ო.ქართველიშვილი, ც. ხოშტარია, ს. ხოშტარია

მიკროპროცესორული სისტემები

ნაწილი I

მიკროკონტროლერის არქიტექტურა

თბილისი

2015

უაკ 681.3

სახელმძღვანელოში განხილულია მიკროპროცესორული სისტემების დაგეგმარების საკითხები. იგი შედგენილია ერთსახელა კურსის პროგრამის მიხედვით და წარმოადგენს მის პირველ ნაწილს. განიხილება მიკროკონტროლერის არქიტექტურა AVR ოჯახის ერთ-ერთ მიკროკონტროლერის მაგალითზე. ნაშრომში აღწერილია მიკროკონტროლერის შემადგენლობაში შემავალი ბლოკების სტრუქტურა, მათი მუშაობის პრინციპი და ურთიერთკავშირი მიკროკონტროლერის მუშაობის დროს. რამდენადაც მიკროკონტროლერები გამოიყენება ობიექტების მართვის სისტემებში, იმდენად მნიშვნელოვანი ადგილი ეთმობა მიკროკონტროლერების ობიექტებთან კავშირის ორგანიზაციას ინტერფეისების საშუალებით. განიხილება სხვადასხვა ინტერფეისის პროტოკოლები. მიკროკონტროლერების სისტემების პროექტირების დროს ერთ-ერთი ცენტრალური ადგილი უკავია სისტემის ფუნქციონირების პროგრამის შედგენას, რაც აღწერილია შემდგომ ნაწილში. ამის გამო წიგნში მოცემული მასალა ორიენტირებულია მიკროპროცესორის ბლოკებთან პროგრამის ურთიერთქმედების შესწავლაზე სხვადასხვა ამოცანის შესრულების დროს.

სახელომძღვანელო განკუთვნილია საქართველოს ტექნიკური უნივერსიტეტის ინფორმატიკისა და მართვის სისტემების ფაკულტეტის შესაბამისი სპეციალობის სტუდენტებისათვის და აგრეთვე იმ სპეციალისტებისთვის, რომლებიც მუშაობენ მიკროპროცესორული სისტემების პროექტირებისა და ექსპლოატაციის სფეროში..

შესავალი

დღესდღეობით სხვადასხვა ობიექტის მართვისთვის ფართოდ გამოიყენება მიკროკონტროლერის ბაზაზე აგებული გამოთვლითი სისტემები.

მიკროკონტროლერი (ინგ. Micro Controller Unit, MCU) არის მიკროსქემა, რომელიც განკუთვნილია ელექტრონული მოწყობილობების მართვისთვის. მიკროკონტროლერის ერთ კრისტალში შერწყმულია პროცესორისა და პერიფერიული მოწყობილობების ფუნქციები, შეიცავს ოპერატიული მეხსიერების მოწყობილობასა (ომმ) და მუდმივი მეხსიერების მოწყობილობას (მმმ). შინაარსობრივად ის ერთ კრისტალში მოთავსებული კომპიუტერია, რომელიც ასრულებს მარტივ ამოცანებს. ერთკრისტალიან მიკრო-ეგმ-ის გამოჩენას უკავშირებენ მართვის პროცესში კომპიუტერული ავტომატიზაციის მასობრივი გამოყენების ერის დასაწყისს.

ერთკრისტალიან მიკრო-ეგმ-ზე პირველი პატენტი გადაეცათ ამერიკული ფირმა Texas instruments-ის თანამშრომლებს მ.კოჩრენს და გ. ბუნუს 1971 წელს. მათ მიერ წარმოდგენილი იდეა იყო, რომ ერთ კრისტალზე განეთავსებინათ არა მარტო პროცესორი, არამედ მეხსიერებაც შეტანა-გამოტანის მოწყობილობებთან ერთად.

1976 წელს ამერიკულმა ფირმა Intel-მა შექმნა მიკროკონტროლერი i8048. ხოლო 1978 წელს ფირმა Motorola-ამ გამოუშვა თავისი პირველი მიკროკონტროლერი MC6801, რომელიც ბრძანებათა სისტემის მიხედვით თავსებადი იყო ადრე გამოშვებულ MC6800 მიკროპროცესორთან. ოთხი წლის შემდეგ 1980 წელს ფირმა Intel-მა გამოუშვა მიკროკონტროლერი i8051. პერიფერიული მოწყობილობების მოხერხებულმა ნაკრებმა, გარე და შიგა პროგრამული მეხსიერების ამორჩევის მოქნილმა შესაძლებლობებმა და მისაღებმა ფასმა განაპირობეს ამ კონტროლერის წარმატება ბაზარზე. ტექნიკური თვალსაზრისით მიკროკონტროლერი i8051 იმ დროისთვის წარმოადგენდა რთულ ნაკეთობას, რომლის კრისტალზე განთავსებული იყო 128 ათასი ტრანზისტორი, რაც 4-ჯერ აღემატება თექვსმეტ თანრიგიან მიკროპროცესორ i8086-ში განთავსებულ ტრანზისტორების რაოდენობას.

