**1. Generikus mikrovezérlő utasításkészlete. Byte orientált utasítások, bit orientált utasítások, műveletek konstanssal, vezérlő utasítások.**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) utasításkészlete RISC típusú (csökkentett bonyolultságú) utasításokat tartalmaz. Jellemző ezekre, hogy két részből épül fel: a mnemonic, vagyis az utasítás kódja, ami arra utal, hogy mit kell elvégezni és az operandus, ami általában az adatmemória egy regisztere, ami arra utal, hogy mivel kell elvégezni az utasítás műveletét.

Mivel egy utasítás rendszerint a 16 bit szó hosszúságú programmemória egyetlen regiszterét foglalja el, ezért az utasításban egyetlen regiszter címét lehet befoglalni.

A két regiszteres utasítások esetén a második regiszternek kitüntetett (fix) címe van. Ezt nevezzük akkumulátornak és A-val jelöljük. Az ilyen típusú utasításokat byte orientált utasításnak nevezzük, mert byte-ok közötti műveletek elvégzését vezérlik. A műveletek lehetnek logikai, aritmetikai vagy regiszter eltoló műveletek.

A bit orientált utasítások egy adott adatmemória regiszter adott bitjére vonatkoznak és ez a két információ az utasítás operandusában kell szerepeljen. Két típusa van: a bitbeállító utasítások (set vagy clear), és a feltételes (bitértéktől függő) utasítások.

Ha az utasítás mnemonikja tartalmazza az L (Literal) indexet, akkor az operandus nem egy adott regiszter címe, hanem egy konstans. Az utasítás műveletét ebben az esetben az akkumulátor és a konstans között kell elvégezni.

Vannak olyan utasítások, amelyek nem tartalmaznak regiszter típusú operandusokat, hanem a mikrovezérlő egy adott állapotát koordinálják. Ezek a vezérlő utasítások. Ilyenek az ugró utasítások, a szubrutin hívó és a visszatérő (szubrutinból, vagy megszakításból) utasítások, valamint az állapot szinkronizáló utasítások (pl. watchdog, meg a stanby mód).

**2. Generikus mikrovezérlő memóriaszervezése. Programmemória szerkezete, adatmemória felépítése, speciális funkciójú regiszterek (SFR).**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) memória szervezése Harvard típusú. Ez azt jelenti, hogy a program és adatmemória különálló egységeket képeznek és a szerkezetük sem egyforma (lásd a mellékelt ábrát).

A programmemória EEPROM típusú (elektromosan újraprogramozható, lassan írható, gyorsan olvasható, de a tápfeszültség lekapcsolásakor nem veszti el a tartalmát) és 16 bit szóhosszúságú regiszterekből épül fel. Minden regiszter egy-egy gépi kódú utasítást tartalmazhat. A programot a 0000H címtől kezdődően kell beírni a programmemóriába, mivel a mikrovezérlő alapállapotba tevésekor (Reset) mindig ez címeződik meg. A programmemória címzése a programszámlálóval (PC) történik, amelyik rendszerint 13-16 bit hosszúságú. (lásd a mellékelt rajzot). A megcímzett memóriarekesz tartalma az utasításregiszterbe (IREG) íródik át, hogy a teljes utasítás ciklus alatt az utasításdekóder (IDEC) rendelkezésére tudjon állni.

Az adatmemória RAM típusú (nagy sebességgel írható és olvasható, de a tápfeszültség lekapcsolásakor elveszti a tartalmát) és 8 bit hosszúságú regiszterekből épül fel, ez határozza meg az adat szélességét. Az adatbusza az aritmetikai-logikai egységgel (ALU) kommunikál, A címbuszra pedig az utasításdekóder küldi az utasításokban szereplő operandus címét. Mivel a generikus mikrovezérlő esetén az operandus címe 8 bites, ezért egy adatmemória lap 2exp8=256 byte-ot tartalmaz és lapozással lehet a memória kapacitását növelni. Az adatmemória két féle regisztert tartalmaz: általános célú regisztereket (adat regiszterek) és speciális célú regisztereket (SFR). A SFR regiszterek hardveresen vannak huzalozva a mikrovezérlő szerkezeti elemeihez és periféria moduljaihoz, és az a szerepük, hogy az utasítások segítségével (szoftveresen) lehessen beállítani és követni a hardver elemeket. A SFR regiszterek vezérlő és állapot biteket tartalmaznak.

