на 2;
- $F_{OSC}/(4 \cdot (SSPxADD + 1))$.



Фиг. 12.4. Времедиаграми на работата на SPI в режим „главен”

12.2.3. Режим „Подчинено устройство”

В режим „подчинен” данните се предават и приемат, като се използва външен тактов сигнал на входа SCK. След като бъде буфериран и последният бит, флагът за прекъсване SSPxIF се вдига.

В подчинен режим блокът **може да работи в режим SLEEP**. Когато бъде получен байт, ако прекъсването е разрешено, микроконтролерът ще се „събуди”.

Когато модулът SPI е в подчинен режим с разрешен извод $\overline{CS}$, (SSPxCON1<3:0> = 0100), модулът SPI ще се ресетира, ако изводът $\overline{CS}$ се свърже към V_{DD}.

Ако SPI се използва в подчинен режим при CKE=1, то управлението на извод $\overline{CS}$ трябва да бъде разрешено.

Сигналът $\overline{CS}$ може да се използва, за да се гарантира правилният обмен, независио от броя клиентски устройства. В противен случай съществува риск клиентът да излезе от синхронизация с хост-устройството. Ако клиентът пропусне бит, това ще повлияе на всички останали до края на обмена. Използването на сигнала $\overline{CS}$ позволява на клиента и хоста да се синхронизират в началото на всеки обмен.

Шината SPI позволява верижното свързване на множество клиентски устройства. Изходът на първия клиент се свързва с входа на втория, неговият изход се свързва с входа на третия и т.н. Изходът на последния клиент се свързва с входа на хост-устройството. Всеки клиент изпраща по време на втората група тактови импулси точно копие на това, което е получил по време на първата група тактови импулси. Цялата верига работи като дълъг комуникационен преместващ регистър, като се нуждае само от една $\overline{CS}$ линия от хост-устройството.

12.2.4. Ограничения на интерфейса SPI

Синхронните последователни интерфейси, подобни на Microwire и SPI, осигуряват прост и надежден обмен на данни, но имат известни недостатъци:

- Не са особено подходящи за случаите, когато има повече от едно главно устройство, свързано към шината;
- Устройствата не са адресируеми;
- Не е предвидено потвърждение за успешно получено съобщение;
- Не са особено гъвкави – не е толкова лесно да се добави ново устройство, дори и работещо като подчинено – най-малкото е необходима допълнителна линия за избора му.

Тези ограничения са избегнати при следващия разглеждан интерфейс, който поддържа блок MSSP – I^2C.

12.3. Режим I^2C

12.3.1. Същност на интерфейса I^2C

Интерфейсът I^2C (Inter-Inegrated Cicuits) е двупроводен интерфейс, разработен от фирмата Philips Corporation. Предвиден е да осигури връзка между интегралните схеми в рамките на една система, като бъде достатъчно гъвкав и да поддържа различни технологии и скорост на обмен. Първоначалната спецификация е за скорост на обмен 100 Kbps. Поддържа се също разширена такава (бърз режим). Възможна е комуникация на бързи и бавни устройства, свързани към обща шина.

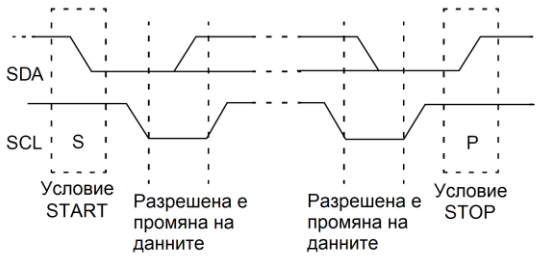
Използва се детайлен протокол, който гарантира надеждния обмен на данните. При предаването на данните едно устройство е „главно” – това, което

започва обмена по шината и генерира тактовия сигнал, докато другото (другите) устройство е „подчинено”. Целият протокол на обмен в „подчиненото” устройство се реализира апаратно, с изключение на функцията „общо извикване (адресиране)”, докато в главното устройство някои части от протокола се реализират програмно.

При интерфейса I²C всяко устройство има адрес. Когато главното устройство иска да започне предаване, първо предава адреса на подчиненото устройство, с което „иска да говори”. Всички устройства, свързани към шината „слушат”, за да установят, дали предаваният адрес е техният. Един бит от адреса определя посоката на данните – дали главното устройство предава към адресираното подчинено или ще чете от него.

Изходните стъпала на линията за тактов сигнал (SCL) и тази за данни (SDA) трябва да са с отворен дрейн или отворен колектор, за да могат да се свържат чрез „жично ИЛИ” към шината. Използват се външни изтеглящи резистори, за да се гарантира високо ниво на линията, когато никое устройство не я изтегля към ниско ниво. Броят на устройствата, които могат да се свързват към I²C шината се ограничава само от нейното максимално допустимо натоварване - 400 pF. По времето, когато няма обмен данни по шината (неактивно състояние), и двете линии – SCL и SDA са във високо ниво.

Началото и края на обмена на данни се определят от съответни условия - START и STOP (фиг. 12.5). START условието се определя като преход от високо в ниско ниво на линията SDA, когато линията SCL е във високо ниво. STOP условието се определя като преход от ниско във високо ниво на SDA, когато SCL е във високо ниво.

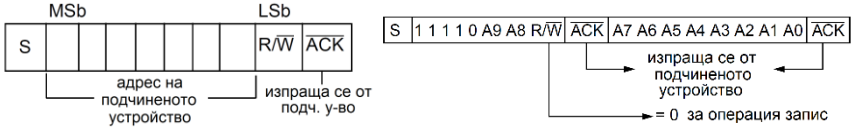


Фиг. 12.5. Условия START и STOP за начало и край на обмена на данни

Възможни са два формата на адреса – 7-битов (Фиг. 12.6 а) и 10-битов (Фиг. 12.6 б).

Всички данни се обменят побайтово, без ограничение в броя на байтовете в едно съобщение. След всеки байт подчиненото устройство генерира бит за потвърждение (ACK). Ако главното устройство не получи потвърждение за приет адрес или данни, трябва да прекрати обмена.

Ако подчиненото устройство се нуждае от време преди обмена на следващ байт, може да задържи линията SCL в ниско ниво, с което ще накара главното да изчака. Обменът ще продължи, когато подчиненото освободи линията SCL.



(а) *(б)*

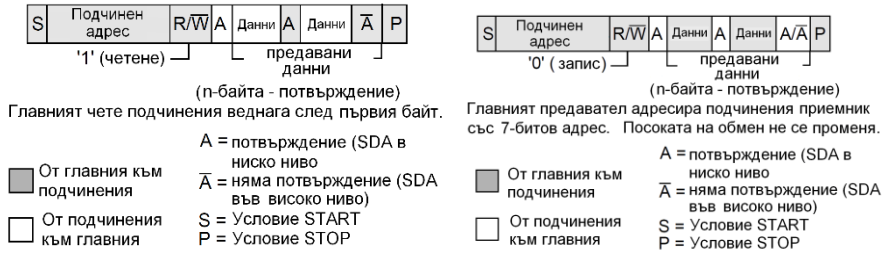
S - условие START; $R/\overline{W}$ *- импулс за четене/запис;* $\overline{ACK}$ *- потвърждение*

Фиг. 12.6. Два формата на адреса на подчиненото устройство – (а) 7-битов и (б) 10-битов

Това времезакъснение ще позволи на подчиненото устройство да буферира приетите данни или да извлече данните, които трябва да изпрати, преди да разреши тактовия сигнал.

Възможно е главното устройство да се нуждае да задържи шината, за да продължи обмена. Тогава то трябва да изпрати следващ стартов бит на мястото на стопов бит или пък последен бит ACK, когато е в режим на приемане.

На фиг. 12.7 са показани примерни последователности за приемане (а) и предаване (б) на данни със 7-битов адрес за главно устройство.



(а) *(б)*

Фиг. 12.7. Примерни последователности за приемане (а) и предаване (б) на данни със 7-битов адрес за главно устройство

Арбитриране

Всяко хост-устройство трябва да наблюдава шината за старт и стоп битове. Ако открие, че тя е заета, не може да изпраща ново съобщение, докато не се освободи. Възможно е обаче, две хост-устройства да започнат обмен по едно или почти едно и също време. Ако това се случи, се налага арбитриране. Всеки предаващ проверява състоянието на линията SDA и го сравнява с това, което очаква да открие. Първият, който открие несъвпадащи състояния, губи арбитрирането и трябва да спре да предава по SDA.

Арбитриране при предаване от страна на клиента също може да се наложи, когато хостът адресира няколко клиента, макар че това е по-рядко срещана ситуация. Ако две главни устройства изпращат съобщения към две клиентски устройства на фазата адресиране, хостът, който изпраща по-малък адрес, печели арбитрирането. Ако два хоста изпращат съобщения до един и същи клиентски адрес и се случи така, че адресите да се отнасят до няколко клиента, процесът на арбитриране трябва да продължи във фазата на обмен на данни.

12.3.2. Използвани изводи и конфигуриране

Блок MSSP, работещ в режим I^{2}C, поддържа всички функции за подрежими „главен" и „подчинен" (включително „общо извикване"). Поддържа се също възможността за генериране на апаратни прекъсвания при START и STOP битовете, за да се определи дали шината е свободна (при наличие на повече от едно главни устройства). Поддържат се спецификациите за стандартен режим, както и 7-битово и 10-битово адресиране.

Когато изводи SCL и SDA са входове, към тях има свързан филтър, който работи в режимите 100 kHz и 400 kHz. Когато тези изводи са изходи в режим 100 kHz, има възможност за контрол на стръмността на фронтовете, независимо от системната честота.

За обмен на данни се използват два извода – SCL, за тактова честота, и SDA - за данни. Те се конфигурират автоматично при разрешаване на режим I^{2}C. Блокът MSSP се разрешава чрез установяване в единица на бит SSPEN (SSPxCON1).

В режим I^{2}C се използват седем програмно достъпни регистъра:

- Три управляващ и регистъра – SSPxCON1, SSPxCON2 и SSPxCON3;
- Регистър за състояние - SSPxSTAT;
- Регистър с маска за стойността в SSPxSR, използвана при сравнение за откриване на адреса на подчинено устройство – SSPxMSK;
- Буферен регистър за сериен обмен – SSPxBUF;
- Преместващ регистър - SSPSR (не е програмно достъпен);
- Регистър за адрес и коефициент на делене за генератора на честота за обмен – SSPADD.

Чрез битове в регистър SSPxCON1 могат да се избират **шест различни варианта** (подрежима) на режим I^{2}C:

- I^{2}C подчинен (7-битов адрес), с разрешени прекъсвания при старт и стоп битове;
- I^{2}C подчинен (10-bit адрес), с разрешени прекъсвания при старт и стоп битове;

- I^2C главен, управляван от фърмуера (клиентът е в състояние на изчакване);
- I^2C главен, тактова честота = OSC/4 (SSPxADD +1);
- I^2C подчинен (7-битов адрес);
- I^2C подчинен (10-bit адрес).

Преди задаване на режим I^2C, изводите SCL и SDA трябва да бъдет програмирани като входове чрез съответните битове в регистри TRIS. При избиране на режим I^2C и установяване на бит SSPEN бит в единица, изводите SCL и SDA се конфигурират съответно като такива за тактов сигнал и за данни. За правилната работа на блока на тези изводи трябва да се предвидят външни изтеглящи към захранване резистори.

Чрез бит CKE (SSPxSTAT) се задават нивата на изводите SDA и SCL и за двата подрежима – главен и подчинен – да отговарят на спецификациите за SMBus или за I^2C.

В регистър SSPxSTAT се съдържа разнообразна информация за процеса на обмен – дали е открит START (S) или STOP (P) бит, дали приетият байт е данни или адрес, дали следващият байт е завършекът на 10-битов адрес и дали операцията е запис или четене.

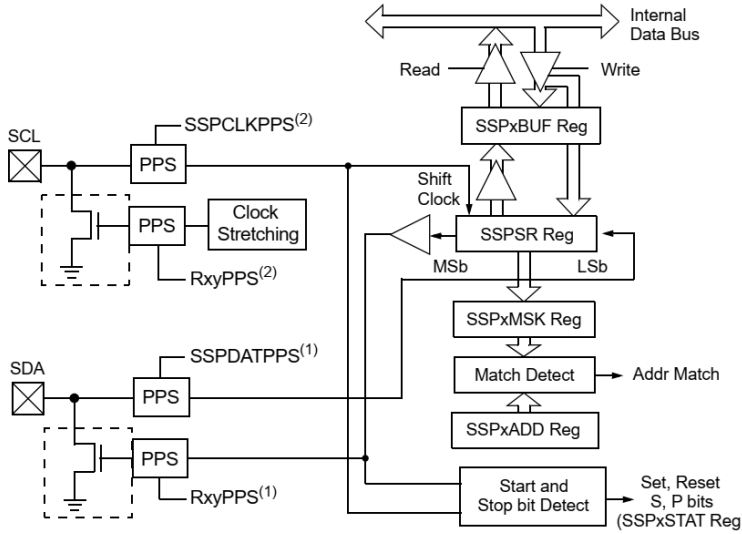
Регистър SSPxBUF е буферният регистър, в който се записват данните за извеждане или приетите данни. През регистър SSPSR данните се изместват бит по бит в посока навътре или навън в/от модула. При приемане регистри SSPxBUF и SSPSR формират двойно буфериран приемник. Това позволява да започва приемането на следващ байт преди предишният приет да е прочетен, без да се загуби. При приемането на байта се вдига флагът SSPIF. Ако бъде напълно приет следващ байт, преди прочитането на SSPBUF, ще настъпи препълване и ще се вдигне също флагът SSPOV (SSPCON<6>), а байтът в регистър SSPSR ще бъде загубен.

Регистър SSPxADD съдържа адреса на подчиненото устройство. При подрежим с 10-битов адрес, потребителят трябва да запише старшия байт на адреса, който има следната стойност - 1111 0 A9 A8 0. След съвпадение на старшата част на адреса се зарежда и младшата - A7:A0.

12.3.3. Режим „Подчинено устройство”

Блоковата схема на блок MSSP за режим I^2C, подчинен, е показана на фиг. 12.8.

