Pārtraukumi

Parasti mikrokontrollerus pielieto sistēmas, kuras reaģē uz ārējiem notikumiem. Notikumi apzīmē stāvokļa maiņu kontroles sistēmās un vispārīgi pieprasa sava veida atbildes reakciju no mikrokontrollera. Reakcijas variējas no skaitītāja vērtības palielināšanas uz vieninieku (skaitītāja inkrementēšana), kad darba elements krusto noteikto līmeni (piemēram, lai sistēma atslēgtos, ja kāds ienāk iekārtas darbības apgabalā). Pieņemot, ka kontrolleris var novērot notikumu, tas ir, pastāv ievades rinda, kas maina savu stāvokli, lai apzīmētu notikumu, paliek aktuāls jautājums, kā kontrolleris izvēlas ievades pozīciju, lai nodrošinātu pareizu un savlaicīgu reakciju.

Cits variants ir periodiski pārbaudīt signāla stāvokļa maiņu. Tomēr šai metodei ir savi trūkumi: nelietderīga procesora laika izšķēršana gadījumā, ja signāls maina savu stāvokļi bieži, ka arī šo metodi ir grūti mainīt vai paplašināt. Galu galā, mikrokontrollerim parasti ir daudz vairāk uzdevumu, nekā tikai gaidīt vienu notikumu, tādējādi ar pārtraukumu palīdzību stāvokļa fiksēšana notiek netraucējot citas programmas daļas izpildei.

No otras puses, signāls var būt fiksēts ar noteiktu maksimālo periodu, lai izvairītos no stāvokļa maiņas kļūdainas fiksēšanas, tāpēc stāvokļa maiņas skaitīšanas kodu var palaist no vairākam programmas vietam. Sakarā ar to, ka programmu izmērs aug, noteikt, kurā programmas vietā ir jānolasā stāvokļa maiņu kļūst par laikietilpīgu procesu, līdz ar ko šī pozīcijas ir jāpārskata katru reizi, kad galvenās programmas kods tiek izmainīts. Līdz ar to šāda veida fiksēt stāvokļa maiņu kļūst neizdevīgi un datorprogrammu izstrādātāji meklē jaunus veidus, kā fiksēt šos retos notikumus.

Par laimi, mikrokontrolleri pats par sevi piedāvā ērtu veidu kā to realizet, ko sauc par pārtraukumiem. Šajā gadījumā programma pārveido signālu un ievieš pārtraukumu tikai, ja ir fiksēta stāvokļa maiņa. Tikmēr, kamēr nav stāvokļa maiņas, galvena programma darbojas nereaģējot uz ārējo notikumu. Tiklīdz notikumam mainas stāvokļis, miktrokontrolleris paziņo par pārtraukumu (ISR), kas no ša brīža pārvalda notikumu. ISR nodrošina programmas izstrādātājs.

Pārtraukumu kontrole

Pārtraukumu raksturo divi galvenie biti mikrokontrollera pārtraukumu logikā:

Pārtraukumu atļaujošais (interrupt enable, IE) bits tiek uzstādīts ar programmētāju, lai indicētu ka kontrollerim ir jāizsauc ISR reaģējot uz notikumu. Pātraukumu zīmes (interrupt flag, IF) bits ir uzstādīts ar mikrokontrolleri tiklīdz iestājas notikums un tiek attīrīts automatiski ieejot ISR, vai manuāli ar programmas palīdzību.

Būtībā IF bits parāda, ka pārtraukuma nosacījums ir noticis, bet IE bits atļauj pārtraukumam notikt. IE un IF biti parasti tiek sniegti katram pārtraukumu avotam, par kuru processiem atbild kontrolleris. Tomēr, lai ieekonomētu bitus kontrolleri, vairākus līdzīgus pārtraukuma avotus dažkārt apvieno vienā IE bitā. Piemēram, Motorola HCS12 mikrokontrollerī, katra atsevišķa diskrēta ieeja I/O portā var ģenerēt pārtraukumu. Bet tur ir tikai viens IE bits un līdz ar to tikai viens ISR signals par visu portu. Tomēr katrai ieejai ir savs IF bits, kas ļauj pārtraukumiem strādāt bez grūtībām šajā sistēmā.