**3. Generikus mikrovezérlő regisztercímzése. Direkt címzés, indirekt címzés, lapválasztás, táblakezelés, veremkezelés.**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) a gépi kódú utasításokkal éri el az adatregisztereit. Az utasítások operandusa rendszerint egy adatmemória regiszter, ennek a tartalmával történik az utasításban szereplő feladat elvégzése. Az operandusok címzése kétféle képen történhet.

Direkt címzés esetén az utasítás tartalmazza az operandus regiszter abszolút címét. Mivel az utasítás az operandus címét 8 biten kódolja, összesen 2exp8=256 byte-ot lehet megcímezni, tehát egy adott memórialap regisztereit.

Indirekt címzés esetén az utasítás nem tartalmazza az operandus regiszter abszolút címét, hanem csak egy utalást (IND) arra, hogy az utasításdekóder hol keresse azt. A mikrovezérlő hardverstruktúrája úgy van megtervezve, hogy az operandus címét egy speciális funkciójú regiszter, a FSR (File Select Register) regiszter tartalmazza. Ezért mielőtt egy indirekt címzésű utasítást használunk, előbb fel kell töltenünk a FSR regiszter tartalmát az operandus megfelelő címével. Mivel a FSR regiszter is 8 bites, összesen 2exp8=256 byte-ot lehet megcímezni, tehát egy adott memórialap regisztereit. Indirekt címzést akkor használunk, amikor egy programhurokban több regiszter tartalmával akarjuk ugyanazt a műveletet elvégezni, a hurok többszöri átjárásával (pl. iteráció).

A táblakezelés egy módszer, amivel a programmemóriába beírt adatokat (pl. szövegeket, konstansokat, melyeket nem akarunk elveszteni a tápfeszültség lekapcsolásakor) tudjuk áttenni az adatmemóriába. Erre szolgál a „RETLA Lit” utasítás, amelyik a szubrutin visszatérő alapfunkción kívül (RET), az utasításban szereplő 8 bites konstanst (Lit) az akkumulátorba teszi. Több ilyen RETLA utasítást egymásután írva szövegsorokat tudunk memorálni és utána behívni az adatmemóriába (az akkumulátorba), az utasítások iteratív apellálásával.

A veremtár egy speciális adatmemória (tehát RAM), ahova a szubrutin, illetve a megszakítás hívásakor a soron következő utasítás címét elmentjük. Ez egy LIFO (Last In First Out) típusuk memória, mivel a beágyazott hívások esetén mindig a legutoljára betett memóriacím hívódik elő először, a RET típusú utasítás hatására. A generikus mikrovezérlő veremtára 8 byte mélységű és egy számláló (SP – Stack Pointer) címezi meg, amit az utasításdekóder vezérel. Ebből következik, hogy 8 réteg mélységben ágyazhatunk be ismételt meghívásokat.

**4.  Generikus mikrovezérlők megszakításrendszere. Megszakításvektorok, prioritások, vezérlő regiszterek, időbeli viszonyok.**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) megszakításrendszere az utasításdekóder köré van szervezve, hiszen a megszakítás aszinkron folyamat (nincsen az utasítás ciklussal szinkronizálva), ezért a vezérlő hardver kell szabályozza.

Megszakítást kérhet elvileg a mikrovezérlő bármelyik periféria modulja, a saját IRQ (Interrupt ReQuest) jele aktiválásával. A megszakítás azonban csak akkor történik meg, ha az engedélyezve van. Két engedélyező jelre van szükség ahhoz, hogy egy periféria modul megszakítást tudjon kezdeményezi. Egy specifikus (modulra jellemző) megszakítás engedélyező bitre (INTEN) és egy általános (minden modult egyszerre) engedélyező megszakítás bitre (GIE).

Ha egy periféria modul mindkét engedélyező bitje 1-esre van állítva, és egy megszakításkérés érkezik tőle, akkor az éppen végrehajtásban levő főprogrami utasítás még befejeződik, a következő utasítás címe elmentődik a veremtárba és a program hardveres vezérléssel átugrik a 0004H címre. Ez az úgynevezett megszakításvektor, vagyis az a programmemória cím, ahonnan a megszakítás rutin kezdődik. Ugyanakkor, a megszakításvektorra való ugrás pillanatában a GIE engedélyező bit hardveresen törlődik (0-ra áll), azért, hogy egy újabb esetleges megszakításkérés ne legyen kiszolgálva mindaddig, amig az éppen futó megszakításrutin be nem fejeződik.