В подчинен режим изводи SCL и SDA трябва да се конфигурират като входове. След като бъде разрешен блокът MSSP, той очаква да настъпи условие START, след което 8 бита се записват последователно в регистър SSPSR. При

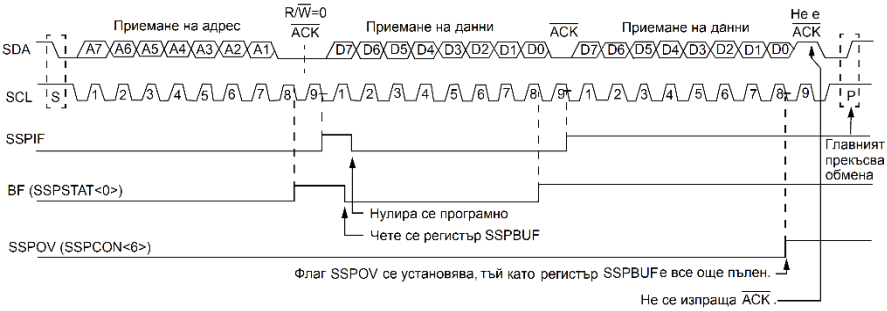


(1) Избраният извод за SDA трябва да е един и същ за вход и изход
(2) Избраният извод за SCL трябва да е един и същ за вход и изход

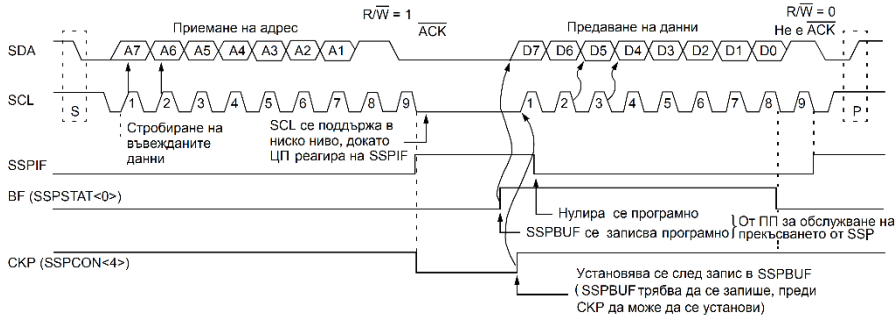
Фиг. 12.8. Структурна схема на блок MSSP за режим I²C, подчинено устройство

съвпадение на адрес или приета данна след приет валиден адрес, блокът генерира автоматично импулс за потвърждение (Acknowledge - ACK) и след това зарежда регистър SSPxBUF с приетата в регистър SSPSR данна.

На фиг. 12.9 и 12.10 са показани времедиаграми на процесите на приемане и предаване на блока в режим I²C като подчинен, като примерите са за 7-битови адреси.



Фиг. 12.9. Времедиаграми при приемане, режим I²C, подчинено устройство, 7-битов адрес



Фиг. 12.10. Времедиаграми при предаване, режим I²C, подчинено устройство, 7-битов адрес

В режим SLEEP блокът може да приема адреси и данни. При съвпадение на адрес или завършване на приемането на байт, ако е разрешено прекъсване от SSP, микроконтролерът ще се „събуди”.

Обикновено при обмена на данни се адресира само едно подчинено устройство. Изключение е изпращането от главното устройство на адрес за **общо извикване**, при което се адресират всички устройства, които от своя страна трябва да отговорят с потвърждение. Това е резервиран адрес в I2C протокола. Състои се от нули и се придружава от сигнал R/$\overline{W}$=0. Ако трябва да се разпознава той от подчинено устройство, е необходимо бит *GCEN (SSPxCON2*) да бъде установен в единица.

12.3.4. Режим „Главно устройство”

Режимът се избира чрез задаване на подходяща комбинация от стойности в битове SSPM на регистър SSPxCON1 и установяване в 1 на бит SSPEN.

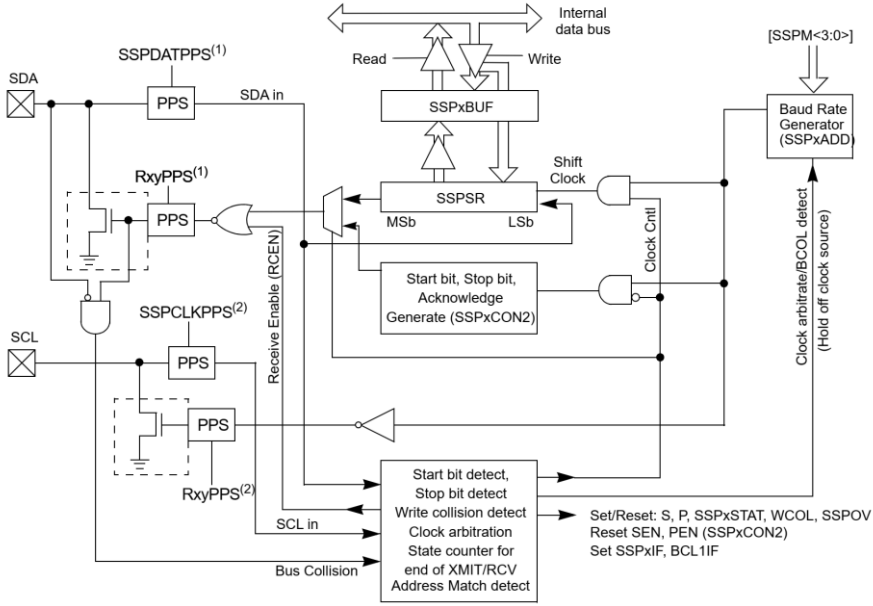
Изводи SCL и SDA трябва да са конфигурирани като входове.

Блоковата схема на блок SSP в режим I²C, главен, е дадена на фиг. 12.11.

Управлението на I²C шината може да се поеме от главното устройство, когато бит P е установен в единица или шината е свободна – битове S и P са нули. Възможността да се определи, дали общата шина е свободна, позволява включването на **повече от едно главно устройство** към нея.

Тактовата честота за обмен се генерира в главното устройство от специален генератор BRG (Baud Rate Generator). Стойността, която се зарежда в него се съдържа в младшите 7 бита на регистър *SSPADD*.

Флагът за прекъсване *SSPIF* може да се установи (и да се генерира прекъсване, ако е разрешено) при следните събития:



(1) Избраният извод за SDA трябва да е един и същ за вход и изход
(2) Избраният извод за SCL трябва да е един и същ за вход и изход

Фиг. 12.11. Структурна схема на модул SSP в режим I²C, главно устройство

- Условие START;
- Условие STOP;
- Приет/ предаден байт;
- Прието/ предадено потвърждение;
- Повторен старт (START).

В режим SLEEP модулът I²C може да приема адреси и данни, при което, ако прекъсванията са разрешени, микроконтролерът ще се „събуди".

По-подробна информация относно действията при приемане и предаване в режим „главен", работата на блока при наличие на повече от един главен, някои съображения при свързване на модулите към обща I²C шина и др., са дадени в [7].

Въпроси за самоконтрол

1. Каква е същността на последователния (серийния) и паралелния обмен на данни?

2. По какво се различават синхронният и асинхронният последователен обмен на данни? А полудуплексен и напълно дуплексен?
3. В какви режими може да работи блок MSSP?
4. Посочете някои основни характеристики на блока при отделните режими на работа.
5. Изяснете всеки от режимите и подрежимите му на работа, като ползвате дадените блокови схеми. Какви са техните предимства и недостатъци?
6. Какъв е форматът на обменяните данни в различните режими?
7. В кои случаи при работата на блок MSSP се генерират прекъсвания?
8. Може ли блок MSSP да работи в режим Sleep и при какви условия (режими)?

13. Блок разширен универсален синхронно–асинхронен приемо–предавател (EUSART)

13.1. Предназначение и режими на работа на блок EUSART

Блок EUSART (Enhanced Universal Synchronous Asynchronous Receiver Transmitter) е един от двата блока за I/O обмен на данни в последователен (сериен) код. Той още е познат под името сериен комуникационен интерфейс (Serial Communications Interface - SCI).

Блок EUSART може да бъде конфигуриран в един от следните режими на работа:

- **Асинхронен режим** (напълно дуплексен режим);
- **Синхронен режим** (полудуплексен режим).

От своя страна синхронният режим може да бъде:

- *Главен синхронен режим (host)*, когато МК съдържащ EUSART изработва синхроимпулсите при приемане/предаване на данни;
- *Подчинен синхронен режим (client),* когато МК съдържащ EUSART приема синхроимпулси, изработени от външен източник (МК, устройство, система) при приемане/предаване на данни.

Асинхронният режим може да се използва за връзка с периферни устройства като CRT терминали или персонални компютри.

Синхронният режим на работа може да се използва за комуникация с периферни блокове като АЦП, ЦАП, серийни EEPROM и други.

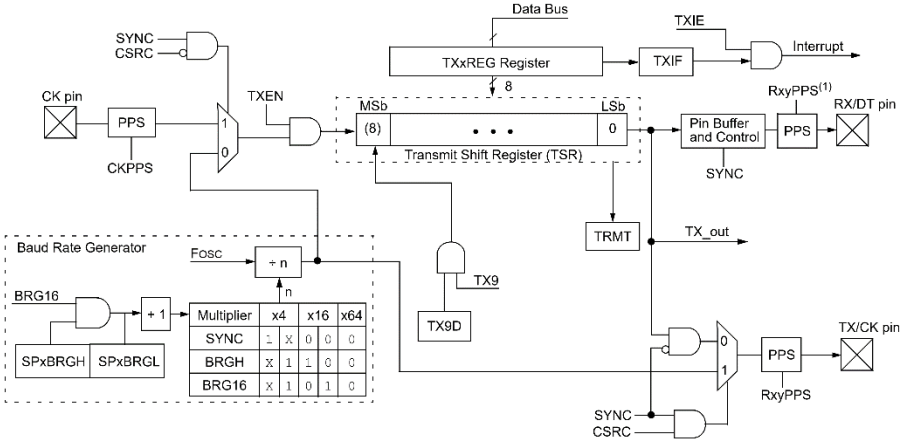
Блок EUSART има следните основни характеристики:

- Напълно дуплексен режим на асинхронно предаване и приемане;
- Входен буфер за два символа;
- Изходен буфер за един символ;
- Програмируем 8-битов или 9-битов формат на символите;
- Детектиране на адрес в 9-битов режим;
- Регистриране на грешка при препълване на входния буфер;
- Регистриране на форматна грешка при приемане на символ;
- Режими на полудуплексен синхронен обмен за главно и за подчинено устройство;
- Програмируема полярност на тактовия сигнал в синхронен режим;
- Възможност за работа в режим Sleep.

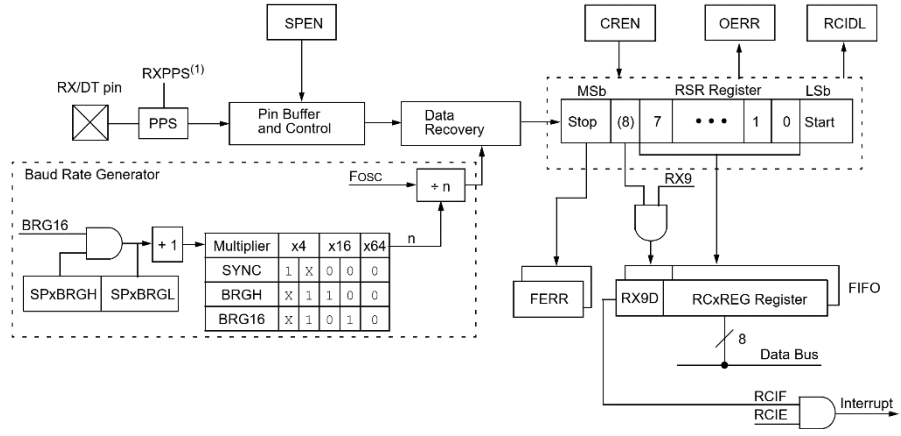
Няколко допълнителни възможности на EUSART го правят много подходящ за реализация на интерфейс LIN (Local Interconnect Network):

- Автоматично откриване и калибриране на честотата на обмен;
- Събуждане при приемане на сигнал Break;
- Предаване на 13-битов символ Break.

На фиг. 13.1 и 13.2 са дадени блоковите схеми съответно на предавателя и приемника на блок EUSART.



Фиг. 13.1. Блокова схема на предавателя на блок EUSART



Фиг. 13.2. Блокова схема на приемника на блок EUSART

Изходът от предавателя TX_out се свързва с извод TX/CK или вътрешно с конфигурируема логическа клетка (вж. *Тема 14. Конфигурируеми логически клетки*).

Работата на блока се управлява от три регистъра - TX1STA, RC1STA и BAUD1CON. Назначаването на входно-изходни изводи за необходимите сигнали се извършва чрез съответните им PPS регистри.

Блок EUSART може също да се използва за комуникация в мултипроцесорни системи, като има възможност за откриване (детектиране) на 9-битов адрес.

13.2. Асинхронен режим на работа на блок EUSART

В този режим блок EUSART използва стандартен NRZ (non-return-to-zero) формат (един стартов бит, 8 или 9 бита данни и един стопов бит), като най-често се обменят 8 бита данни (вж. фиг. 12.1).

Форматът NRZ се реализира с две нива: високо логическо ниво - V_{OH} (Mark), което представлява бит данни логическа „1“ и ниско логическо ниво - V_{OL} (Space), което представлява бит данни логическа „0“. Форматът NRZ – без връщане към нулата, се състои в това, че при последователно предавани битове с една и също стойност, линията остава в изходното ниво (високо или ниско), без състоянието ѝ да се променя до неутрално ниво за определено време между два бита. Неактивното състояние на линията (когато не се извършва обмен) е високо ниво (т. нар. маркерна единица). Форматът на всеки предаван символ се състои от стартов бит, следван от 8 или 9 бита данни и винаги завършва с един или повече стопови бита. Стартовият бит винаги с ниско ниво, а стоповите – винаги във високо ниво.

За получаване на стандартни честоти за предаване може да се използва вграденият 8/16–битов генератор BRG. Блок EUSART предава и приема първо младшите битове на данните. Предавателят и приемникът са функционално независими един от друг, но използват един и същ формат на данните и една и съща честота на обмен. Контрол по четност на данните не се поддържа апаратно, но може да се реализира програмно, като за целта се използва деветият бит данни.

Асинхронен предавател

Предавателят на EUSART се разрешава за асинхронен обмен чрез конфигуриране на битове:

- TXEN = 1 (регистър TX1STA) – разрешава предавателя;
- SYNC = 0 (регистър TX1STA) – задава асинхронен режим на обмен;
- SPEN = 1 (регистър RC1STA) – разрешава EUSART и конфигурира извод TX/CK като изход. Ако това е извод, споделящ и аналогова функция, то тя трябва да се забрани чрез съответния ANSEL бит.

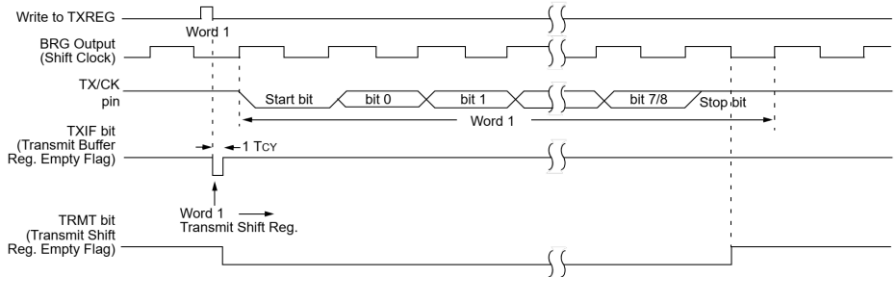
Полярността на предаваните данни може да се управлява чрез бит SCKP в регистър BAUD1CON.