Neatkarīgi no IE un IF bitiem, kontrollerim ir papildus vadības biti (pārtraukumu režīms) priekš dažam citam pārtraukuma avotiem. Tos izmanto, lai izvēlētos, kura tieši signālu izmaiņa izsauks pārtraukumu. Reizēm ir pat iespējams, lai programma noreaģētu uz to, ka ieejas signāls nav mainījies. To sauc par līmeņa pārtraukumu.

Tā kā visu pieejamo pātraukumu atslēgšana no programmas ir neizdevīgi un laikietilpīgi (gadījumā, ja programmu tomēr nav jāpārtrauc ar ISR signālu), mikrokontrollerim ir arī globāla pārtraukuma ieslēgšanas bits, kas ieslēdz/izslēdz visus atļautos pārtraukumus.

Līdz ar to, ISR tiek ieslēgs gadījumā, ja gan pārtraukuma avota IE bits, gan globālais IE bits ir aktivizēti.

Atslēdzot pārtraukumus nenozīmē, ka jūs laidīsiet garām notikumus. Notikuma iestāšanas biežums tiek glabāts konkrētā IF bitā, neatkarīgi no tā, vai IE bits ir uzstādīts vai nē (tas attiecas gan uz globālo, gan un vietējo IE). Tātad, ja notikums notiek laikā, kad tās pārtraukums tika atslēgts, un tas pārtraukums vēlāk tiek ieslēgts atkal, tad atbilstošais ISR būs izsaukts, kaut arī nedaudz novēloti. Vienīgais gadījums, kad jūs varat palaist garām notikumu ir tad, kad otrais notikums iestājas pirms pirmais ir fiksēts. Šajā gadījumā otrais notikums (attiecīgi pēdējais notikums, kas notika) tiks apstrādāts un pirmais (respektīvi visi iepriekšējie) notikums tiks zaudēts.

Neatkarīgi no parastiem pārtraukumiem, kuri var tikt izslēgti, daži kontrolieri piedāvā arī nemaskējamos pārtraukumus (NMI), kuru nevar izslēgt ar globālo IE bitu. Šie pārtraukumi noder ļoti svarīgos gadījumos, kad reakciju uz notikumu, nedrīkst atlikt, neatkarīgi no tā, vai tas ietekmēs programmas darbību vai nē. NMI pātraukumam var būt arī savs kontroles bits, lai atsevišķi ieslēgt/izslēgt pārtraukumu. Tas ir realizēts, piemeram, Texas Instruments MSP430 saimes mikrokontrolleros.

Restartējot mikrokontrolleri, gan globalie, gan lokālie pārtraukumi parasti ir atslēgti. Taču programmas izveidotājam ir jāparedz iespēja kodā norādīt globāla IE pārtraukuma bita ieslēgšanu, pirms programma sāk savu darbību.

Pārtraukumu vektora tabula

Papildus pārtraukuma ieslēgšanas uzdevumam, programmētājam ir jāparedz veidu, kā pateikt mikrokontrollerim kuru pārtraukumu servisu izsaukt. Pārtraukumu pārveidošanu ISR tiek panākta ar pārtraukumu vektoru tabulas palīdzību, kas ietver ierakstus par katru atsevišķu pārtraukuma vektoru. Katram vektoram ir sava fikseta adrese vektoru tabulā un fikseta adrese programmas atmiņā. Vektora adresei ir jāsatur ISR sākuma adresi (HCS12), vai arī pārlekšanas instrukciju uz ISR (Atmega16). Ja pārtraukuma nosacījums iestājas un ir jāizsauc attiecīgo ISR, kontrollera programma vai nu pārlec uz tabulā noradīto vietu, vai uz atbilstošo vektoru, atkarība no tabula ievadītiem datiem. Šajā gadījumā gala rezultāts ir lēciens uz atbilstošo ISR.

Pārtraukuma prioritātes

Tā kā kontrolierim ir vairāk nekā viens pārtraukuma avots un tas faktiski var izpildīt dažādu pārtraukumu funkciju, rodas jautājums, kā rīkoties situācijās, kad divi vai vairāki pārtraukuma notikumi iestājas vienlaicīgi. Šāda uzdevuma realizēšanai tika ieviestas pārtraukumu prioritātes.

Lielākā daļa kontrolleru izmanto pārtraukuma vektoru kā norādi uz prioritāti. ATmega16, piemēram, statiski piešķir augstāko prioritāti pārtraukumam ar vismazāko pārtraukuma vektoru. Ja kontrolleris paredz arī NMI, tie, visticamāk, būs ar augstāko prioritāti.