A megszakításrutin utolsó utasítása egy RETI típusú utasítás kell legyen, amelyik hatására a veremtárból átíródik a programszámlálóba (PC) a főprogram soron következő utasításának a címe, tehát a program visszaugrik a főprogramba és folytatja azt, ahonnan abbahagyta. Ugyanakkor a GIE általános megszakításengedélyező bit hardveresen visszaállítódik 1-esre, tehát egy újabb esetleges megszakításkérést tud fogadni a rendszer.

A megszakításengedélyező biteket egy SFR regiszterben kell beállítani, ami a megszakítás engedélyezést kontrolálja hardveresen huzalozva. A generikus mikrovezérlőnk esetében ennek a neve modulINTCTR regiszter. (ahol a modul a megszakítást kérő periféria neve, pl. TMR1INTCTR). A vezérlőbiteket a legegyszerűbben egy bitbeállító utasítással ( BS vagy BC) lehet beállítani.

A generikus mikrovezérlő egy megszakításvektorral rendelkezik, de vannak olyan mikrovezérlők, amelyekbe két vagy több megszakításvektort is beágyaztak. Ebben az esetben szintén egy SFR regiszterrel lehet beállítani, hogy melyik megszakításvektornak legyen nagyobb prioritása (ha két megszakítás érkezik egyszerre, melyik legyen hamarabb kezelve). A prioritás bitek beállítása hasonló módon történik, mint az engedélyező bitek beállítása.

**5. Generikus mikrovezérlők programozása. Tokok programozása, programozói állapot, konfigurációs bitek, hardverkörnyezet, áramkörben történő soros programozás (ICSP).**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) programozásán azt a folyamatot értjük, amely során az assembler nyelven megírt program átfordítódik gépi, bináris kóddá és ez átíródik a fejlesztő platformról a mikrovezérlő programmemóriájába.

Mivel a programmemória EEPROM típusú memória, vagyis elektromos impulzusokkal programozható (írható) memória, a gépi kódok beírása egy speciális hardver áramkörre van szükség, melynek egy része a mikrovezérlőbe van integrálva. A másik része (a nagyobb feszültségű és áramú rész) a mikrovezérlőn kívül van megvalósítva, ezt nevezzük a Programozónak. A Programozó bemenete rendszerint USB porton keresztül kapcsolódik a számítógéphez, amin a fejlesztő platform fut és ezen keresztül fogadja az érkező bináris kódokat. A mikrovezérlő felé öt szállal kapcsolódik, ezek a tápfeszültség (+ és GND), egy vezérlő bit (MCLR) és a soros szinkron kommunikációs vonal (DAT és CLK).

A Programozó vezérlőbitjét a mikrovezérlő RESET bemenetére kell kötni, ezen keresztül lehet vezérelni a programozási üzemmód beállítását a mikrovezérlőn (ez rendszerint úgy történik, hogy a Programozó a vezérlő pínre a tápfeszültségnél nagyobb feszültséget kapcsol, ennek hatására a mikrovezérlő átkapcsol a programozási üzemmódba).

Programozási üzemmódban a mikrovezérlő az arra kijelölt két port bemenetén (a generikus mikrovezérlő esetén DAT=PB6, CLK=PB7) fogadja az adatokat és írja be a programmemóriába a hardveres programozási protokoll szerint.

A mikrovezérlőbe be lehet írni a programot, vagy akár újraírni akkor is, ha már be van forrasztva a céláramkori lapra. Ebben az esetben az említett öt pínt ki kell hozni a lap szélére egy kompatibilis tűsorra, hogy a Programozót hozzá lehessen csatlakoztatni. Arra is kell ügyelni, hogy a céláramköri lapon a programozásra dedikált pínekre csak olyan alkatrészek kapcsolódjanak, amelyek nem akadályozzák meg a programozó jelek átvitelét. Ezt a környezetet nevezzük ICSP (In Circuit Serial Programming) környezetnek és eljárásnak.