Ядрото на предавателя е серийният преместващ регистър TSR. Той получава данни в паралелен код от буферния регистър TXREG, който от своя страна

се зарежда с данни програмно. Предаването започва, когато се запише данна в TXREG. Регистърът TSR не се зарежда, докато не се изпрати стоповия бит на предходната данна. Веднага след това регистърът TSR се зарежда с нови данни от регистър TXREG (ако има такива). Регистър TSR не е програмно достъпен. След като данните се прехвърлят от TXREG в TSR, регистър TXREG е празен и флагът TXIF (PIR3) се установява в единица. Това прекъсване може да се забрани или разреши чрез бит TXIE (PIE3). Флагът TXIF се установява в единица, независимо от състоянието на разрешаващия бит TXIE, и не може да се нулира по програмен път, а само при запис на нови данни в регистър TXREG. Този флаг, обаче се установява в единица, само ако бит TXEN=1.

Докато флаг TXIF показва състоянието на регистър TXREG, то бит TRMT (TX1STA) показва състоянието на преместващия регистър TSR. Бит TMRT е само за четене и се установява в единица, когато се изпразни преместващия регистър TSR. Това събитие не предизвиква заявка за прекъсване, така че ако потребителят иска да провери дали преместващият регистър TSR е празен, трябва го направи по програмен път.

На фиг. 13.3 е е показан пример за предаване на един символ.



Фиг. 13.3. Времедиаграми при предаване на един символ в асинхронен режим на обмен

За да се избере 9-битов формат на данните за предаване, трябва бит TX9 (TX1STA) да се установи в единица. Деветият бит трябва да бъде записан в TX9D (TXSTA), преди да се запишат останалите осем бита в регистър TXREG. Това е необходимо, защото записът на данни в регистър TXREG ще доведе до незабавно прехвърляне на данните в регистър TSR, ако той е празен. По такъв начин в регистър TSR може да бъде зареден неправилен девети бит.

Съществува също така режим на предаване на 9-битов адрес при наличие на множество приемници.

Асинхронен приемник

Предавателят на EUSART се разрешава за асинхронен обмен чрез конфигуриране на битове:

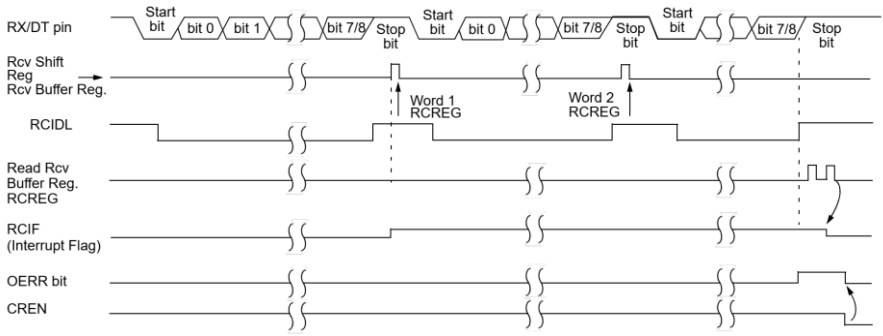
- CREN = 1 (регистър RC1STA) – разрешава приемника;
- SYNC = 0 (регистър TX1STA) – задава асинхронен режим на обмен;
- SPEN = 1 (регистър RC1STA) – разрешава EUSART и конфигурира извод RX/DT като вход. Ако това е извод, споделящ и аналогова функция, то тя трябва да се забрани чрез съответния ANSEL бит.

Данните постъпват на извод RX/DT и управляват блока за възстановяване на данните. Той представлява високо-скоростен преместващ регистър, работещ с честота, 16 пъти по-висока от тази на предаване, докато основният приемащ сериен преместващ регистър работи с честотата на обмен.

Ядрото на приемника е преместващият сериен регистър RSR. След откриване на стопов бит, приетите данни в регистър RSR се прехвърлят в регистър RCREG, ако той е празен. При завършване на приемането се вдига флаг RCIF в регистър PIR3 (фиг. 13.4). Обслужването на прекъсването може да бъде разрешено/ забранено чрез бит RCIE на регистър PIE3. Флагът RCIF е само за четене и се нулира апаратно, когато се прочете регистър RCREG и е вече празен. Регистър RCREG е двойно буфериран, т.е. той е памет тип FIFO с две нива. Ето защо е възможно в регистър RCREG да бъдат приети и записани две последователни данни и да започне приемането на трета в регистър RSR. При откриване на стоповия бит на третия байт, ако регистър RCREG е все още пълен, се вдига флагът за препълване OERR (RC1STA). Регистърът RCREG може да бъде прочетен два пъти за получаване на двата символа от FIFO, но докато не се изчисти флагът за препълване, нови няма да се приемат. Флагът за препълване OERR може да се нулира или чрез нулиране на бит CREN (RC1STA), или чрез ресетиране на блока EUSART посредством нулиране на бит SPEN (RC1STA).

Флагът за форматна грешка FERR (RC1STA) се установява в единица, ако се открие стопов бит, равен на нула. Битът FERR и деветият приеман бит се буферират по същия начин, както приеманите данни. Четенето на регистър RCREG ще зареди битове RX9D и FERR с нови стойности, ето защо е важно бит FERR да се прочете преди четене на регистър RCREG, за да не се изгуби старата информация за битове RX9D и FERR.

Блок EUSART поддържа приемане на 9-битови данни, което се задава с установяване в 1 на бит RX9 на регистър RC1STA. Деветият, най-старши бит на приеманите данни се записва в бит RX9D на регистър RC1STA. При четене на 9-битови данни от FIFO буфера на приемника, бит RX9D трябва да се прочете преди осемте младши бита, намиращи се в регистър RCREG.



На времедиаграмата са показани три последователно получени думи на вход RX с четене на регистър RCREG след третата, което води до вдигането на флага OERR.

Фиг. 13.4. Времедиаграми при приемане в асинхронен режим на обмен

Детектиране на адрес

Когато в системата присъстват повече от един приемника, например в такива с интерфейс RS-485, за разпознаване на устройствата при обмен може да се използва специален режим с детектиране на адрес. Той се разрешава с установяване в 1 на бит ADDEN (RC1STA). Той изисква задаване на обмен на 9-битови данни. Всички други символи се игнорират. При получаване на адрес се проверява програмно, дали съответства на този на приемника. При съвпадение софтуерът трябва да забрани режима за детектиране на адрес чрез нулиране на бит ADDEN преди пристигането на следващия стопов бит. Когато потребителската програма открие края на съобщението, приемникът отново трябва да се постави в режим на детектиране на адрес чрез установяване на бит ADDEN в 1.

13.3. Генератор на честота за обмен

Генераторът за честота на обмен BRG е 8-битов или 16-битов таймер, който се използва както при асинхронен, така и при синхронен режим на работа на блок EUSART. По подразбиране той е 8-битов. Шестнадесетбитов режим се избира с установяване в 1 на бит BRG16 (BAUD1CON). Регистровата двойка SPBRGH, SPBRGL определя периода на на BRG. В асинхронен режим множителят на честотата за обмен се определя от битове BRGH (TX1STA) и BRG16 (BAUD1CON). В синхронен режим бит BRGH не се използва.

В табл. 13.1 са показани формулите за изчисление на скоростта на обмен при зададена тактова честота и желана скорост на обмен, които са валидни за различните режими. Предимството да се използва висока скорост на обмен, дори и необходимост от ниска такава, е в намаляването на стойността на грешката при изчисленията.

Таблици с изчислени константи за регистри SPBRG, SPBRGL за примерни стандартни скорости на обмен и определени тактови честоти са дадени в [7].

Табл. 13.1. Формули за изчисляване на скоростта на обмен

Конфигурационни битове			Режим на BRG/ EUSART	Формула
SYNC	BRG16	BRGH		
0	0	0	8-битов/ асинхронен	$F_{OSC}/[64(n+1)]$
0	0	1	8-битов/ асинхронен	$F_{OSC}/[16(n+1)]$
0	1	0	16-битов/ асинхронен	
0	1	1	16-битов/ асинхронен	$F_{OSC}/[4(n+1)]$
1	0	x	8-битов/ асинхронен	
1	1	x	16-битов/ асинхронен	

x – стойността е без значение
n е стойността в регистрова двойка SPBRGH, SPBRGL

При запис на нова стойност в регистри SPBRG, SPBRGL таймерът на генератора се нулира, така че обменът на данни започва веднага с новата определена скорост.

Ако честотата на обмен се промени по време на приемане на символ, той може да се загуби или да възникне грешека при приемане. Това може да се предотврати чрез тестване на бит RCIDL.

Приложение за реализация на шина LIN

- **Автоматично откриване на честотата на обмен**

Блок EUSART поддържа режим на автоматично откриване и калибриране на честотата на обмен (Auto-Baud Detect - ABD). В този режим посоката на тактовия сигнал е към BRG. Вместо BRG да тактува пристигащия по RX сигнал, сигналът по RX управлява BRG. Генераторът се използва да определи периода по приетия символ 55h (ASCII „U”), който е символът за синхронизация при шина LIN. Уникалната особеност на този символ е последователността от пет предни фронта, включително и този на стоповия бит. Тази възможност улеснява приложението на блок EUSART за реализация на шина LIN.

- **Изпращане на символ Break**

Блок EUSART може да изпраща специални символи Break, изисквани по стандарта на шина LIN. Символът Break се състои от стартов бит, следва от 12 нулеви бита и стопов бит.

За да се изпрати символ Break, трябва да се установят в 1 битове SENDB и TXEN в регистър TX1STA. След това предаването на символа Break се инициира чрез запис в регистър TXREG. Стойността, записана в TXREG се игнорира, а се изпращат само нули.

Бит SENDB се ресетира автоматично апаратно след изпращането на съответния стопов бит. Това позволява на потребителя да презареди FIFO със следващия изпратен байт, следващ след символа Break character (обикновено символа за синхронизация в спецификацията на шина LIN).

Бит TRMT в регистър TX1STA индицира, че има активна операция на предаване или неактивно състояние (idle), точно както по време на нормално предаване.

Работа по време на режим Sleep

По време на режим Sleep всички тактови сигнали към EUSART са спрени. Поради това генераторът на честота за обмен е неактивен и не може да се извърши приемане на символи. Функцията за автоматично събуждане (Auto-Wake-up) позволява на контролера да се събуди при активност на линията RX/DT. Тя налична само в асинхронен режим на работа.

Функцията за автоматично събуждане се разрешава чрез установяване в 1 на бит WUE в регистър BAUD1CON. При това операциите за нормално приемане на символи по RX/DT се забранява и EUSART остава в чакащ режим, наблюдавайки за събитие за автоматично събуждане, независимо от режима на процесора. Събитието за автоматично събуждане се състои от преход от високо в ниско ниво на линията RX/DT. (Това съвпада с началото на Sync Break символ-сигнал за събуждане при LIN протокол.) При събитие за събуждане EUSART генерира заявка за прекъсване в RCIF.

За да се избегнат грешки в символите или фрагментиране на символи по време на събитие за събуждане, символът, предизвикващ събуждане, трябва да се състои само от нули. Следователно първият символ, който се предава, трябва да се състои от нули. Това е време с продължителност 10 или повече бита, препоръчително 13 бита при шина LIN или различен брой битове при устройства, комуникиращи по интерфейс RS-232.

13.4. Синхронен режим на работа на блок EUSART

Синхронната серийна комуникация се използва обикновено в системи с едно главно устройство (хост) и едно или повече подчинени (клиенти). Честотата за обмен се генерира в главното устройство и тя се осигурява за всички устройства в системата.

В синхронен режим се използват две сигнални линии: двупосочна линия за данни и линия за тактов сигнал. Клиентите използват външен такт, осигуряван от хоста, при приемане и предаване на данните. Тъй като данните са двупосочни, синхронният обмен е само полудуплексен. В синхронен обмен не се използват стартови и стопови битове.

Блокът EUSART може да работи и като хост, и като клиент.

13.4.1. Синхронен главен режим

За работа на EUSART в синхронен главен режим, е необходимо да се конфигурират следните битове:

- SYNC = 1 (регистър TX1STA) – за синхронен режим;
- CSRC = 1 (регистър TX1STA) – за конфигуриране на блока като главно устройство;
- SREN = 0 (регистър RC1STA) - за предаване; SREN = 1 - за приемане;
- CREN = 0 (регистър RC1STA) - за предаване; CREN = 1 - за приемане;
- SPEN = 1 (регистър RC1STA) – за разрешаване на EUSART.

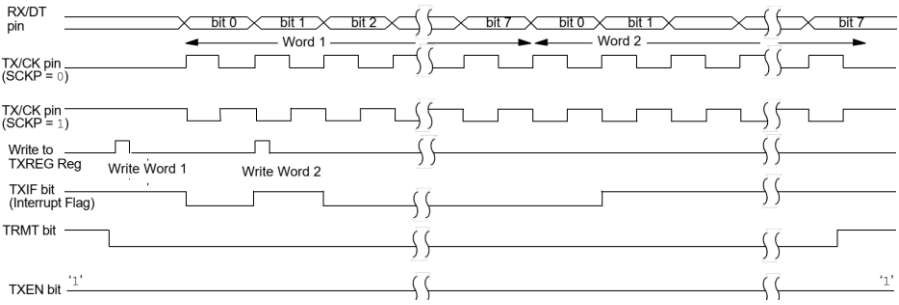
При това линията TX/CK се конфигурира за предаване на тактов сигнал, а RX/DT за обмен на данни. Ако са им присвоени изводи, споделящи и аналогови функции, то те трябва да се забранят чрез съответните ANSEL битове.

Поляритетът на тактовия сигнал може да се избира – възможност, предвидена за съвместимост с интерфейс Microwire. Той се избира чрез бит SCKP в регистър BAUD1CON. При бит SCKP, установен в 1, неактивното състояние на линията е високо ниво, а данните се обменят по падащ фронт на всеки тактов импулс. Ако SCKP=0, неактивното състояние на линията за тактов сигнал е ниско ниво, а данните се обменят по преден фронт на тактовия сигнал.

Предаване в синхронен главен режим

Предаването ще започне, когато в регистър TXREG бъдат записани данни. Ако регистър TSR все още съдържа част от или цял символ, данната ще се задържи в TXREG до изпразването на TSR.

На фиг. 13.5 е показано предаване на две 8-битови думи.



Фиг. 13.5. Времедиаграми при предаване в синхронен режим на обмен

Приемане в синхронен главен режим

След като е избран синхронен режим, приемането се разрешава чрез установяване на бит SREN (RC1STA) или на бит CREN (RC1STA) в единица. Ако SREN=1, а CREN=0, се приема само една дума, като се генерира тактов сигнал само за нея. При това, при завършване на приемането, SREN автоматично се нулира. Ако CREN=1 приемането продължава, докато CREN не се нулира. Ако и двата бита са единица, тогава бит CREN е с приоритет, като SREN се нулира след приемането на първата дума.

Механизмът на функциониране на приемника в този режим е същият, както при асинхронен режим, с изключение на флаг FERR, който не се използва в синхронен режим, поради различния формат на обменяните данни.