ამჟამად არსებობს 200-ზე მეტი მოდიფიკაციის მიკროკონტროლერი, რომლებსაც დაახლოებით ოცი კომპანია ამზადებს. სპეციალისტებში პოპულარობით სარგებლობს 8-თანრიგა Microchip Technology ფირმის PIC და Atmel ფირმის AVR მიკროკონტროლერების ოჯახები, 16-თანრიგიანი TI-ფირმის მიკროკონტროლერი MSP430, ასევე 32-თანრიგიანი მიკროკონტროლერი ARM არქიტექტურით, რომელსაც ამუშავებს ARM Limited ფირმა.

მიკროკონტროლერების პროექტირების დროს საჭიროა დავიცვათ ბალანსი ერთი მხრივ ზომებსა და ღირებულებას შორის, ხოლო მეორე მხრივ- მოქნილობასა და წარმადობას შორის. სხვადასხვა გამოყენებისათვის ოპტიმალური თანაფარდობა ამ და სხვა პარამეტრებისთვის საგრძნობლად განსხვავდება, ამიტომ არსებობს მიკროკონტროლერების ტიპების დიდი რაოდენობა, რომლებიც განსხვავდება პროცესორული მოდულის არქიტექტურით, ჩაშენებული მოწყობილობის ზომითა და ტიპით, პერიფერიული მოწყობილობის ნუსხით, კორპუსის ტიპებით და ა.შ.

ჩვეულებრივი კომპიუტერის მიკროპროცესორისგან განსხვავებით, მიკროკონტროლერებში ხშირად გამოიყენება მეხსიერების ჰარვარდის არქიტექტურა, ე.ი. შესაბამისად, ომმ-სა და მმმ-ში მონაცემები და ბრძანებები განმხოლოებულად ინახება.

გარდა ომმ-ისა მიკროკონტროლერს შეიძლება ჰქონდეს ჩაშენებული ენერგოდამოუკიდებელი მეხსიერება პროგრამებისა და მონაცემების შენახვისათვის. ხშირ

შემთხვევაში კონტროლერებში საერთოდ არ არსებობს სალტეები გარე მეხსიერებასთან მიერთებისთვის. ყველაზე იაფფასიანი ტიპის მეხსიერება გვაძლევს საშუალებას მხოლოდ ერთჯერადად მოვახდინოთ პროგრამის ჩაწერა. ასეთი მოწყობილობები გამოიყენება მასობრივ წარმოებებში, სადაც კონტროლერის პროგრამების განახლება არ ხორციელდება. კონტროლერების სხვა მოდიფიკაციებს გააჩნიათ შესაძლებლობა მოახდინოს პროგრამების მრავალჯერადი შეცვლა ენერგოდამოუკიდებელ მეხსიერებაში.

ქვევით მოცემულია პერიფერიული მოწყობილობების არასრული სია, რომელიც შეიძლება მიკროკონტროლერებში იყოს ჩაშენებული:

უნივერსალური ციფრული პორტები, რომლებიც შეიძლება აიწყოს, როგორც შეტანაზე, ასევე გამოტანაზე;

- სხვადასხვა შეტანა-გამოტანის ინტერფეისები, როგორებიცაა: UART, PC, SPI, CAN USB, IEEE 1394, Ethernet;
- ანალოგულ-ციფრული და ციფრულ-ანალოგური გარდამსახები;
- კომპარატორები;
- განივ-იმპულსური მოდულატორები;
- ტაიმერი;
- უკოლექტორო ძრავების კონტროლერები;
- დისპლეებისა და კლავიატურის კონტროლერები;
- რადიოსიხშირული მიმღებები და გადამცემები;
- ჩაშენებული ფლეშ-მეხსიერების მასივები;
- ჩაშენებული ტაქტური გენერატორი და მოდარაჯე ტაიმერი.

მწარმოებლები ცდილობენ უზრუნველყონ საკუთარი ნაკეთობების მუშაობა მაღალ სიხშირეებზე, ამავე დროს დამკვეთებს აძლევენ მოდიფიკაციის არჩევანის საშუალებას, რომლებიც გათვლილია სხვადასხვა სიხშირისა და კვების ძაბვაზე მუშაობისთვის.