Programozáskor, a végrehajtható (egzekutábilis) utasításokon kívül, az úgynevezett konfigurációs bitek felprogramozása is megtörténik. A Programozó ezeket a biteket, az utasításbitektől eltérően, a programmemória más, speciális címére fogja beírni. Ezek a speciális programmemória regiszterek hardveresen vannak huzalozva a mikrovezérlő egyes áramköreihez (hasonlóan ahhoz, ahogy az adatmemória SFR vannak huzalozva), és meghatározzák, vagy befolyásolják a mikrovezérlő adott áramköreinek a működését. Mivel ezek a vezérlő bitek a programmemóriába vannak beírva, a tápfeszültség kimaradásakor sem veszítik el értéküket (eltérően a SFR regiszterektől).

A legfontosabb konfigurációs bitek az órajel generátor működését beállító bitek, a Reset áramkör működését meghatározó bitek, a Watchdog működését engedélyező bit és a programmemória adatvédelmét korlátozó bitek.

**6. Generikus mikrovezérlők speciális regiszterei (SFR). Hardverkörnyezet, STATUS regiszter, WDT (Watch Dog Timer) regiszter, FSR (File Select Regiszter), PCL/PCLATH regiszterek.**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) speciális funkciójú regiszterein (SFR) értjük azokat az adatmemória regisztereket, amelyek hardveresen huzalozva vannak a mikrovezérlő egyes áramköreihez, rendszerint a periféria modulok vezérlő és állapot bitjeihez.

A SFR regisztereket az általános célú regiszterekhez hasonlóan szoftveresen lehet írni és olvasni a mikrovezérlő gépi kódú utasításai segítségével. Rendszerint a megfelelő vezérlő biteket írjuk (pl. a BS vagy BC utasításokkal) illetve a megfelelő állapotbiteket olvassuk (pl. a BTSS vagy BTSC utasításokkal) és segítségükkel tudjuk beállítani egy-egy periféria modul helyes működését, illetve meg tudjuk állapítani, hogy éppen milyen állapot fázisban vannak. A legfontosabb SFR regiszterek a STAUS, WDT, FSR és PC regiszterek.

A STATUS regiszter az utasításdekóderhez (IDEC) kapcsolódó regiszter, többnyire a jelző (flag) biteket és a lapválasztó biteket tartalmazza. A jelző bitek közül a legfontosabbak a Z és a CY flagek, amelyek akkor jeleznek (1-esre állnak), ha az ALU-ban elvégzett utolsó művelet eredménye zéró (00H), illetve colt átvitel (carry). A lapválasztó bitek (P0, P1, P2) segítségével az adatmemória aktuális lapját lehet beállítani. Minden adatmemória lap 256byte kapacitású, hogy 8 bittel lehessen megcímezni egy-egy regisztert. Erre azért van szükség, mert az utasítások operandus címe, illetve a FSR indirekt címző regiszterek 8 bitet tartalmaznak.

A WDT regiszter a watchdog áramkör időzítését végzi. Ebbe a regiszterbe beírt konstans határozza meg, hogy a watchdog áramkör inicializálása után (a CLRWDT utasítással) mennyi idő kell eltelnie amig az áramkör nem generál egy Reset jelet, amellyel a mikrovezérlőt alapállapotba hozhatja. (t=Tosc \* WDT, vagyis az órajel generátor periódusa szorozva a WDT regiszterbe beírt konstanssal). Ahhoz, hogy ez ne történjen meg, a programba megfelelő idöközönként egy-egy CLRWDT utasítást kell elhelyezni, ami újraindítja a watchdog számlálóját.

A FSR regiszter az adatmemória indirekt címzést szolgálja. Indirekt címzés esetén az utasítás nem tartalmazza az operandus regiszter abszolút címét, hanem csak egy utalást (IND) arra, hogy az utasításdekóder hol keresse azt. A mikrovezérlő hardverstruktúrája úgy van megtervezve, hogy az operandus címét egy speciális funkciójú regiszter, a FSR (File Select Register) regiszter tartalmazza. Ezért, mielőtt egy indirekt címzésű utasítást használunk, előbb fel kell töltenünk a FSR regiszter tartalmát az operandus megfelelő címével.

A PC (Program Counter) regisztereket a programmemória címzésére használjuk. Mivel a programmemória címbusza rendszerint 13-16 bites (13 bit a generikus mikrovezérlőnk esetén), ezért két 8 bites SFR regiszterre van szükségünk, hogy a tartalmát írni és olvasni tudjuk. Ezek a PCL és a PCLATH nevű regiszterek, melyek a programszámláló alsó, illetve felső 8 bitjét tartalmazzák. A PCLATH regiszter abban különbözik a PCL regisztertől, hogy tamponként működik (latch), vagyis tartalma nem íródik be rögtön a programszámláló felső felébe (PCH), hanem csak akkor, amikor a PCL regisztert is írjuk. Erre azért van szükség, hogy mind a 16 bit egyszerre íródjon be a programszámlálóba.