13.4.2. Синхронен подчинен режим

Синхронният подчинен режим се различава от синхронния главен по източника на тактова честота за обмен. В подчинен режим модул EUSART получава синхроимпулси отвън през извод TX/CK. Това позволява микроконтролерът да приема данни по време на режим SLEEP.

За работа на EUSART в синхронен подчинен режим, е необходимо да се конфигурират битове SYNC, SREN, CREN и SPEN по аналогичен начин, както за работа синхронен главен режим, с изключение на бит CSRC, който трябва за режим на клиент да бъде нулиран.

Принципът на работа, както и последователността на конфигуриране на предаване и приемане на данни в този режим са почти идентични, както при синхронен главен режим [7].

Въпроси за самоконтрол

1. В какви режими може да работи блок EUSART?
2. Какво представлява генераторът за честота BRG и кои режими на работа на блок EUSART поддържа?
3. Какъв формат на данните се поддържа от блок EUSART при асинхронен обмен?
4. Изяснете механизма на функциониране на предавателя и приемника, като ползвате дадените блокови схеми.
5. Какви грешки се регистрират при асинхронен и при синхронен обмен?
6. Може ли блок EUSART да работи в режим SLEEP и кога?
7. В какви случаи се генерират заявки за прекъсване?
8. Какви стандартни интерфейси за обмен на данни могат да се реализират с използване на блок EUSART?

14. Конфигурируеми логически клетки

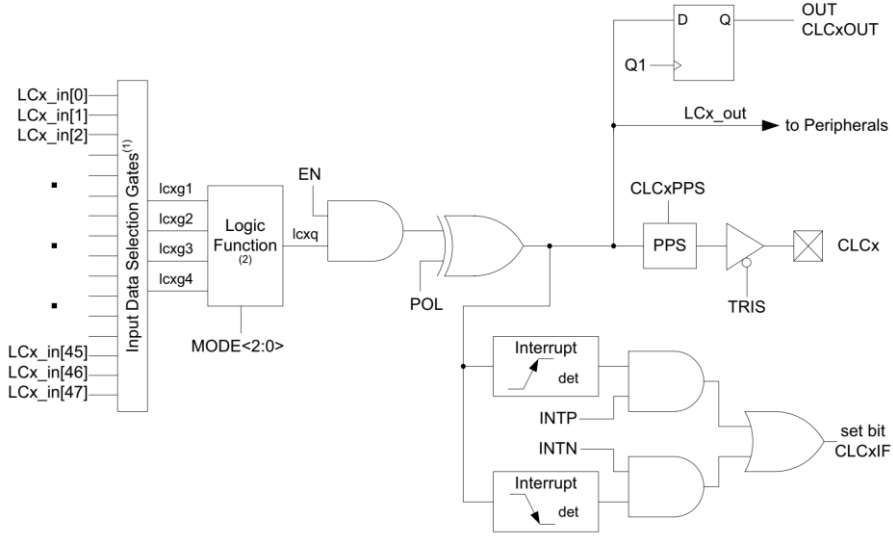
Блокът с конфигурируеми логически клетки (Configurable Logic Cell - CLCx) включва четири клетки програмируема логика, които работят с честота, извън ограниченията за процесора при изпълнение на програмата. Те имат възможност за до 32 входни сигнала и чрез кофигурируеми логически елементи ги редуцират до четири сигнала с логически нива, които управляват една от осем избираеми логически функции с един изход.

Източниците на входни сигнали могат да бъдат комбинации от следните такива:

- I/O изводи;
- Вътрешни тактови сигнали;
- Периферни блокове;
- Регистрови битове.

Изходният сигнал може да бъде насочен вътрешно към периферен блок и изход на МК.

На фиг. 14.1 е показана опростена функционална схема на пътя на сигналите през CLCx.



Фиг. 14.1. Функционална схема на конфигурируема логическа клетка

Възможните конфигурации на CLCx са:

- Комбинационна логика: AND, NAND, AND-OR, AND-OR-INVERT, OR-XOR, OR-XNOR;
- Тригери: SR, синхронен D тригер със Set и Reset входове, D тригер с разрешаващ вход и със Set и Reset входове, синхронен JK с Reset вход.

Програмирането на блок CLCx става чрез конфигуриране на четирите стъпала на пътя на логическия сигнал, които са:

- Избор на данни;
- Логическо стъпало;
- Избор на логическа функция;
- Полярност на изхода.

Всяко стъпало с е конфигурира чрез съответните специални регистри на CLCx. Това дава допълнително предимство, да може CLCx да се реконфигурира по време на изпълнение на програмата.

Четири 32-входни мултиплексора се използват за избор на входни сигнали към следващото стъпало. Изходите от тях се насочват към входа на желаната логическа функция посредством логическо стъпало. Всяко такова стъпало може да насочва всяка комбинация от четирите избрани входа. Логическото стъпало не изпълнява само трасираща функция. То може да се конфигурира да насочи всеки входен сигнал като инвертирани или неинвертирани данни. Насочените сигнали се умножават логически във всяко такова стъпало. Изходът от него може също да се инвертира, преди да премине към стъпалото за изпълнение на логическа функция.

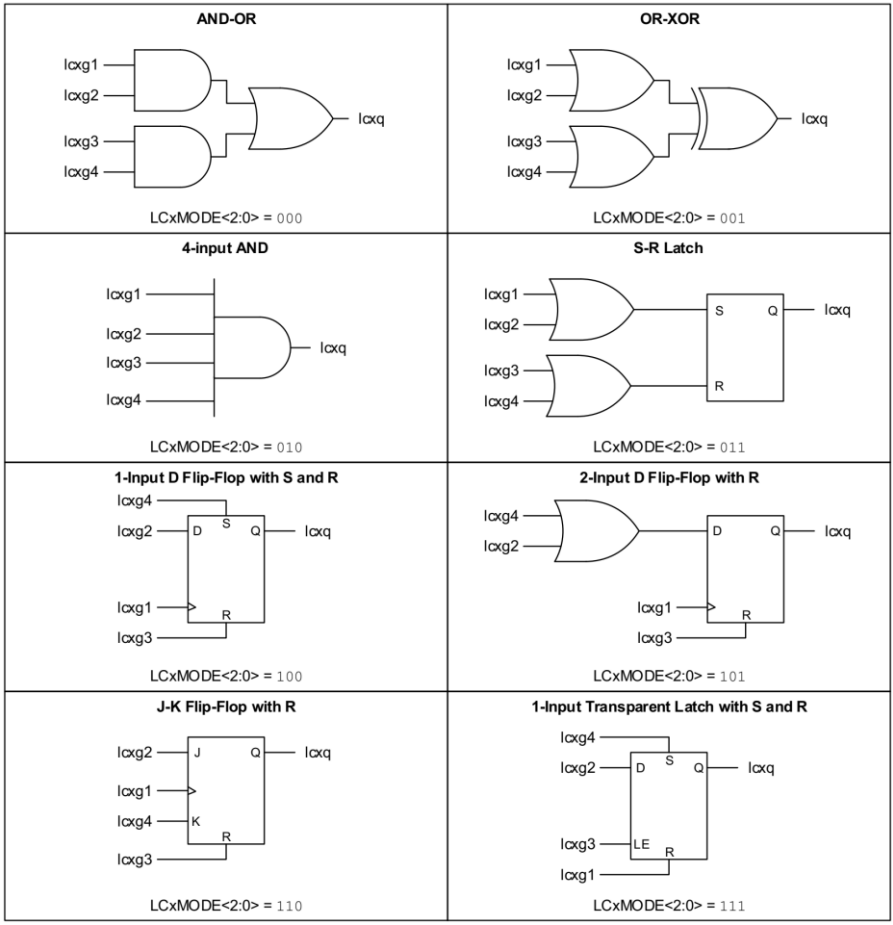
На кратко това стъпало представлява логически елемент AND/NAND/ OR/NOR с 1 до 4 входа. Когато всеки вход, както и изходът са инвертирани, се изпълнява логическа операция OR на всички разрешени входове за данни. Когато входовете и изходът не са инвертирани, се изпълнява логическа операция AND на всички разрешени входове.

Логическите функции са осем на брой:

- AND-OR;
- OR-XOR;
- AND;
- асинхронен SR тригер;
- синхронен D тригер със Set и Reset входове;
- синхронен D тригер с вход Reset;
- синхронен JK тригер с вход Reset;

- асинхронен D тригер с разрешаващ вход и Set и Reset входове.

Възможните логически функции са показани на фиг. 14.2. Всяка логическа функция има четири входа и един изход. Четирите входа са четирите изхода на предходното логическо стъпало. Изходният сигнал се трасира до стъпалото за инверсия и оттам до други периферни блокове, изход на МК и обратно към самата CLCx.



Фиг. 14.2. Програмируеми логически функции на CLCx

Последното стъпало на CLCx е това за полярността на изходния сигнал. Установяване в 1 на бит LCxPOL в регистър CLCxPOL инвертира нивото на изходния сигнал. Промяната на полярността, докато прекъсванията са разрешени, ще доведе до заявка за прекъсване.

Прекъсвания от CLCx

При промяна на изходното ниво на CLCx се генерира заявка за прекъсване. За целта всяка CLCx клетка има детектиращи схеми за преден и заден фронт. За обслужването му трябва също да е установена в 1 съответната маска.

Работа по време на режим Sleep

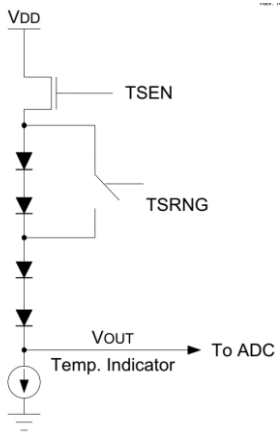
Блок CLC работи независимо от системния такт и ще продължи да работи по време на режим Sleep, ако източниците на входни сигнали са активни. Също така HFINTOSC остава активен, когато блокът CLC е разрешен и той е избран като източник на тактов сигнал, независимо от избрания системен такт.

15. Температурен индикаторен блок

Микроконтролерите PIC16(L)F18855/75 имат схема за измерване на температурата на силициевия чип в диапазона между -40°C и +85°C, като изходното напрежение е пропорционално на температура на устройството. Изходът на схемата е вътрешно свързан към АЦП.

Схемата може да се използва за детектиране на температурен праг или поточен температурен индикатор, в зависимост от нивото на калибриране. Едноточково калибриране позволява на схемата да измерва температура, близка до тази точка. Двуточково калибрирането позволява на веригата да измерва поточно в целия температурен диапазон.

На фиг. 15.1 е показана в опростен вид схемата за измерване на температура. Получаване на пропорционално напрежение се постига чрез измерване на пада на напрежение върху няколко полупроводникови прехода.



Фиг. 15.1.

Изходната характеристика на температурния индикатор се изразява в уравнения 15.1 и 15.2.

- за високия диапазон:

$$V_{OUT}=V_{DD}-4V_T \quad (15.1)$$

- за ниския диапазон:

$$V_{OUT}=V_{DD}-2V_T \quad (15.2)$$

Схемата за измерване на температура е интегрирана с блока FVR. Работата ѝ се разрешава чрез установяване в 1 на бит TSEN в регистър FVRCON. Когато е забранена, не консумира ток.

Тя може да работи във високия и в ниския обхват, които се избират чрез бит TSRNG в регистър FVRCON. Когато той е установен в 1, осигурява по-широк диапазон на напрежението, което води до по-висока разрешаваща способност, но може да се различава при различните чипове. Този обхват изисква по-високо начално напрежение, следователно и по-високо V_{DD}.

Ниският диапазон се избира чрез нулиране на бит TSRNG в регистър FVRCON. Той генерира по-малък спад на напрежение, като по такъв начин е необходимо по-ниско напрежение за задействане на схемата. Ниският диапазон е предвиден за работа при ниски напрежения.

Когато температурният индикаторен блок работи в ниския диапазон, МК може да работи при всякакво работно напрежение, което отговаря на спецификациите. Когато работи във високия диапазон, Захранващото напрежение на МК V_{DD} трябва да е достатъчно високо, за да осигури необходимото отместване на напрежението, за да работи правилно температурната измерваща схема. Препоръчителните минимални стойности на V_{DD} за високия и ниския диапазон на измерване са съответно 3,6 V и 1,8 V.

Изходният сигнал от схемата се измерва с използване на вградения блок АЦП. За целта е резервиран един канал на АЦП.

За да се гарантира точно измерване на температурата, потребителят трябва да изчака поне 200 μs след като входният мултиплексор на АЦП е свързан с температурната измервателна схема, преди да се извърши преобразуването. Освен това трябва да се осигури времезакъснение от 200 μs между две последователни преобразувания.

16. Блок за откриване преминаване през нулата на променливотоков сигнал

Блокът за откриване на преминаване през нулата (Zero-cross detection – ZCD) детектира моментите на преминаване през нулевия потенциал на променливотоков сигнал. В действителност прагът на напрежение, който се открива - VCPINV, е 0.75V.

Към източника на входен сигнал е необходимо да се свърже последователно токоограничаващ резистор, чийто импеданс и мощност зависят от пиковите стойности на напрежението.

Блокът осигурява ток, протичащ към или от ZCD извода, за да поддържа постоянно напрежение на него, като по такъв начин го предпазва от право напрежение върху диодите за защита от електростатичен разряд. Когато приложеното напрежение е по-голяма опорното, блокът консумира ток, в обратния случай блокът генерира ток. Тези процеси поддържат напрежението на извода постоянно за целия диапазон на приложеното напрежение. Опростената блокова схема на ZCD блока е показана на фиг. 16.1.

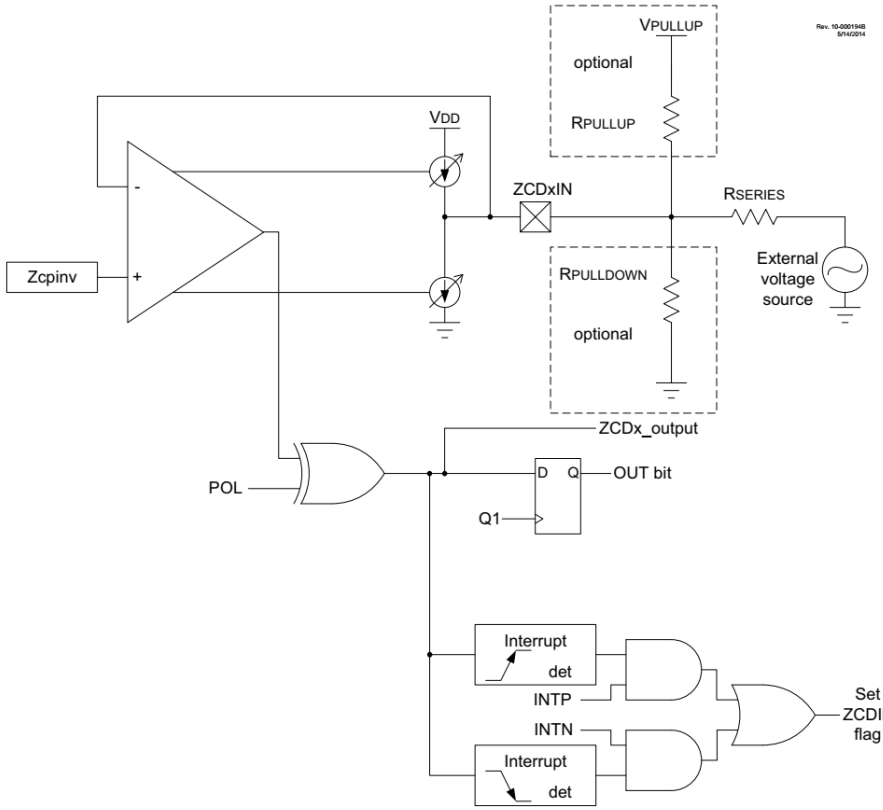
ZCD блокът може да се използва при мониторинг на променливотокови сигнали, например за измерване на период на сигнали, точно измерване на продължителни времеви периоди, фазово управление, превключвания с намаляване на електомагнитните смущения и др.