Taču prioritātes var tikt izmantotas ne tikai lai noteiktu, ko izpilda ka pirmo, tās var arī izmantot, lai noteiktu, vai pārtraukums var pārtraukt aktīvo ISR: Pārtraukums ar augstāko prioritāti pārtrauks ISR ar zemāku prioritāti (tīkla pārtraukums). Daži kontrolleri, piemēram, ATmega16, ļauj pārtraukt ISR visiem pārtraukumiem, ja to pārtraukuma bits ir uzstādīts. Tā kā pārtraukums ne vienmēr ir nepieciešams, daudzi kontrolieri atslēdz globālo IE bitu pirms ISR izpildes, vai sniedz kādu citu līdzekļu lai izvēlēties ISR, nosakot vai to var pārtraukt, vai ne.

Protams, statisku prioritāšu piešķiršana ne vienmēr atbilst programmas prasībām. Rezultātā daži kontrolieri ļauj lietotājam dinamiski piešķirtu prioritātēs dažiem pārtraukumiem. Citi ļauj lietotājam izvēlēties ISR ietvaros, kādiem pārtraukumiem ir atļauts pārtraukt ISR.

Pārtraukumu vadība

Lai pārbaudītu pārtraukuma nosacījumu kontrolleri diskretizē ieejas signālu pārbaudes cikla sākumā un salīdzina blakus esošas diskretizācijas vērtības. Ja tiek fiksēts pārtraukuma nosacījums, tad tiek uzstādīta pārtraukuma zīme (karogs). Nākošais solis ir pārbaudīšana, vai IE bits ir uzstādīts, un ja tas tā ir, un neviens cits pārtraukums ar augstāku prioritāti negaida apstrādi, tad tiek aktivizēts pārtraukums un izsaukta ISR. Jāņem vērā, ka notikuma noteikšana ir aizkavēta ar diskretizācijas ķēdi. Tādējādi signālam ir jābūt stabilam vismaz vienu cikla laiku, lai kontrolleri varētu to apstrādāt. Īsākus signālus piefiksēt nevar.

Ārējie notikumi nav labvēlīgi, jo tie tiek radīti ar ārējo aparatūru, kas visticamāk ir saistīts ar kontrolleri bez aizsardzības shēmas. Tas var radīt kļūdaino signāla salīdzināšanu, jo pie tuviem pārtraukuma nosacījumiem esošais troksnis uz signāla, var radīt pārtraukumu tur, kur tā nav. Lai novērstu šāda trokšņa ietekmi uz programmu, daži mikrokontrolleri paredz trokšņu slāpēšanu.

Iekšējiem notikumiem (taimera notikumi, vai paziņojums, ka ir saņemts baits no sērijas interfeisa), attiecīgais modulis nodrošina IF bita ieslēgšanu. No šā brīža darbojas parasta pārtraukuma loģika. Protams, iekšējos notikumus neietekmē troksnis, tāpēc šajā gadījumā nav jāveic aizsardzības pasākumi.

ISR izsaukums

Lai izsauktu ISR, pirmkārt, kontrollerim ir jāsaglabā atmiņas adrese, kur ir jāatgriežas pēc pārtraukuma izpildes. Kad tas ir paveikts, tiek iestādīts pārtraukums (visbiežāk ar globālo pārtraukumu zīmi), un izpildītas ISR instrukcijas. ISR izpilde ir līdzīga parastas apakšprogrammas izpildei, vienīga atšķirība ir tāda, ka iziet no ISR izpildes var ar instrukcijas „atgriezties no pārtraukuma” RETI palīdzību, kas atceļ to, ko kontrolleris darīja pirms ISR

izsaukšanas: tā aktivizē globālo IE bitu un atjauno reģistrus, ja tie ir saglabāti, ka arī ielādē programmas izpildes beigu adresi PC stekā. Daži kontrolieri, piemēram, ATmega16, pārliecinās arī, ka pēc atgriešanās no ISR, vismaz viena no galvenās programmas rindām tiks izpildīta pirms nākamas ISR izsaukšanas. Tas nodrošina to, ka neatkarīgi no tā, cik bieži notiek pārtraukumi, galvenais programma nevar ciest, lai gan tas izpildes laiks būs liels.