**7. Az assembler nyelv szintaktikája. Címkemező, utasításmező, operandusmező, megjegyzésmező, direktívák. A gépi kódra fordítás lépései.**

Mikrovezérlők esetén, a gépi kódú programokat assembler nyelven írjuk meg, a fejlesztői platform szövegszerkesztőjében. Egy assembler nyelven írt utasítás négy mezőben helyezkedik el, szóhatárra igazítva (oszloponként) a szövegszerkesztő egy sorában. Ezek a címke, mnemonik, operandus és komment mezők.

Jellemzően, egy utasítás két fő részből épül fel: a mnemonic, vagyis az utasítás kódja, ami arra utal, hogy mit kell elvégezni és az operandus, ami általában az adatmemória egy regisztere, ami arra utal, hogy mivel kell elvégezni az utasítás műveletét.

Ugró utasítások esetén, vagy a szubrutinok, megszakításrutinok meghívásakor tudnunk kell azt a címet, ahova ugorni akarunk, vagy ahol a meghívott szubprogram kezdődik. Ezért az ugrási pontokat és a szubprogramok kezdetét címkével jelöljük, az ugró utasításokban vagy a szubprogram hívásokban ezekre a címkékre hivatkozunk. A megfeleltetést az assembler fogja elvégezni, átalakítva a címkék helyzetét abszolút címekké.

A megjegyzés mező fontos eleme az assembler nyelven írt gépi kódú programoknak. Mindig egy pontosvesszővel kezdődik a negyedik mezőben és az éppen elvégzendő utasítás folyamatát kommentálja. Azért fontos, mert később (napok, hetek, hónapok múlva) könnyebben vissza tudunk emlékezni arra, hogy mit is akartunk gépi kódban programozni, tekintettel arra, hogy ezen a gépközeli (alacsony szintű) programnyelven a programozás folyamatát relatív nehéz nyomon követni. A kommenteken kívül a nyomon követést elősegíti, ha a programmoduloknak folyamatábrákat készítünk.

Egy assemblerben megírt programnak nem csak végrehajtható (exekutábilis) utasításai vannak. Az assemblert utasítani és egyben informálni kell, hogy hogyan fordítsa le az utasításokat bináris kódba és hogyan küldje el ezeket a mikrovezérlő programmemóriájába. Erre szolgálnak a nem végrehajtható utasítások, vagyis a direktívák. A fontosabb (leggyakrabban használt) direktívák az ORG, amely megmutatja, hogy az utána következő végrehajtható utasításokat milyen programmemória címre kell beírni, az EQU, amely hozzárendel egy címkéhez egy abszolút címet, az END, amelyik a végrehajtható program végét jelöli, és a CONFIG\_, amivel a konfigurációs bitek értékét lehet meghatározni.

Az assemblerben írt gépi kódú program bináris kódra való fordítása (kompilálása) két lépésben történik. Az első átfutáskor az assembler megoldja a címkék átfordítását abszolút címekké, valamint ellenőrzi a szintaktikai (alaki) és szemantikai (értelmezésbeli) hibákat, mint pl.: téves mnemonik jelölés, operandus hiánya, operandus helytelen használata, akkumulátor helytelen használata, címke hiánya vagy téves jelölése. Ezeket hibaüzenetek, vagy figyelmeztetések formájában meg is jeleníti. Ha nem talál hibát, akkor a második átfutáskor elkészíti a fordítási listát (az utasítások kód és cím megfeleltetését, valamint a címkék táblázatát a hozzárendelt abszolút értékkel együtt) és a forrásprogramot (src), valamint megjelenít egy statisztikát a lefordított programról.

**8. Generikus mikrovezérlő időzítő perifériái. Hardverkörnyezet, számláló üzemmód, időzítő üzemmód, kiolvasás/összehasonlítás (Capture/Compare) üzemmód, PWM moduláció.**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) egyik leggyakrabban használt periféria modulja az időzítő (Timer) modul. Használata elősegíti, hogy a legtöbb időzítő, számláló, impulzus generáló funkciókat hardveresen oldjuk meg, vagyis az inicializáló rutinok segítségével beállítjuk a modul működését és utána az önállóan fogja megoldani a problémát, anélkül, hogy a mikrovezérlő részéről értékes utasítás időt és programrészeket kéne erre elhasználni. Rendszerint egy egyszerű mikrovezérlő is egynél több időzítő perifériát tartalmaz.