მიკროკონტროლერების უმეტეს მოდელებში გამოიყენება სტატიკური მეხსიერება ომმ-სა და შიგა რეგისტრებისთვის. ეს აძლევს კონტროლერს შესაძლებლობას იმუშაოს დაბალ სიხშირეზე და ასევე არ დაკარგოს მონაცემები ტაქტური გენერატორის სრული გაჩერების გამო. ხშირადაა გათვალისწინებული სხვადასხვა ენერგოდამზოგველი რეჟიმები, როდესაც დროებით ითიშება პერიფერიული მოწყობილობების ნაწილი და გამომთვლელი მოდული.

მიკროკონტროლერები გამოიყენება სხვადასხვა მოწყობილობებისა და მათი ცალკეული ბლოკების მართვისთვის:

- გამოთვლით ტექნიკაში: დედაპლატები; დისკწამყვანების კონტროლე- რები მყარი და მოქნილი დისკებისთვის CD/DVD;
- ელექტრონიკაში და სხვადასხვა საყოფაცხოვრებო ტექნიკაში, სადაც გამოიყენება მართვის ელექტრონული სისტემები: სარეცხ მანქანებში, მიკროტალღურ ღუმელებში, ტელეფონებსა და თანამედროვე ხელსაწყოებში;
- მრეწველობაში: სამრეწველო ავტომატიკის მოწყობილობებში - პროგრამული რელედან და ჩაშენებული სისტემებიდან პროგრამირებად ლოგიკურ კონტრო- ლერებამდე (პლკ); ჩარხების მართვის სისტემებში (ჩმს).

იმ დროს, როდესაც 8-თანრიგა საერთო დანიშნულების პროცესორები სრულად განდევნა უფრო წარმადმა მოდელებმა, 8-თანრიგა მიკროკონტროლერები კვლავ ფართოდ გამოიყენება. ეს აიხსნება იმით, რომ არსებობს დიდი რაოდენობა მოხმარებისა, სადაც არ

მოითხოვება მაღალი წარმადობა და რაც მთავარია, აქვთ დაბალი ღირებულება, იმავდროულად არსებობს მიკროკონტროლერები, რომლებსაც გააჩნიათ მაღალი გამოთვლითი შესაძლებლობები.

მიკროკონტროლერის პროგრამირება ჩვეულებრივ ხორციელდება ასემბლერის ან C ენებზე, თუმცა არსებობს სხვა ენების კომპილატორები. ასევე გამოიყენება ჩაშენებული ბეისიკის ინტერპრეტატორი.

ცნობილია C ენის კომპილატორები მიკროკონტროლერებისთვის:

- Code Vision AVR (AVR-ისთვის);
- I AR [1] (ნებისმიერი მიკროკონტროლერისთვის);
- Win AVR (AVR/AVR32 მიკროკონტროლერებისათვის);
- Keil (8051 და ARM არქიტექტურისთვის);
- HiTECH (8051 და ARM არქიტექტურისთვის).

პროგრამის გამართვისთვის გამოიყენება პროგრამული სიმულატორები (სპეციალური პროგრამები, რომლებიც ახდენენ მიკროკონტროლერის მუშაობის იმიტაციას პერსონალურ კომპიუტერებზე); შიგა სქემური ემულატორები (ელექტრონული მოწყობილობები, რომლებიც ასევე ახდენენ მიკროკონტროლერის იმიტაციას უშუალოდ ობიექტთან მიერთებით).