ZCD включва бит за състояние – бит OUT в регистър ZCDxCON, който индицира, дали в дадения момент блокът консумира или генерира ток. Значението на този бит може да се инвертира чрез бит POL в същия регистър.

Блокът може също да е източник на прекъсване при промяна на изхода на логиката, ако то е разрешено чрез съответните маски – локалната маста ZCDIE в регистър PIE2, бит INTP (за откриване на преден фронт) или бит INTN (за откриване на заден фронт) в регистър ZCDxCON, както и битове PEIE и GIE в регистър INTCON. За целта има детектори на преден и заден фронт на сигнала. Бит ZCDIF в регистър PIR2 ще се установи в 1, когато сработи кой да е детектор и свързаният с него разрешаващ бит е установен в 1.

Въпроси за самоконтрол към раздел V

1. Опишете предназначението на основните компоненти и работата на конфигурируема логическа клетка.
2. За какви логически функции може да бъде конфигурирана?
3. При кои ситуации CLC може да генерира прекъсване?
4. Какво е предназначението на температурния индикаторен блок, как работи и в какъв температурен диапазон?
5. Опишете предназначението и принципа на работа на блока за откриване на преминаване през нулата на променливотоков сигнал.



Фиг. 16.1. Блокова схема на блок ZCD

Раздел VI. Специални характеристики на МК PIC16(L)F18855/75

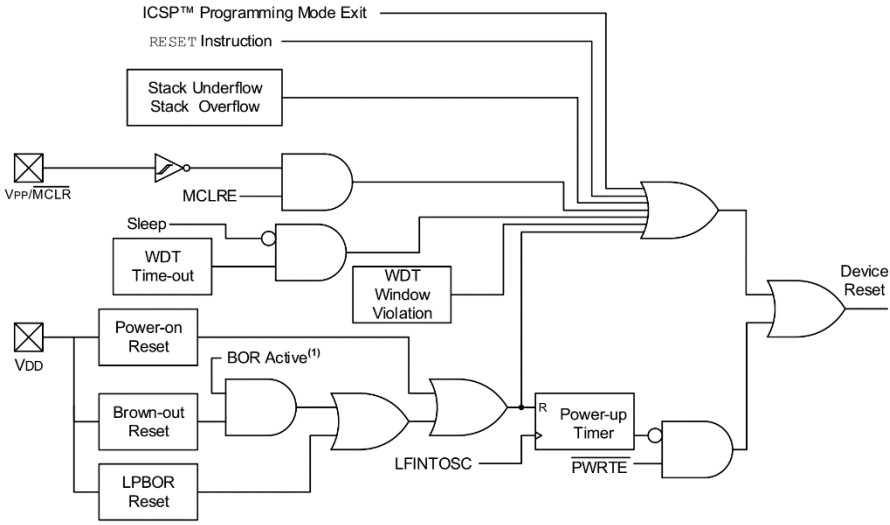
17. Причини и условия за начално установяване на микроконтролера

Има няколко начина за установяване в начално състояние на микроконтролера:

- При включване на захранването - Power-On Reset (POR);
- При смущения в захранването - Brown-Out Reset (BOR);
- Главно нулиране от вход $\overline{\text{MCLR}}$;
- От следящия таймер WDT;
- С инструкция RESET;
- При препълване на стека отгоре;
- При препълване на стека отдолу;
- При изход от режим на програмиране.

За да се позволи на захранващото напрежение VDD да се стабилизира, може да се използва опционен таймер при включване на захранването (Power-up timer). Той може да се разреши, за да увеличи времето след НУ след събитие BOR или POR.

Опростена блокова схема на блока за НУ е показана на фиг. 17.1.



Фиг. 17.1. Опростена блокова схема на блока за НУ

Начално установяване при включване на захранването (POR)

Схемата POR поддържа устройството в състояние на начално установяване, докато V_{DD} достигне приемливо ниво за функциониране на МК. Бавно нарастващото V_{DD}, високите работни честоти или работата на аналогови блокове може да изискват по-високо от минималното V_{DD}. Функциите PWRT, BOR или $\overline{MCLR}$ могат да се използват за удължаване на периода на стартиране, докато не бъдат изпълнени всички условия за работа на устройството.

Начално установяване при смущения в захранването (BOR)

Схемата BOR поддържа устройството в състояние на начално установяване, докато V_{DD} не достигне избираемо минимално ниво. Между POR и BOR може да се реализира пълно покритие на обхвата на напрежението за защита на работата на процесора.

Блок BOR има четири режима на работа, управлявани чрез битове BOREN <1:0> конфигурационните регистри. Четирите режима на работа са:

- BOR е винаги включен;
- BOR е изключва при Sleep;
- BOR се управлява програмно;
- BOR е винаги изключен.

Нивото на напрежението за BOR се избира чрез конфигуриране на бит BOR в конфигурационните регистри.

Осигурено е филтриране на шума за V_{DD}, което предотвратява задействането на BOR при малки смущения. Ако V_{DD} падне под V_{BOR} за продължителност, по-голяма от параметъра T_{BORDC}, устройството ще се ресетира (фиг. 17.2).

- **BOR е винаги включен**

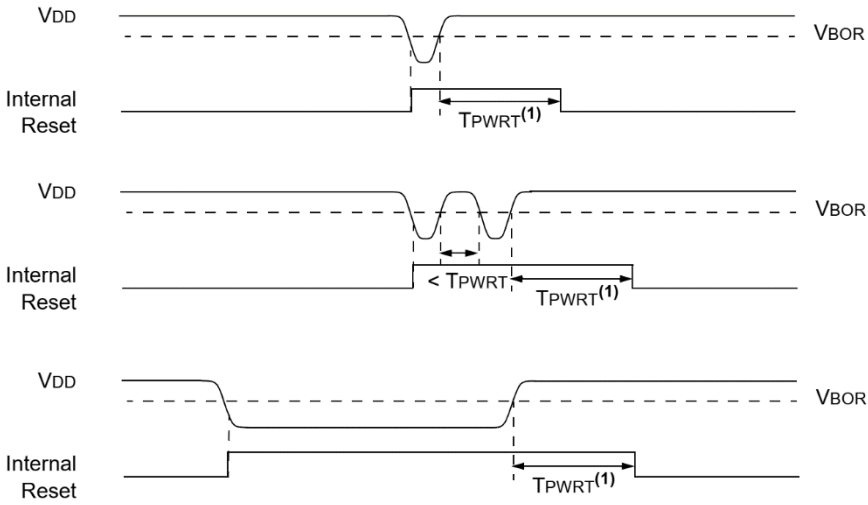
Режимът е избран, когато битове BOREN в конфигурационните регистри са програмирани като „11“. Стартирането на микроконтролера ще се забави, докато блок BOR не е готов и V_{DD} не стане по-високо от праговото напрежение за BOR.

Защитата от смущения в захранването е активна по време на режим Sleep. BOR не забавя събуждането от режим Sleep.

- **BOR е изключен при Sleep**

Когато битове BOREN в конфигурационните регистри са програмирани с „10“, BOR е включен, освен в режим Sleep. Стартирането на устройството ще се забави, докато BOR не е готов и V_{DD} не стане по-високо от праговото напрежение на BOR.

Защитата от смущения в захранването е активна по време на режим Sleep. Събуждането на устройството ще се забави, докато BOR не е готов.



(1) Само ако бит PWRTE е нулиран.

Фиг. 17.2.

- **BOR се управлява програмно**

Когато битове BOREN в конфигурационните регистри са програмирани с „01“, BOR се управлява от бит SBOREN в регистър BORCON. Стартирането на МК не се забавя от условието за готовност на BOR или нивото на V_{DD}.

Защитата от смущения в захранването се включва веднага, след като блок BOR е готов. Състоянието на BOR схемата се отразява в бит BORRDY в регистър BORCON.

BOR не се променя от режима Sleep.

- **BOR е винаги изключен**

Когато битове BOREN в конфигурационните регистри са програмирани като „00“, BOR е изключен по всяко време. Стартирането на устройството не се забавя от условието за готовност на BOR или нивото на V_{DD}.

Начално установяване от вход MCLR

$\overline{\text{MCLR}}$ е незадължителен за използване външен вход, който може да установи МК в начално състояние. Функцията $\overline{\text{MCLR}}$ се управлява от битове MCLRE и LVP в конфигурационните регистри.

Когато $\overline{\text{MCLR}}$ е разрешен и изводът е задържан в ниско ниво, МК се държи в режим на начално установяване. Извод $\overline{\text{MCLR}}$ е свързан към V_{DD} чрез вграден резистор, изтеглящ към захранване.

По веригата за НУ чрез $\overline{\text{MCLR}}$ има вграден филтър, чието предназначение е да открива и игнорира малки импулси.

Когато $\overline{\text{MCLR}}$ е забранен, изводът функционира като вход с общо предназначение и с вградено слабо изтегляне към захранване под програмно управление.

Начално установяване от следящия таймер (WWDT)

Следящият таймер предизвиква начално установяване на МК, ако във процеса на изпълнение на програмата не се срещне инструкция CLRWDT в рамките на периода на таймаут и времевият прозорец остане отворен. Битове $\overline{\text{TO}}$ и $\overline{\text{PD}}$ в регистър STATUS и бит $\overline{\text{WDT}}$ в PCON се променят, за да индицират НУ от WDT, причинено от препълване на таймера, а бит WDTWV в регистър PCON се променя, за да индицира нулиране на WDT, причинено от нарушение на времевия прозорец.

Начално установяване с инструкция RESET

Използването на инструкция RESET ще предизвика НУ на МК, при което бит $\overline{\text{RI}}$ в регистър PCON ще се нулира.

Начално установяване при препълване на стека отгоре/отдолу

При препълване на стека се установяват в 1 битове STKOVF или STKUNF в регистър PCON, в зависимост от вида на препълването. Тези НУ се разрешават чрез установяване в 1 на бит TVREN в конфигурационните регистри.

Начално установяване при изход от режим на програмиране

При излизане от режима на вътрешно-схемно серийно програмиране (ICSP), МК ще се държи така, както когато е настъпило POR (МК не се рестира по време на операции за самопрограмиране/изтриване).

Таймер при включване на захранването

Ако бъда включен, таймерът при включване на захранването забавя стартиране на работата на МК след условие за BOR или POR. Той се използва обикновено, за да позволи на V_{DD} да се стабилизира, преди МК да започне да работи.

Таймерът $\overline{\text{PWRTE}}$ осигурява номинално време от 64 ms при POR или BOR. МК се поддържа начално състояние, докато той е активен. Закъснението, което той осигурява, позволява допълнително време, през което V_{DD} да се повиши до приемливо ниво. Таймерът се разрешава чрез нулиране на бит $\overline{\text{PWRTE}}$ в конфигурационните регистри. Той се стартира след POR и BOR.

Стартова последователност при начално установяване

При POR или BOR се случва следното, преди МК да започне работа:

1. Включва се PWRT до завършване на времезакъснението (ако е разрешен).

2. Таймерът за стартиране на осцилатора сработва до завършване на времезакъснението (ако е необходимо при наличие на кварцов резонатор).

3. $\overline{\text{MCLR}}$ трябва да бъде отпуснат (ако е разрешен).

Общото време на изчакване ще варира в зависимост от конфигурацията на тактовия генератор и таймера PWRT.

Таймерите при включване на захранването и при стартиране на осцилатора работят независимо от НУ от $\overline{\text{MCLR}}$. Ако $\overline{\text{MCLR}}$ се поддържа в ниско ниво достатъчно време, времезакъсненията от PWRT и таймера за стартиране на осцилатора ще изтекат. При установяване на $\overline{\text{MCLR}}$ във високо ниво, МК ще започне работа след 10 такта на F_{OSC} (фиг. 17.3). Това е полезно за целите на тестване или за синхронизиране на повече от едно устройство, работещи паралелно.

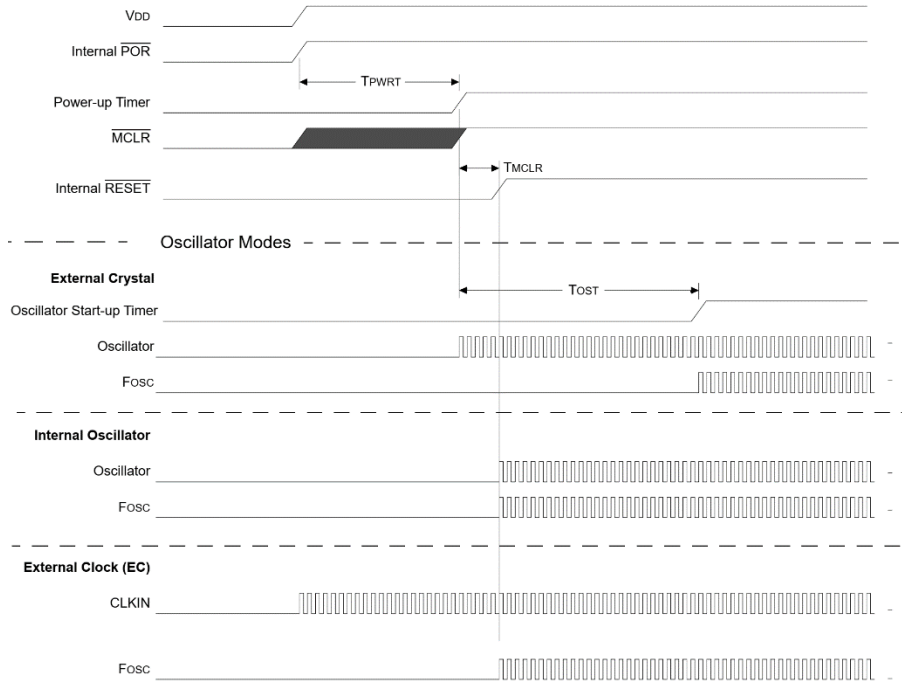
Откриване на причината за НУ

При всяко НУ се актуализират множество битове в регистър STATUS и PCON, за да посочат причината за НУ.