Egy Timer modul három fő egységre oszlik (lásd a mellékelt ábrát): bementi kontrol áramkör, soros bináris számlálók, kimeneti (megszakítást kontroláló) áramkör. A generikus mikrovezérlő számláló része három sorbakapcsolt 8 bites számlálót tartalmaz, vagyis egy 24 bites bináris számlálót alkot, amelyik 000000H-tól FFFFFFH-ig tud számolni, felfele, vagy lefele. A számlálók hardveresen huzalozva vannak három SFR regiszterhez (TMRxL, TMRxM és TMRxH), így lehetséges a számlálók írása és olvasása szoftveresen gépi kódú utasítások segítségével.

A bemeneti áramkört két fő vezérlő bit koordinálja: a számlálást engedélyező bit (TMRxEN), valamint a bejövő impulzusok típusát meghatározó bit (TMRxMD). Ez utóbbi értéke alapján (0 vagy 1), a 24 bites számlálóra két irányból érkezhetnek impulzusok: vagy a mikrovezérlő óragenerátorától érkeznek az impulzusok (ebben az esetben időszámlálásról, mérésről beszélünk), vagy a mikrovezérlő egyik I/O portlábán keresztül a külvilágból (ebben az esetben eseményszámlálásról beszélünk).

Az említett két vezérlő biten kívűl még két vezérlő bittel lehet meghatározni a modul működését: a számlálás irányát (le vagy fel) meghatározó bit (TMRxU/D), valamint a Timer modul üzemmódját meghatározó két bit (CCPxM1 és CCPxM2). Ezek segítségével négy üzemmódot tudunk beállítani a modulon: az előbb ismertetett számláló üzemmódot, a kiolvasó, az összehasonlító és végül az impulzus szélesség moduláló üzemmódot.

A kiolvasó üzemmód esetén, egy vezérlő bit aktiválásakor (CCPxEN) a számlálók tartalma átíródik három párhuzamos regiszterbe, amelyek szintén hardveresen vannak huzalozva három SFR regiszterhez (CCPx1, CCPx2 és CCPx3). Ezáltal nyugodtan lehet hosszabb időn keresztül feldolgozni a számláló értékét egy adott pillanatban, mivel ezt az értéket lementhetjük a CCP tampon regiszterekbe.

Összehasonlító üzemmód esetén a CCP regiszterekbe van lehetőség beírni egy 24 bites értéket és a modul ezzel a számmal fogja összehasonlítani a számlálóba érkező impulzusok számát. Megegyező érték esetén a modul megszakítást kérhet. Ilyen módon meg lehet határozni, hogy a számláló meddig számoljon (időzítsen), mielőtt elindítanánk egy új műveletet.

Impulzus szélesség modulációs üzemmód (PWM) esetén egy második rend három byte-ból álló párhuzamos regiszter is aktiválódik. Ezek szintén hardveresen vannak huzalozva három SFR regiszterhez (PRxL, PRxM és PRxH). Ide kell szoftveresen beírni a generált PWM impulzus periódusának megfelelő 24 bites számértéket. A kitöltési tényezőnek (hogy mennyit áll 1-esben az impulzus egy periódus alatt) megfelelő 24 bites számértéket pedig a CCP regiszterekbe kell beírni. A TMR számlálók bemenetére a mikrovezérlő óragenerátorának az impulzusai kerülnek. A PWM rendszer kimenete a mikrovezérlő valamelyik I/O portjának egyik pínjére vezetődik ki, ezáltal vezérelni lehet egy külső áramkört PWM jellel.

A Timer modulok kimeneti megszakítást kontroláló áramköre biztosítja a megszakításkérő jelet (TMRxINTFL) és be lehet állítani rajta a megszakítást engedélyező biteket (TMRxINTEN és GIE). A Timer modul összes vezérlő és állapot bitjeit két SFR regiszter tartalmazza (TMRxCTR és TMRxINTCTR), ezeken keresztül lehet őket hardveresen beállítani vagy leolvasni.