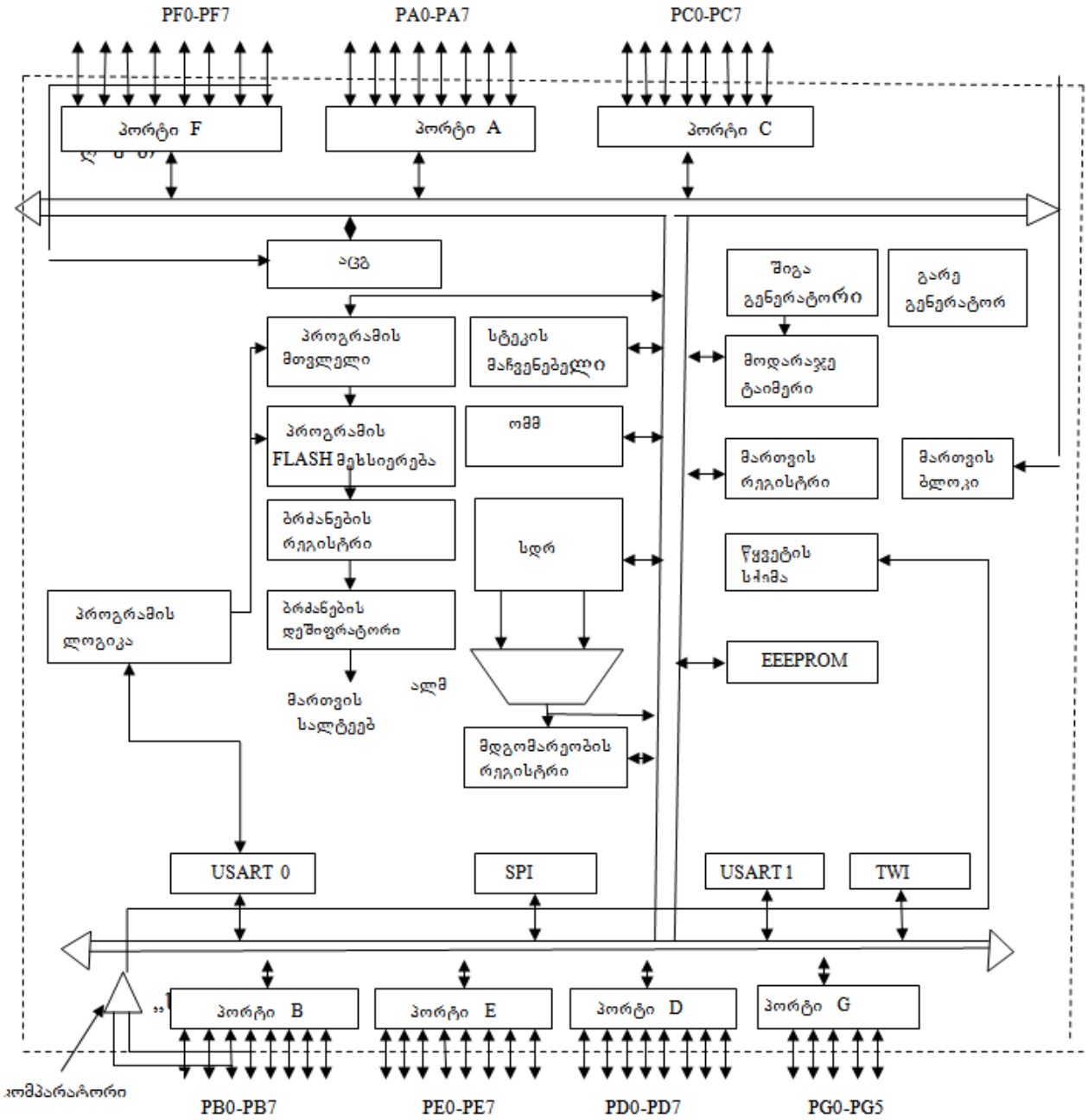
მოცემული წიგნი განკუთვნილია მიკროკონტროლერის არქიტექტურის შესასწავლად. მაგალითის სახით განიხილება AVR ოჯახის ერთ-ერთი ყველაზე სრული მიკროკონტროლერი Atmega 128.

თავი I

მიკროკონტროლერ Atmega 128-ის სტრუქტურა

AVR Atmega 128 არის 8-ბიტიანი მიკროკონტროლერი, რომელიც ბაზირე- ბულია RISC არქიტექტურაზე.

მიკროკონტროლერ Atmega 128–ის სტრუქტურული სქემა მოცემულია 1.1.სურათზე.



სურ.1.1. Atmega 128 მიკროკონტროლერის სტრუქტურული სქემა

Atmega 128 მიკროკონტროლერი შედგება შემდეგი კომპონენტებისგან:

- 128 კბაიტიანი პროგრამირებადი ფლეშ-მეხსიერება;

- 4 კბაიტიანი ელექტრონულად წაშლადი მუდმივი მეხსიერების მოწყობილობა (EEPROM);
- 4 კბაიტიანი ოპერატიული მეხსიერების მოწყობილობა (ომმ - SRAM);
- 32 საერთო მოხმარების (სწრაფი წვდომის) რეგისტრი;
- 192 შეტანა-გამოტანის რეგისტრი;
- ოთხი მოქნილი ტაიმერ/მთვლელი შედარების და ფართო იმპულსური მოდულაციის რეჟიმებით;
- 2 უნივერსალური სინქრონულ/ასინქრონული მიმღებ/გადამცემი (USART);
- ორგამტარიანი ინტერფეისი (TWI) ორიენტირებული ბაიტების გადაცე- მაზე;
- 8 არხიანი 10 თანრიგა აცგ;
- პროგრამირებადი მოდარაჯე ტაიმერი შიგა გენერატორით;
- მიმდევრობითი პორტი SPI;
- არითმეტიკულ-ლოგიკური მოწყობილობა (ალმ);
- შემადარებელი მოწყობილობა (კომპარატორი);
- გარე ობიექტებთან დაკავშირებისათვის, 8 გამოსასვლელიანი 7 პორტი;