Visa darbību ķēde no notikuma līdz ISR pirmās instrukcijas izpildei izraisa aizturi uz notikumu, kas tiek apkopota pārtraukuma latentumā. Latentums parasti mēdz būt robežās 2-20 ciklu, atkarībā no tā, ko tieši kontrolieris dara, pirms ISR izpildes. Laika kritiskās sistēmās, pārtraukuma latentums ir būtisks mikrokontrollera raksturīgais lielums, tāpēc tās augšējā robeža parasti noteikta rokasgrāmatā. Ja latentuma jautājums ir aktuāls, ir jāmeklē kontrolleri, kuriem ir augstas oscilatoru frekvences, un kuri nesaglabā reģistru atmiņā.

Pārtraukumu pakalpojumu rutīnas

Pārtraukuma pakalpojumu rutīnas ietver kodu, kas ir vajadzīgs, lai reaģētu uz pārtraukumu. Tas varētu būt: pārtraukuma karoga noņemšana, ja tas vēl nav izdarīts, vai pārtraukuma atslēgšana, ja tas vairs nav nepieciešams. ISR var ietvert arī kodu, kas reaģē uz notikumu, kuriem ir trigera pārtraukums. Tomēr lēmums, ko darīt ISR un ko darīt pamata programmā bieži vien nav viegli un ir atkarīgs no situācijas, tāpēc labs mikrokontrollera dizains prasa lielas pārdomas un pieredzi.

Pārtraukums vai skaitītāja inkrementēšana

Sākuma jautājums ir vai pielietot pārtraukumu sistēmu, vai inkrementēt skaitītāju. Kā, piemēru apskatīsim pogu, kas darbināta ar cilvēku. Tik ilgi, kamēr poga ir nospiesta, līdzstrāvas motoram jābūt aktīvam. Pēc pirmā acu uzmetiena, tas izskatās pēc skaitītāja inkrementēšanas uzdevuma, jo mēs esam ieinteresēti stāvokļa saglabāšanā, tomēr ar to nepietiek. Patiesībā, izvēle galvenokārt ir atkarīga no to, kas vēl notiek mūsu programmā. Ja tas ir vienīgais notikums, tad var pielietot pārtraukumu un kontrollera miega režīmu galvenajā programmā. Gadījumā, ja programmai paralēli tiek novēroti arī citi notikumi, tad izdevīgāk ir izmantot skaitītāja inkrementēšanu. Nav jāuztraucas par jūsu risinājuma neprecizitāti. Galu galā, gan cilvēki, gan līdzstrāvas motors nav ļoti precīzi instrumenti, tāpēc laika skaitīšana te nav kritiska. Veicot pārbaudi ik pēc pāris milisekundēm būs diezgan pietiekami, un jūs varat iekļaut arī citu kodu šajās 1-10 ms.

Reakcija ISR vai darba programma

Nākamais lēmums, kas ir jāpieņem, kurā programmas daļā reaģēt uz notikumu, ISR vai galvenajā programmā. Acīmredzama izvēle būtu ISR, jo to izsauc, reaģējot uz notikumu, bet tas ne vienmēr būs prātīgs lēmums: ISR pārtrauc galveno programmu, līdz ar ko, ja ISR ir garš, uzdevumu izpildes atlikšana būs arī ievērojama. Lai gan galvenā programma izpilda mazos darbus, kuri var nedaudz tikt atlikti, taču nav ieteicams atlikt tos uz ilgu laiku. Tātad jums vajadzētu domāt par to, kā aizture ietekmēs jūsu galveno programmu.

Ja programmai ir jāreaģē uz vairākiem pārtraukuma nosacījumiem, tad ir vajadzīgs apsvērt aiztures sekas. Ņemiet vērā, ka pat ja pārtraukums notiek reti, tas nenozīmē ka laiks nav svarīgs.

Dažas lietas par ko vēl ir jāpadomā. Pirmais ir, kā paziņot pamata programmai, ka notikums ir iestājies. Operētājsistēmas ietvaros ir iespējams nosūtīt notikumu, vai ielikt datus rindā, vai ko līdzīgu. Bez operētājsistēmu, jums ir ierobežots flagu izmantošanas skaits, ko jūs varat iestādīt ISR un notīrīt galvenajā programmā.

Vēl viena problēma ir papildus kavēšanas. Reaģējot uz notikumu galvenā programma pagarina pārtraukuma latentumu, kas ne vienmēr ir pieņemami. Tas īpaši ir izjūtams daudz uzdevumu sistēmas, kurās ISR var izraisīt uzdevumu pārplānošanu.