**9. Generikus mikrovezérlő kommunikációs perifériái. Hardverkörnyezet, I/O portok, párhuzamos portok, szinkron soros portok, aszinkron soros portok. Az üzenetek szerkezete.**

Egy 8 bites generikus mikrovezérlő (egy olyan mikrovezérlő modell, amely a legtöbb kis és közepes bonyolultságú mikrovezérlő általános felépítési elveit tükrözi és szerkezeti elemeit tartalmazza) rendelkezik számos kommunikációs perifériával, amelyeken keresztül tartja a kapcsolatot a külvilággal. Ezek két féle típusúak: párhuzamos vagy soros adatátviteli egységek, aszerint, hogy az adatok átvitele egyszerre történik 8 biten, vagy ütemesen, bitenként viszik át egyetlen adatvonalon. Minden mikrovezérlő rendelkezik legalább egy hardveres párhuzamos és egy hardveres soros kommunikációs megoldással.

A mikrovezérlők alapvető kommunikációs eszköze a párhuzamos I/O port. Ez 8 bit hosszúságú és egy mikrovezérlő egyszerre több porttal is rendelkezhet (PortA, PortB, PortC stb.). Ezeken keresztül tartja a kapcsolatot a külvilággal. Az I/O portok 8 bitenként vannak szervezve, mivel minden port hardveresen kapcsolódik két, speciális funkciójú (SFR) regiszterhez. Az egyik SFR regiszter (DIRx) meghatározza a port bitjeinek az irányát (IN vagy OUT), míg a másik SFR regiszter (PORTx) meghatározza a port bitjeinek az értékét. Ha pl. a DIRA regiszter valamelyik bitjét 0-ra állítjuk, akkor a PortA megfelelő bitje kimenetnek (OUT) konfigurálódik. Ha viszont a DIRA adott bitjét 1-re állítjuk, akkor az A port megfelelő bitje bemenetnek (IN) konfigurálódik. Ha pl. az A port valamelyik bitje kimenetre van állítva, akkor az értékét (0 vagy 1) úgy állítjuk be, hogy a PORTA regiszter megfelelő bitjébe írjuk be ezt az értéket. Ha viszont az A port adott bitje bemenetre van állítva, akkor a PORTA regiszter megfelelő bitjét beolvasva megkapjuk az adott bemeneten levő logikai állapotot. Az I/O portok kezelését nagymértékben megkönnyíti az a lehetőség, hogy a portok paramétereit szoftveresen, gépi kódú utasításokkal tudjuk kezelni.

Az I/O portok pínjei (bitjei) multiplexelve vannak több belső periféria modulhoz, azért, hogy ezek a modulok is tudjanak a külvilággal kommunikálni. Azt, hogy melyik pín milyen modulhoz van hozzárendelve az adott modul SFR regisztereiben található vezérló bitekkel tudjuk kiválasztani. Ilyen módon lehet hozzárendelni I/O píneket a belső, soros kommunikációs modulokhoz is. Ezek rendszerint két félék lehetnek: aszinkron soros kommunikációs modulok, amelyek csak egy fizikai szálon kommunikálnak (ez az adatszál, a DAT) és szinkron soros kommunikációs modulok, amelyek két fizikai szálon kommunikálnak (az adatszál a DAT, és az órajel a CLK). Van olyan szinkron modul is, amely két adatvezetéket használ, egyet a bejövő adatoknak (MISO) és egyet a kimenő adatoknak (MOSI). Ezen kívül még egy engedélyező vezetéket (CS) is használ. Ilyen a struktúrája a leggyakrabban használt SPI nevű szinkron soros modulnak is. Egy másik gyakran használt szinkron soros modul az I2C. A gyakrabban implementált aszinkron soros modulok közül kiemelkedik az USART modul, amelyet mindkét üzemmódra lehet konfigurálni. Aszinkron üzemmódban egy relatív biztonságos adatátviteli protokollt használ, amely START és STOP bitek használatára épül.

Az aszinkron soros protokollok általában legalább egy START bitet használnak a kommunikáció indítására és időzítést használnak az adatbitek küldésére. Minden bit egy periódusnyi időre marad az adatbuszon. A vevő modul a START bittel szinkronizálja magát, és ugyanazt a periódusú időzítést generálja saját magának, mint az adó periódusa. A vevő az időzítés fázisát eltolja fél periódussal, hogy az adatok beolvasását a bitek vonalidejének (az az idő ameddig egy adatbitet az adatvonalon tartanak) a közepén kezdeményezze. Így nagyobb adatátviteli sebességet lehet elérni.