Регистър за управление на захранването

Регистърът за управление на захранването (PCON) съдържа флагове, с цел разграничаване на следните причини за НУ: включване на захранването (POR), смущения в захранването (BOR), инструкция Reset (RI), от вход $\overline{\text{MCLR}}$ (RMCLR), таймаут на WDT (RWDT), нарушение на времевия прозорец на следящия таймер (WDTWV), препълване на стека, съответно отдолу (STKUNF) и отгоре (STKOVF).

Състоянието на съответните битове в регистъра ще се промени по апаратен път по време на процеса на НУ, ако то е причинено от съответното условие. Те обаче трябва да се установят в начално състояние по програмен път. Битовете в регистър PCON могат да се установяват също по програмен път, така че потребителският код да може да бъде тестван, но няма да се предизвика начално установяване на МК.

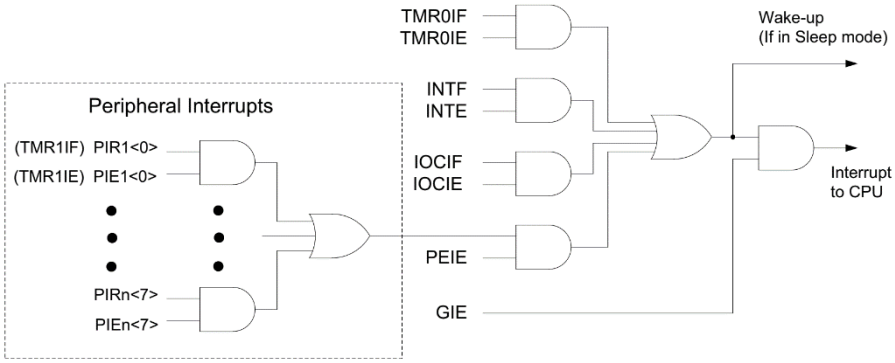

VDD
Internal POR
TPWRT
Power-up Timer
MCLR
TMCLR
Internal RESET
Oscillator Modes
External Crystal
TOST
Oscillator Start-up Timer
Oscillator
FOSC
Internal Oscillator
Oscillator
FOSC
External Clock (EC)
CLKIN
FOSC

Фиг. 17.3.

18. Прекъсвания

Функцията за прекъсване позволява на определени събития да изпреварят нормалния ход на програмата. Фърмуерът се използва за определяне на източника на прекъсване и действие съгласно него. Някои прекъсвания могат да бъдат конфигурирани, за да събудят MCU от режим Sleep. Много от периферните блокове могат да бъдат източници на прекъсвания.

Логическата схема за обслужване на прекъсванията при МК PIC16(L)F18855/75 е показана на фиг. 18.1.



Фиг. 18.1. Логическа схема за обслужване на прекъсванията при PIC16(L)F18855/75

Прекъсванията са забранени при всяко НУ. Те се разрешават чрез установяване в 1 на следните битове:

- Бит GIE в регистър INTCON;
- Индивидуалните разрешаващи битове (маски) за отделните събития, водещи до прекъсване;
- Бит PEIE в регистър INTCON (ако индивидуалната маска се намира в някой от регистри PIEx);

Регистри PIR1, PIR2, PIR3 и PIR4 регистрират индивидуалните прекъсвания чрез флаговете за прекъсвания. Флаговете се установяват в 1, независимо от състоянието на битове GIE, PEIE и индивидуалните маски.

Когато възникне прекъсване и бит GIE е установен в 1, се случва следното:

- Текущата извлечена инструкция се игнорира;
- Бит GIE се нулира;
- Текущото съдържание на програмния брояч PC се записва в стека;

- Критичните регистри автоматично се съхраняват в сенчестите регистри;
- PC се зарежда с вектора на прекъсване 0004h .

Апаратният механизъм за обслужване на прекъсванията се състои в следното:

Подпрограмата за обслужване на прекъсванията (ПОП, Interrupt Service Routine - ISR) трябва да определи източника на прекъсване чрез последователна проверка на флаговете. Флаговете трябва да се нулират преди излизане от ISR, за да се предотвратят повтарящи се прекъсвания. Тъй като бит GIE е нулиран, всяко прекъсване, което възникне по време на изпълнението на ISR ще бъде регистрирано чрез флага си, но процесорът няма да започне да го обслужва. Индивидуалните флагове за прекъсвания се установяват в 1, независимо от състоянието на разрешаващите битове (маски). Всички прекъсвания ще бъдат игнорирани, докато битът GIE е нулиран. Всяко прекъсване, възникнало, докато битът на GIE е 0, ще бъде обслужено, когато бит той бъде установен в 1 отново.

В случай, че обслужването на дадено прекъсвания е разрешено и възникне заявка за такова (тоест настъпило е определено събитие и флагът се е вдигнал), след завършване на изпълнението на текущата инструкция по апаратен път се извършват няколко действия:

- нулира се битът *GIE*, с което се забранява други прекъсвания да прекъсват текущото;
- адресът на следващата след нея инструкция се записва в стека;
- изпълнява се преход към вектора за прекъсване. Тоест програмният брояч PC се зарежда с адрес 0004h, където трябва да се намира първата инструкция от подпрограмата за обслужване на прекъсванията (ПОП).

ISR от своя страна, трябва да завършва със специална инструкция RETFIE за връщане от прекъсване, изпълнението на която предизвиква излизане от ISR чрез извличане на предишния адрес от стека, възстановява записания контекст от сенчестите регистри и установява в 1 бит GIE.

Закъснение при обслужване на прекъсване

Латентността на прекъсването се дефинира като времето от момента на възникване на събитието на прекъсване до началото на изпълнението на кода от вектора на прекъсването. Латентността за синхронни прекъсвания е три или четири цикъла на инструкция. За асинхронни прекъсвания латентността е от три до пет цикъла на инструкция, в зависимост от това кога настъпва прекъсването.

Прекъсванията по време на режим Sleep

Някои прекъсвания могат да се използват за събуждане от режим Sleep. За целта периферното устройство трябва да може да работи без системния такт. Съответните разрешаващи прекъсването битове трябва да са установени в 1, преди МК да влезе в режим Sleep.

При събуждане от режим Sleep, ако битът GIE също е установен в 1, процесорът ще се разклони към вектора на прекъсването. В противен случай процесорът ще продължи да изпълнява инструкциите след инструкция SLEEP. Инструкцията, която е веднага след инструкцията SLEEP, винаги ще се изпълнява преди разклонение към ISR.

Вход за прекъсвания от външни източници INT

Изводът INT може да се използва за генериране на асинхронно прекъсване, задействано по фронт. Това прекъсване се разрешава чрез установяване в 1 на бит INTE в регистър PIE0. Бит INTEDG в регистър INTCON определя по кой фронт ще настъпи прекъсването. Когато бит INTEDG е установен в 1, ще се регистрира прекъсване по преден фронт. Когато бит INTEDG е нула, ще възникне прекъсване по заден фронт. Бит INTF в регистър PIR0 ще се установи в 1, когато на извод INT се появи зададен фронт. Ако битове GIE и INTE също са установен в 1, процесорът ще пренасочи изпълнението на програмата към вектора на прекъсването.

Автоматично съхраняване на контекста при прекъсване

При влизане в прекъсване адресът на връщане (стойността на PC) се записва в стека. Освен него, следните регистри също се записват автоматично в сенчестите регистри: W, STATUS (освен битове $\overline{TO}$ и $\overline{PD}$), BSR, FSR и PCLATH.

При излизане от ISR тези регистри се възстановяват автоматично. Всички промени в тези регистри по време на ISR ще бъдат загубени. Ако е необходимо промените в някои от тях да се запазят, съответният сенчест регистър трябва да бъде променен и стойността ще бъде възстановена при излизане от ISR. Регистрите в сянка се намират в банка 31 и са достъпни за четене и запис. В зависимост от приложението на потребителя може да се наложи да бъдат запазени и други регистри.

Програмни аспекти на обслужването на прекъсванията

Няколко неща трябва да се имат предвид при писането на програми, в които ще се обслужват прекъсвания (каквито са в повечето случаи програмите за обслужване на реални устройства):

- В повечето случаи (с някои изключения) при обслужването на прекъсванията флаговете за заявки **не се нулират по апаратен път** (автоматично) - това трябва да се направи програмно;

В противен случай може да се стигне до рекурсивно обслужване на фалшиви заявки.

- Обслужване на **повече от един източници на прекъсвания**:

При микроконтролерите от фамилия PIC16 векторът за прекъсване е само един. Това означава, че в общия случай множество прекъсвания се обслужват от една и съща подпрограма. Ето защо тя трябва да започва с разпознаване на източника, което се прави като се проверяват програмно един след друг флаговете за заявки на очакваните прекъсвания. Това става с инструкции за условни преходи (*btfss* и *btfsc*), при което се изпълнява евентуално преход към участък от ISR, в който се обслужва съответното прекъсване. Участъкът завършва с инструкция за връщане от ISR. По такъв начин ISR представлява разклонена програма с една входна и множество изходни точки. Във всеки подобен участък, както бе споменато по-горе, не трябва да се пропуска (освен в редки изключения) инструкция, която да нулира флага за прекъсване.

- **Приоритет на прекъсванията**

При PIC16 **няма апаратен механизъм** за задаване на приорите на прекъсванията, ако се налага такъв. Приоритетът на множество прекъсвания се определя по програмен път чрез последователността на тестване на флаговете в началото на ISR. Ясно е, че при едновременното възникване например на две заявки, ще се премине към обслужването на първата разпозната, като тя ще се окаже с по-висок приоритет.

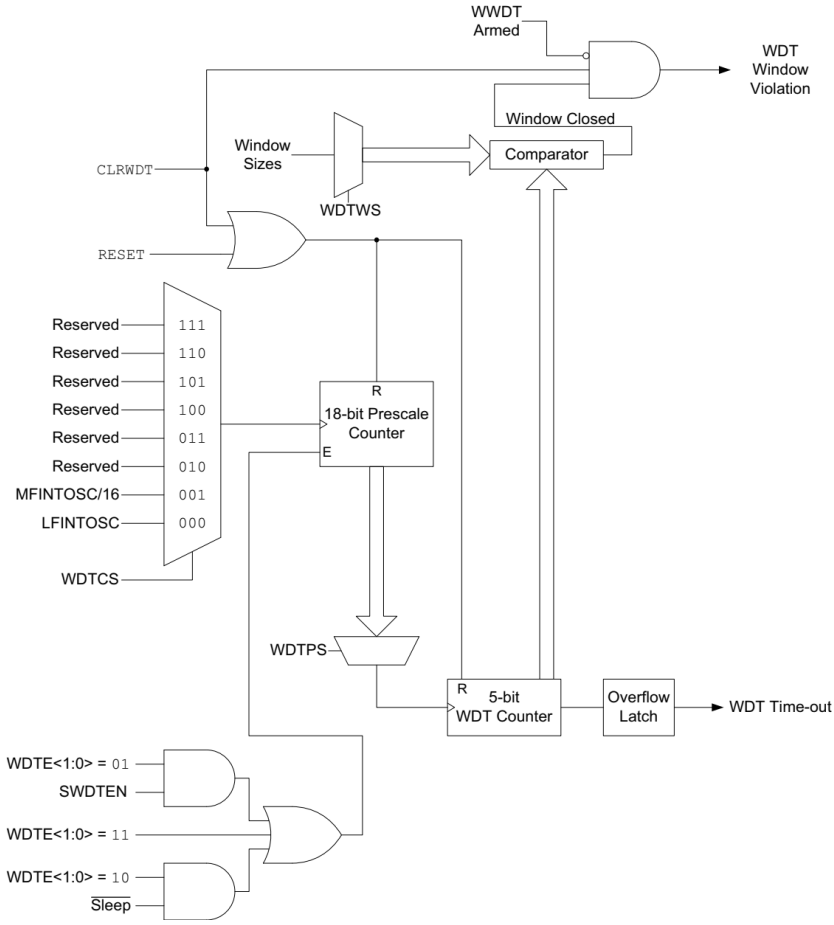
- **Влагане на прекъсвания**

Да си спомним, че при преход към ISR бит **GIE автоматично се нулира**, забранявайки обслужването на следващи прекъсвания. Ако желаем едно прекъсване да прекъсва друго (обикновено с по-нисък приоритет), е необходимо в съответния участък да установим бит GIE в единица.

19. Прозоречен следящ таймер

Следящият таймер (Watchdog Timer - WDT) представлява системен таймер, който генерира условие за НУ, в случай че в процеса на изпълнение на програмата не се срещне инструкция CLRWDT по време на периода на таймаут на брояча му. Предназначението му е да спомогне системата да се възстанови (да възобнови работата си нормално) след неочаквани събития. Прозоречният следящ таймер (Windowed Watchdog Timer - WWDT) се различава по това, че инструкции CLRWDT се възприемат само, ако се изпълняват по времето на специален времеви интервал (прозорец) в периода на таймаут на таймера.

Блоковата схема на следящия таймер е показана на фиг. 19.1.



Фиг. 19.1 Блокова схема на блок WWDT

Блокът WWDT има следните характеристики:

- Избираем източник на тактов сигнал;
- Няколко работни режима: WDT е винаги включен; изключен в режим Sleep; управлява се по програмен път; WDT е винаги изключен;
- Конфигурируем период на таймаут от 1 ms до 256;
- Конфигурируем времеви прозорец от 12.5 до 100% от периода на таймаут;
- Няколко условия за НУ;
- Възможност за работа по време на режим Sleep.

Тактовият сигнал за WDT може да бъде вътрешна: LFINTOSC с честота 31 kHz или MFINTOSC/16 с честота 31.25 kHz, в зависимост от съдържанието, или на битове WDTCCS<2:0>, или на WDTCS<2:0> в регистър WDTCON1.

Периодът на таймаут на WDT се задава чрез битове WDTPS в регистър WDTCON0 и може да бъде в диапазона от 1 ms до 256 sec (номинална стойност). След НУ подразбиращото се време е 2 sec.

Режими на работа на WDT

Режимите на работа на следящия таймер се задават чрез битове WDTE<1:0> в конфигурационните регистри.

- **WDT е винаги включен**

Задава се чрез установяване на битове WDTE в „11“. Защитата чрез WDT е активна по време на режим Sleep.

- **WDT е изключен в режим Sleep**

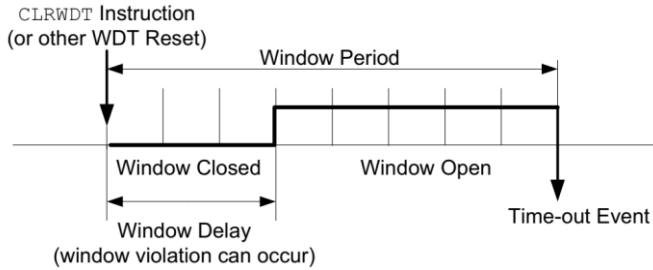
За целта битове WDTE се установяват в „10“. Тогава WDT е включен, с изключение на режим Sleep.

- **WDT се управлява програмно**

Когато битове WDTE се установят в „01“, WDT се управлява чрез бит SEN в регистър WDTCON0. Защитата чрез WDT остава непроменена по време на режим Sleep.