A szinkron soros protokollok az órajel frontjait használják az adatbitek szinkronizálására. Általában az adó modul az órajel felfutó élein teszi ki a következő adatbitet az adatvonalra. A vevő az adatbitek beolvasását az órajel lemenő élein fogja elvégezni. Mivel az órajel közös (mindig az adó generálja és a CLK vonalon elküldi a vevőnek), valamint a kitöltési tényező 50% (egy periódus alatt az órajel ugyanannyi ideig van 1-ben, mint amennyit 0-ban), biztosítva van az olvasás félperiódusos eltolása, vagyis az olvasás mindig az adatbit vonalidejének a közepén történik.

**10. Mikrokontrolleres fejlesztés lépései és eszközei. Integrált fejlesztő környezetek (IDE), a programfejlesztés lépései, programozási alapelvek: címkiosztás, változók használata, paraméterátadás, feladat (task) kezelés, watchdog használata.**

A mikrovezérlős alkalmazások fejlesztőse rendszerint egy kompatibilis integrált fejlesztői platformon történik (IDE – Integrated Development Environment). Azért nevezik integrált platformnak, mert egy platformon gyűjti össze az összes fejlesztéshez szükséges modult. Ezek a következők: a Szövegszerkesztő (Editor), ahol a programot és direktíváit írjuk meg és szerkesztjük szöveges formában; a Fordító (Compilator), amelyik az assembler nyelven (vagy más magasabb szintű nyelven) megírt programot átkonvertálja bináris kódokká; a Szimulátor, amelyik a lefordított programot off-line futtatja akár lépésenként és kijelzi a különböző regiszterek pillanatnyi értékeit; opcionálisan az Emulátor, amivel a programot on-line lehet futtatni akár az alkalmazás áramköri környezetében; ; és a Programátor, amivel a lefordított gépi kódú programot fel lehet tölteni a mikrovezérlő programmemóriájába. A legismertebb IDE platformok a Microchip cég MPLAB fejlesztői környezete, vagy az ARDUINO nevű, nyílt forráskódú fejlesztői platform.

A programfejlesztés lépései:

1 – A feladat elemzése

2 – Az alkalmazás megtervezése

3 – Folyamatábra (Organigram) elkészítése

4 – Kódolás, felhasználva a gépi programozás elveit (megszakításkezelés, perifériakezelés stb.)

5 – Fordítás, ellenőrzés, szimulálás (debugolás)

6 – Betöltés, felprogramozás

7 – Futtatás

8 – Korrekciók

9 – Dokumentálás

A kódolás műveletekor (4. lépés) figyelembe kell venni a gépi kódú programozás alapelveit. A címkiosztás az EQU direktívával történik és minden olyan változónak meg konstansnak adunk az elnevezésére jellemző nevű címkét, amelynek abszolút címét vagy értékét a programban használjuk. A változók címének kijelölésekor figyelembe kell venni a SFR regiszterek, valamint az általános célú regiszterek címkiosztásait (nem keveredhetnek egymással). A főprogram és a szubrutinok, valamint a megszakításrutinok közös változóit paraméterátadásra használhatjuk (pl. a megszakításrutin kiszámítja az értékeit és a főprogram felhasználja vezérlésre, vagy fordítva). A főprogram változóit feladat (task) kezelésre is használhatjuk (pl. a főprogramban változtatjuk a változó értékét, a megszakításrutin vagy valamelyik szubrutin pedig az adott értéknek megfelelő feladatot fogja elvégezni).

Fordításkor (5. lépésnél): Az assemblerben írt gépi kódú program bináris kódra való fordítása (kompilálása) két lépésben történik. Az első átfutáskor az assembler megoldja a címkék átfordítását abszolút címekké, valamint ellenőrzi a szintaktikai (alaki) és szemantikai (értelmezésbeli) hibákat, mint pl.: téves mnemonik jelölés, operandus hiánya, operandus helytelen használata, akkumulátor helytelen használata, címke hiánya vagy téves jelölése. Ezeket hibaüzenetek, vagy figyelmeztetések formájában meg is jeleníti. Ha nem talál hibát, akkor a második átfutáskor elkészíti a fordítási listát (az utasítások kód és cím megfeleltetését, valamint a címkék táblázatát a hozzárendelt abszolút értékkel együtt) és a forrásprogramot (src), valamint megjelenít egy statisztikát a lefordított programról.