Времеви прозорец на следящия таймер

Следящият таймер има възможност да работи в прозоречен режим, който се управлява от конфигурационни битове WDTCWS<2:0> битове WINDOW<2:0> в регистър WDTCON1. В този режим инструкция CLRWDT трябва да се изпълни в рамките на разрешения времеви прозорец за периода на WDT. Ако инструкция CLRWDT се срещне извън този прозорец, това ще е събитие, причина за НУ от следящия таймер, подобно на случа при таймаут (фиг. 19.2).



Фиг. 19.2.

Продължителността на прозореца се задава с конфигурационни битове WDTCWS<2:0> или с битове WINDOW<2:0> в регистър WDTCON1, ако WDTCWS<2:0> = 111.

В случай на нарушаване на прозореца, ще възникне НУ и бит $\overline{\text{WDTWV}}$ в регистър PCON ще се нулира. Той се установява в 1 при POR или мое да се установи по програмен път.

WDT се нулира при кое да е от следните събития: НУ; изпълнение на валидна инструкция CLRWDT; влизане в режим Sleep на МК; забрана на WDT; таймерът за старт на осцилатора (OST) работи; запис в регистър WDTCON0 или WDTCON1.

Когато е в прозоречен режим, WDT трябва да бъде активиран, преди инструкцията CLRWDT да нулира таймера. Това става чрез четене на регистър WDTCON0. Изпълнението ѝ, без той да е активиран, ще предизвика нарушаване на времевия прозорец.

Работа на WDT по време на режим Sleep

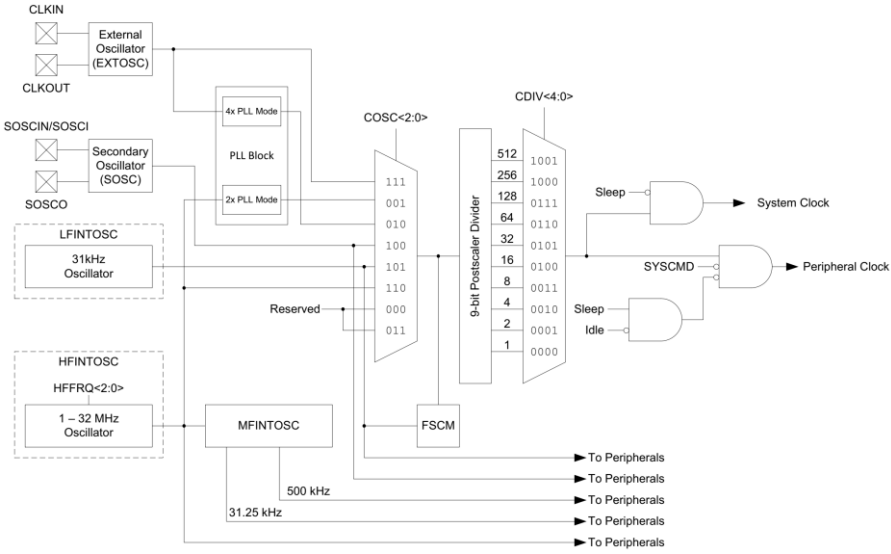
Когато МК влезе в режим Sleep, WDT се нулира. Ако той е разрешен по време на Sleep, продължава да брои. Когато МК излезе от режим Sleep, WDT отново се нулира.

WDT остава нулиран, докато OST, ако е разрешен, приключи времезакъснението си.

Ако настъпи таймаут на WDT, докато МК е в режим Sleep, няма да настъпи НУ. Вместо това МК ще се събуди и ще възобнови работата си. Битове $\overline{\text{TO}}$ и $\overline{\text{PD}}$ в регистър STATUS променят състоянието си, за да индицират това. Бит $\overline{\text{RWDT}}$ в регистър PCON също може да се използва.

20. Блок генератор на тактови сигнали

Генераторният блок предоставя голямо разнообразие от източници на тактови сигнали и различни функции за избор, които му позволяват да се използва в широк спектър от приложения, като същевременно максимизира производителността и намалява консумацията на енергия. Неговата блокова схема е показана на фиг. 20.1.



Фиг. 20.1. Блокова схема на генераторен блок за тактови сигнали

Източниците на тактови сигнали могат да бъдат външни тактови генератори, кварцови и керамични резонатори. Освен това източник системен тактов сигнал може да бъде и един от двата вградени генератора и PLL блока, като е възможен програмен избор на честоти. Допълнителните функции на тактовия генератор (ТГ) включват:

- Избираем *външен или вътрешен източник* на системна тактова честота по програмен път.
- *Възел Fail-Safe Clock Monitor* (FSCM) за откриване на сривове във външни източници на тактови сигнали (LP, XT, HS, ECH, ECM, ECL) с възможност за автоматично превключване към вграден генератор.
- *Резонаторен стартов таймер* (Oscillator Start-up Timer - OST), който осигурява стабилност на честотата от източници с кварцови резонатори

От всички тези източници на тактови сигнали може да се получи голямо разнообразие от тактови сигнали с различни честоти.

Режими на външен и вграден тактов генератор

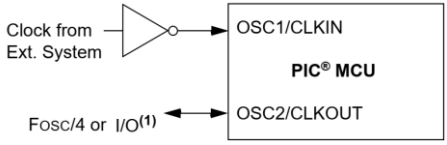
Режимите на външен и вграден източник на тактов сигнал се избират чрез битове RSTOSC<2:0> в регистър CONFIG1.

Ако се избира *външен източник на тактов сигнал*, заедно с битове RSTOSC, трябва да се и използват битове FEXTOSC в същия регистър.

Друг начин е чрез програмиране на битове NOSC<2:0> и NDIV<4:0> в регистър OSCCON1.

Различните подрежими на *вграден ТГ* се избират чрез чрез битове RSTOSC<2:0>. Когато се налага системният такт да се превключи към вътрешен източник на тактов сигнал по време на нормална работа, това става чрез битове NOSC<2:0> в регистър OSCCON1.

Външният генераторен блок може да се конфигурира за един от следните режими чрез битове FEXTOSC<2:0> в регистър CONFIG1 (фиг. 20.2):



Фиг. 20.2. Работа с външен тактов генератор – режим EC

1. *ECL (External Clock Low-Power)* – с ниска консумация (честоти под 500 kHz);

2. *ECM (External Clock Medium Power)* – със средна консумация (500 kHz до 8 MHz);

3. *ECH (External Clock High-Power)* – с висока консумация (честоти над 8 MHz);

4. *LP (Low-Power Crystal)* – с ниска консумация с кварцов резонатор (32 kHz);

5. *XT (Medium Gain Crystal or Ceramic Resonator Oscillator)* - със средна консумация с кварцов или керамичен резонатор (100 kHz до 4 MHz);

6. *HS (High Gain Crystal or Ceramic Resonator)* - с висока консумация с кварцов или керамичен резонатор (честоти над 4 MHz).

При режимите ECH, ECM и ECL се използват сигнали от външна логика като тактови. При режимите LP, XT и HS е необходимо към МК да се свърже външен кварцов или керамичен резонатор. Всеки режим е оптимизиран за различен честотен диапазон.

Външният генераторен блок може също да се използва с блок PLL.

Вграденият генераторен блок INTOSC включва два възела и може също да се използва с PLL (Phase Lock Loop) блока. Той може да генерира тактови сигнали с ниска и висока честота: LFINTOSC (Low-Frequency Internal

Oscillator) за честота 31 kHz и HFINTOSC (High-Frequency Internal Oscillator) за честоти в диапазона от 1 до 32 MHz. И двата са фабрично калибрирани.

В допълнение на споменатите два вградени честотни генератора, тактовият генератор съдържа и **делителен блок**, наречен MFINTOSC, чието предназначение е да осигури няколко специални честоти за други блокове в МК. Възелът MFINTOSC от входния сигнал HFINTOSC генерира два тактови сигнала – с честота 500 kHz (MFINTOSC) и 31.25 kHz (MFINTOSC/16).

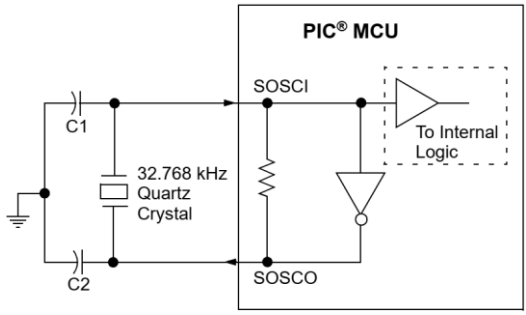
Периферни блокове, които използват тактова честота от MFINTOSC 500 kHz, са TMR1, TMR3, TMR5, SMT1, SMT2 и CLKREF. Тези, които използват тактова честота 31.25 kHz, са WDT, TMR2, TMR4, TMR6, SMT1, SMT2 и CLKREF.

Резонаторен стартов таймер

Ако генераторният блок е конфигуриран за режими LP, XT или HS, се използва специален таймер - Oscillator Start-up Timer (OST), който отброява 1024 такта на входа OSC1. Това вемезакъснение следва след НУ от включване на захранването или след излизане от режим Sleep. Таймерът OST осигурява времезакъснение, което гарантира, че генераторната схема, която използва кварцов или керамичен резонатор, е стартирала генерациите и осигурява стабилна тактова честота за генераторния блок.

Вторичен честотен генератор

Вторичният честотен генератор е отделен блок, който може да се използва за алтернативен източник на тактов сигнал. Той е оптимизиран за честота 31 kHz и може да се използва с външен кварцов резонатор, свързан към изводи SOSCI и SOSCO (фиг. 20.3) или с външен честотен генератор, свързан към извод SOSCIN.



Фиг. 20.3. Вторичен честотен генератор с кварцов резонатор

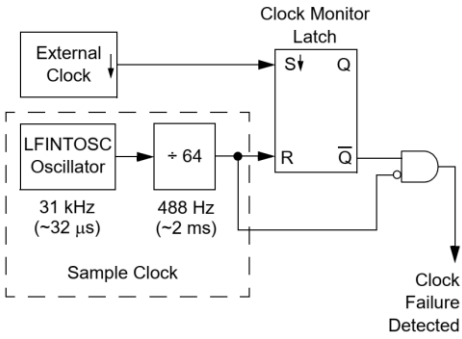
Превключване на източниците на тактов сигнал

Източникът на тактов сигнал може да се превключва между външен и вътрешен по програмен път чрез битове NOSC и NDIV в регистър OSCCON1.

Мониторинг за смущения в тактовия сигнал

Възелът FSCM позволява на МК да продължи работата си в случай, че отпадне тактовия сигнал от външен източник. FSCM се разрешава чрез бит

FCMEN в конфигурационните думи и е приложим за всички режими на външен източник на тактов сигнал: LP, XT, HS, EC, както и за вторичния честотен генератор.



Фиг. 20.4. Блокова схема на блок FSCM

Възелът FSCM открива отпадане на сигнала чрез сравняване на външната честота с опорна тактова честота в FSCM. Последната се генерира чрез разделяне на LFINTOSC на 64 (фиг. 20.4). Възелът съдържа тригер, който се установява на всеки заден фронт на външния тактов сигнал. Опорният тактов сигнал нулира тригера на всеки преден фронт. Отпадане/ сущение на външния такт се регистрира, когато изтече половин период на опорния сигнал, преди външният сигнал да премине в ниско ниво.

21. Енергоспестяващи режими

Енергоспестяващите режими са удобни за редица приложения (особено при такива с батерийно захранване), при които не е необходимо микроконтролерът (или всички негови блокове) да работи постоянно, а се изчаква настъпването на някакво събитие.

При МК PIC16(L)F18855/7 има два режима, предназначение за намаляване на консумацията: режим DOZE и режим Sleep.

21.1. Режим DOZE

Режимът DOZE позволява икономия на енергия чрез ограничаване работата на процесора и достъпа до програмна памет (PFM), без да се засяга работата на периферните блокове. Той се различава от режим SLEEP, който ще разгледаме по-нататък, по това, че тактовия генератор продължава да работи.

Когато бит Doze Enable (DOZEN) е установен в 1, процесорът изпълнява само един цикъл на инструкция от всички N цикъла, като това се задава чрез битове DOZE <2:0> в регистър CPUDOZE. Например, ако DOZE <2:0> = 100, съотношението на цикъла на инструкции е 1:32. Процесорът и паметта изпълняват един цикъл на инструкция и след това се поставя в свободен режим за 31 цикъла на инструкции. По време на неизползваните цикли периферните устройства продължават да работят с честотата на системния такта.

Режим Doze е илюстриран на фиг. 21.1.

В примера на фиг. 21.1 бит DOZEN = 1, битове DOZE<2:0> = 001 (отношение 1:4), битът за възстановяване при прекъсване ROI = 1.

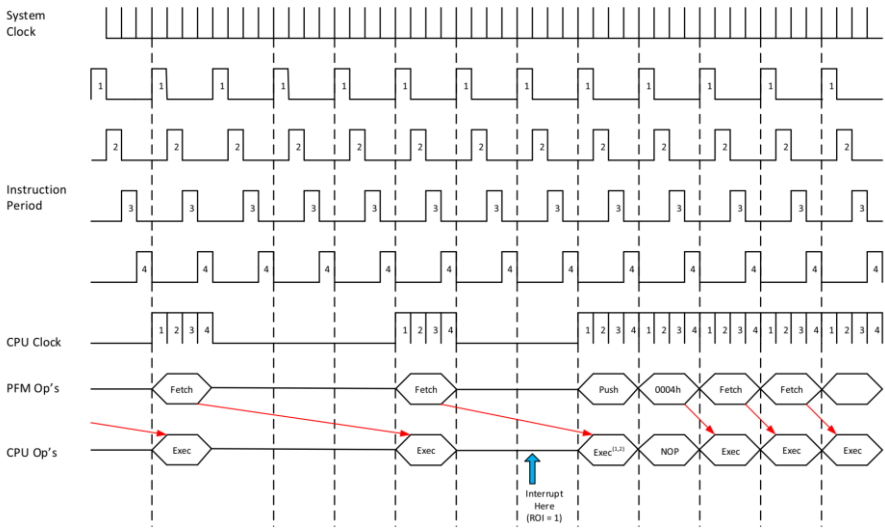
Както и по време на нормална работа, PFM извлича инструкция за следващия цикъл. Тактовите поредици Q продължават през цялото време.

Ако възникне прекъсване и в това време бит ROI = 0, подпрограмата за обслужването му ще продължи да се изпълнява с честота, избрана чрез битове DOZE<2:0>. Продължителността на прекъсването ще се увеличи до отношението, зададено в тях.

Ако по време на прекъсване бит ROI = 1, бит DOZEN се нулира и ЦП работи с пълна честота. Предварително извлечената инструкция се изпълнява, след което се изпълнява ПОП. На фиг. 22.1 прекъсването възниква по време на втория цикъл на инструкция на Doze периода, Като веднага извежда ЦП извън него. Ако бит DOE = 1, когато се изпълнява инструкция RETFIE, DOZEN се установява в 1, а ЦП работи с редуцирана честота, зависеща от състоянието на битове DOZE<2:0>.

21.2. Режим Sleep

МК влиза в режим Sleep при изпълнение на инструкция SLEEP, като едновременно с това бит IDLEN в регистър CPUDOZE трябва да е нулиран. Ако ин-



(1) Инструкциите от няколко цикъла се изпълняват до завършването им, преди да се извлече тази на адрес 0004h.
(2) Ако предварително извлечената инструкция нулира бит GIE, ISR няа да се изпълни, но DOZEN все още е нулиран и ЦП ще възобнови работата си на пълна честота.

Фиг. 21.1. Режим Doze

струкция SLEEP се изпълнява, докато той е установен в 1, ЦП ще влезе в режим IDLE.

Влизане в режим Sleep

При влизане в режим Sleep става следното:

1. WDT ще се нулира, но ще продължи да работи, ако е разрешен по време на Sleep.
2. Бит $\overline{PD}$ в регистър STATUS се нулира.
3. Бит $\overline{TO}$ в регистър STATUS се установява в 1.
4. Тактовият сигнал към ЦП се забранява.
5. 31 kHz LFINTOSC, HFINTOSC и SOSC остават незасегнати и периферните блокове, които ги използват могат да продължат да работят по време на Sleep.
6. Таймер1 и периферните блокове, които го използват, които го използват, продължават да работята по време на Sleep, когато за него е избран като източник на тактов сигнал LFINTOSC, T1CKI или вторичният генератор.
7. Работата на АЦП не се засяга, ако е избран FRC генератор.

8. I/O портове остават в състоянието, което са имали преди влизане в режим Sleep.

9. Условията за НУ, с изключение на WDT, не се влияят от режим Sleep.

За да се намали консумацията на ток, трябва да се имат съобразят следните условия:

- I/O изводи не трябва да са плаващи;
- Външни схеми, консумиращи ток от I/O изводи;
- Вътрешни схеми, генериращи ток към I/O изводи (например, блокове ЦАП и FVR);
- Утечки на изводи с включено слабо изтегляне към захранване;
- Блокове, които използват кой да е тип тактов генератор.

I/O изводи, конфигуриране като високо-импедансни входове, трябва да се свържат към V_{DD} или V_{SS} външно, за избягване на утечки от плаващи входове.

Излизане от режим Sleep

МК може да се изведе от режим Sleep посредством следните събития:

1. Външен ресет от вход $\overline{\text{MCLR}}$, ако е разрешен.
2. НУ от смущения в захранването (BOR), ако е разрешен.
3. НУ при включване на заханването (POR).
4. Следящият таймер, ако е разрешен.
5. Външни прекъсвания.
6. Прекъсвания от вградени блокове, които могат да работят по време на Sleep.

При изпълнението на инструкция SLEEP следващата инструкция (на адрес PC+1) се извлича предварително. За да се събуди микроконтролерът от режим SLEEP чрез прекъсване, съответният бит за разрешаването му трябва да е в единица. Събуждането се осъществява независимо от състоянието на глобалната маска *GIE*. Но от нея зависи как ще продължи изпълнението на програмата по-нататък:

- Ако *GIE=1*, се изпълнява инструкцията след SLEEP и тогава се извършва преход към вектора за прекъсване (адрес 0004H). Ако изпълнението на инструкцията след SLEEP не е желателно, то след нея трябва да се постави NOP.
- Ако *GIE=0* изпълнението на програмата продължава с инструкцията след SLEEP.

При излизане от Sleep WDT се нулира, независимо от причината за събуждане.

Излизане от режим Sleep с прекъсвания

Когато прекъсванията са забранени глобално (GIE=0), но има разрешени прекъсвания чрез локални маски и при това са възникнали заявки за прекъсвания (има вдигнати флагове) от такива, с изключение на това от превключване на източника на тактов сигнал, се случва следното:

- Ако прекъсването настъпи преди инструкция SLEEP:
  - Инструкция SLEEP ще се изпълни като NOP;
  - WDT и входният му делител няма да се нулират;
  - Бит $\overline{TO}$ в регистър STATUS няма да се установи в 1;
  - Бит $\overline{PD}$ в регистър STATUS няма да се нулира.
- Ако прекъсването настъпи по време или след изпълнението на инструкция SLEEP:
  - Инструкция SLEEP ще завърши напълно (ще се изпълни);
  - МК веднага ще се събуди от Sleep;
  - WDT и входният му делител ще се нулират;
  - Бит $\overline{TO}$ в регистър STATUS ще се установи в 1;
  - Бит $\overline{PD}$ в регистър STATUS ще се нулира.

Дори и ако сме проверили флаговете преди изпълнението на инструкция SLEEP, е възможно те да се установят в 1, преди изпълнението ѝ да завърши. За да се определи, дали се е изпълнила, трябва да се тества бит $\overline{PD}$. Ако той е установен в 1, инструкция SLEEP се е изпълнила като NOP.

Маломощен Sleep режим

МК PIC16F18855/75 имат вграден регулатор на напрежение с ниски загуби (Low Dropout - LDO), който позволява на I/O изводи да работят при напрежения до 5.5V, докато вътрешната логика работи при по-ниски стойности на напрежението. LDO и свързаната с него опорна схема трябва да останат активни, когато МК е в режим Sleep. Този подрежим се избира чрез бит VREGPM в регистър VREGCON. По такъв начин се дава възможност на потребителя да оптимизира работния ток по време на Sleep, в зависимост от изискванията на приложението.

При МК PIC16LF18855/75 няма подобен подрежим – те са винаги в режим на най-ниска консумация, тъй като имат по-ниски максимални стойности на VDD и I/O изводи.

По подразбиране LDO и референтната му схема остават в нормална конфигурация по време на Sleep. МК е в състояние да излезе от режим Sleep бързо, тъй като всички схеми са активни. В маломощен Sleep режим, при събуждане от Sleep, е необходимо допълнително времезакъснение за тези схеми, за да се върнат към нормалната си конфигурация и да се стабилизира тяхната работа.

Този подрежим е много полезен при приложения, при които МК остава в режим Sleep за дълъг период от време. Нормалният Sleep режим на работа е полезен при приложения, при които е необходимо МК да се „събужда“ бързо и често.

Работа на периферните блокове в режим Sleep

Някои периферни блокове, които могат да работят в режим Sleep, няма да работят правилно, когато са в маломощен Sleep режим. Маломощният Sleep режим е предназначен за използване със следните блокове:

- За избягване на смущения в захранването (BOR);
- За следящия таймер (WDT);
- За входа за външно прекъсване и входовете за прекъсвания при промяна на състоянието;
- За Таймер 1 (с външен източник на тактов сигнал).

Крайният потребител трябва да реши, какво е приемливо за приложението му при задаване на стойност за бит VREGPM, за да се гарантира правилната работа на устройството в режим Sleep.

Режим IDLE

Когато бит IDLEN (Idle Enable) е нулиран (IDLEN = 0), инструкция SLEEP ще постави МК в пълен Sleep режим. А когато IDLEN = 1, инструкция SLEEP постави МК в режим IDLE mode. В този режим ЦП и всички операции с паметта се прекратяват, но тактовите сигнали за периферните блокове продължават да са активни. Режимът е подобен на режим DOZE, с изключение на това, че в режим IDLE ЦП и PFM са изключени.

За разлика от режим Sleep, в режим IDLE периферните блокове, които използват F_{OSC}, продължават да работят. Тези от тях, които използват HFINTOSC, LFINTOSC или SOSC, ще продължат да работят и в двата режима - Idle и Sleep.

Ако CLKOUT е разрешен (CLKOUT = 0, Конфигурационна дума 1), в режим Idle изходът ще продължава да работи.

От режим IDLE се излиза, когато настъпи прекъсване (дори ако GIE = 0), но бит IDLEN остава непроменен. МК може отново да влезе в режим IDLE с инструкция SLEEP.

Ако битът за възстановяване след прекъсване е в 1(ROI = 1), прекъсването, което изведе МК от режим Idle, също води до възстановяване на работата на ЦП с пълна честота, ако doze също е разрешен.

В режим Idle НУ от WDT е забранено, като вместо това ще събуди МК.

WDT може да изведе МК от режим Idle по същия начин, както го „събужда“ от режим Sleep. Бит DOZEN не се засяга.

22. Управление на постоянната памет

Постоянната памет (Nonvoatile Memory – NVM) е два типа –програмната Flash памет (Program Flash Memory - PFM) и EEPROM паметта за данни.

Тя е достъпна по два начина – чрез регистри FSR и INDF и чрез регистровия интерфейс NVMREG.

Времето за запис се управлява от вграден таймер. Напреженията за запис/изтриване се генерират от вграден генератор, предназначен да работи в обхвата на работното напрежение на МК.

NVM може да бъде защитена по два начина: защита на кода и защита от запис.

Защитата на кода (битове $\overline{\text{CP}}$ и $\overline{\text{CPD}}$ в Конфигурационна дума 5) забранява достъпа, четенето и записа до PFM и EEPROM, посредством външен програматор. Тя не засяга функциите за самозаписване и изтриване. Тя може да бъде ресетирана чрез програматор, който изчиства цялата постоянна памет на МК, конфигурационните битове и потребителските идентификационни номера (ID).

Защитата от запис забранява самозаписването и изтриването на част или цялата PFM, което се определя от битове WRT<1:0> Конфигурационна дума 4. Защитата от запис не засяга възможността програматор да чете, записва или изтрива паметта на МК.

Процесите по защитата, записа, четенето, проверката и др. на програмната Flash памет и EEPROM паметта за данни изискват по-задълбочено разглеждане и спазване на редица процедури, дадени в [7], на които тук няма да се спираме.

23. Блок за цикличен контрол с излишък

Цикличният контрол с излишък представлява алгоритъм за проверка за грешки при предаване и съхранение на данни чрез използване на контролна сума. Тя се изчислява от предавателя и се изпраща заедно с данните. Той от своя страна също я изчислява, като по такъв начин се установява правилността на данните, в зависимост от това, дали двете суми съвпадат или не. Ако се установят грешки, е възможно чрез CRC да се коригира информацията или да се поиска тя да се изпрати отново.

Блокът за цикличен контрол с излишък (Cyclic Redundancy Check - CRC) представлява апаратно реализиран генератор на CRC контролни суми, управляван по програмен път.

Той има следните характеристики:

- Може да се използва всеки стандартен CRC до 16 бита;
- Конфигурируем полином;
- Може да се използва всяка начална стойност до 16 бита;
- Възможен стандартен и обърнат ред на битовете;
- Могат да се добавят допълнителни нули автоматично или от потребителя;
- Скенер на паметта за бързи CRC изчисления при потребителски данни в програмната памет;
- Програмно зареждани регистри за данни за изчисляване на CRC стойности без използване на скенера на паметта;

Блокът CRC служи за изчисляване на контролни стойности за програмната памет. С цел по-бързи CRC изчисления, той работи със скенер на паметта. Скенерът може автоматично да предоставя данни към CRC блока. Блокът CRC може да се управлява и чрез директен запис на данни в SFR, без да се използва скенера.

Блокът CRC може да се използва за откриване на побитови грешки във флаш паметта с помощта на вградения скенер на паметта или чрез потребителската RAM памет. Той може да приема до 16-битов полином с до 16-битова начална стойност. След това изчислена контролна сума в CRC ще се генерира в регистри CRCACC <15:0>. Блок използва реализация на преместващ регистър с операция XOR, за да извърши полиномиално деление, необходимо за изчислението на CRC.

24. Забрана на периферни блокове

МК PIC16F18855/75 имат възможност избрани блокове да бъдат забраняване, като при това се поставят в режим с възможно най-ниска консумация. При всяко условие за НУ те са включени.

Забрана на блок

Забраната на блок има следните ефекти:

- Всички тактови и управляващи входове към блока са спрени; няма логически преходи и той не функционира;
- Блокът се поддържа в състояние на НУ;
- Всички SFR регистри се държат като "нереализирани" – записът в тях е забранен, а при четене се чете 00h;
- Изходите на блока са забранени; I/O изводи се присвояват на следващия блок в зависимост от приоритета.

Когато се забрани блок, всички свързани с него регистри за избор на входовете също са забранени.

Разрешаване на блок

Когато блокът се разреши отново чрез съответен управляващ бит, той е в състояние на НУ, а SFR регистрите се зареждат с подразбиращите се стойности при НУ при включване на захранването.

В зависимост от блока, може да му отнеме до един пълен цикъл на инструкция, за да се стартира той. Не трябва да има взаимодействие с блока (напр. запис в регистри) за време поне една инструкция след неговото повторно стартиране.

Забрана на системния такт

Установяването в 1 на бит SYSCMD в регистър PMD0 забранява системния такт F_{OSC} към периферните блокове. Не всички използват SYSCLK, така че не се засяга работата на всички периферни блокове.

Въпроси за самоконтрол към раздел VI

1. Кои събития могат да установят микроконтролера в начално състояние и какво е предназначението им?
2. Какво е предназначението на таймерите PWRT и OSC? Каква е последователността на генерираните времезакъснения, ако се използват и двата?
3. Обяснете логическата схема за обслужване на прекъсванията, показана на фиг. 22.1.
4. Избройте последователността от програмни и апаратни действия, свързани с обслужването на прекъсване.
5. Как се извършва обслужване на повече от едно прекъсване при наличието на един вектор? Как се задава приоритет на прекъсванията?

6. Какви са основните функционални възможности на блока за генериране на тактови сигнали и какво е характерно за всеки от тях?
7. Какво е предназначението на следящия таймер? Може ли да работи той в режим SLEEP?
8. Какви видове енергоспестяващи режими има микроконтролерът и каква е тяхната същност?
9. Каква последователност от действия се изпълнява за влизане и излизане в режим Sleep?
10. Кои събития могат да изведат микроконтролера от режим SLEEP? Какво е общото при тях?
11. Какво е предназначението на блока за цикличен контрол с излишък?
12. Какво означава, да бъде забранен периферен блок и как става забраняването и разрешаването?

Използвана литература

1. В. Ранковска. Микропроцесорна схемотехника. Габрово, УИ „Васил Априлов”, 2012. ISBN 978-954-683-478-2
2. Катцен, С. PIC микроконтроллеры: Все, что вам необходимо знать. М. Додэка, 2008.
3. Кенаров, Н. PIC микроконтролери. I част. Варна, “Млад конструктор”, 2003 г.
4. Кенаров, Н. PIC микроконтролери. II част. Варна, Млад конструктор, 2006 г.
5. Уилмсхерст, Т. Разработка встроенных система с помощью микроконтроллеров PIC. Принципы и практические примеры. СПб, Корона-Век, 2008 г.
6. M. Predko. Programming and Customizing the PIC Microcontroller. McGraw-Hill, 2008.
7. PIC16(L)F18855/75 Data Sheet. Full-Featured 28/40/44-Pin Microcontrollers. Microchip Technology Inc., 2015-2020.
8. PIC18(L)F24/25K42 Data Sheet. 28-Pin, Low-Power, High-Performance Microcontrollers with XLP Technology. Microchip Technology Inc., 2016-2017.
9. https://bg.wikipedia.org/wiki/CRC#cite_note-1