მიკროპროკონტროლერის ბირთვი აერთიანებს 32 მრავალფუნქციურ სწრაფი წვდომის რეგისტრს, რომლებსაც ასევე საერთო მოხმარების რეგისტრებს უწოდებენ (General Purpose Registres). ეს 32-ივე რეგისტრი პირდაპირ უკავშირდება არითმეტიკულ-ლოგიკურ მოწყობილობას (ალმ), სადაც სრულდება სხვადასხვა არითმეტიკული და ლოგიკური ოპერაცია რეგისტრებში ჩაწერილ მონაცემებზე და შედეგი ბრუნდება ერთ-ერთ რეგისტრში.

მიკროპროკონტროლერის შემადგენლობაში შედის სხვადასხვა დანიშნულების დამამახსოვრებელი მოწყობილობა: მიკროკონტროლერის მართვის პროგრამის ჩასაწერად და შენახვისათვის გამოყოფილია ენერგოდამოუკიდებელი პროგრამული მეხსიერება (FLASH- მეხსიერება),რომელშიც პროგრამა ინახება მიკროპროცესორის კვებიდან გამორთვის შემთხვევაშიც. მისი ტევადობა 128 კბაიტს შეადგენს; მონაცემთა შენახვისათვის გამოიყენება ოპერატიული დამამახსოვრებელი მოწყობილობა (ომმ) 4 კბაიტის ტევადობით, რომელშიც შესაძლებელია მონაცემთა როგორც ჩაწერა, ისე ამოკითხვა; მუდმივი მონაცემთა შენახვისათვის, რომლებიც არ იცვლება მიკრო კონტროლერის მუშაობის განმავლობაში, გამოიყენება ასევე მუდმივი დამახსოვრების მოწყობილობა (EEPROM) 4 კბაიტ ტევადობით, რომელშიც ინფორმაცია არ იცვლება კვების გამორთვის შემდეგაც.

წარმოდგენილ მიკროკონტროლერს გააჩნია სხვადასხვა პერიფერიული მოწყობილობების ფართო ნაკრები: ანალოგურ ციფრული გარდამქმნელი, რომელიც ანალოგურ ფორმით წარმოდგენილ სიგნალებს გარდასახავს მიკროკონტროლერისთვის შემდგომ დასამუშავებლად მისაღებ ციფრულ ანუ ორობით ფორმატში; შემადარებელი მოწყობილობა (კომპარატორი), რომელიც ადარებს ორ ანალოგურ სიგნალს; 4 ტაიმერ/მთვლელი, რომელთა საშუალებით შესაძლებელია დროის საჭირო ინტერვალის აღრიცხვა; მოდარაჯე მთვლელი შესაძლებლობას იძლევა პროგრამის ხელახალი გაშვების, მისი ავარიული გაჩერების (“ჩამოკიდების”) შემთხვევაში. აღნიშნული მოწყობილობები წარმოდგენილია ბლოკების სახით. თითოეული ბლოკი თავის მხრივ შეიცავს რეგისტრებს, რომელთა საშუალებითაც ხდება პროცესორული ბირთვის ურთიერთობა შესაბამის მოწყობილობებთან სხვადასხვა რეჟიმში დაყენების ან მათი მუშაობის შედეგის გამოყენების მიზნით. ეს

რეგისტრები შეტანა-გამოტანის რეგისტრებია. ასეთი რეგისტრი მიკროკონტროლერში არის 192, თითოეულ მათგანს აქვს თავისი მისამართი.

მიკროკონტროლერის ტაქტირებისათვის იგი შეიცავს სხვადასხვა სახის რამდენიმე სატაქტო სიგნალების გენერატორს: შიგა ანუ ჩაშენებულს და გარე გენერატორს, რომლის ფუნქციონირება გარე სქემებით ხორციელდება.

გარე მოწყობილობილობების მიერთებისთვის კონტროლერის შემადგენლობაში შედის 7 ორმიმართულებიანი 8-თანრიგიანი პორტი: **PA, PB, PC, PD, PE, PG, PF.** აგრეთვე მიმდევრობითი პორტები **USART, SPI, TWI.**

1.1. მეხსიერების ორგანიზაცია

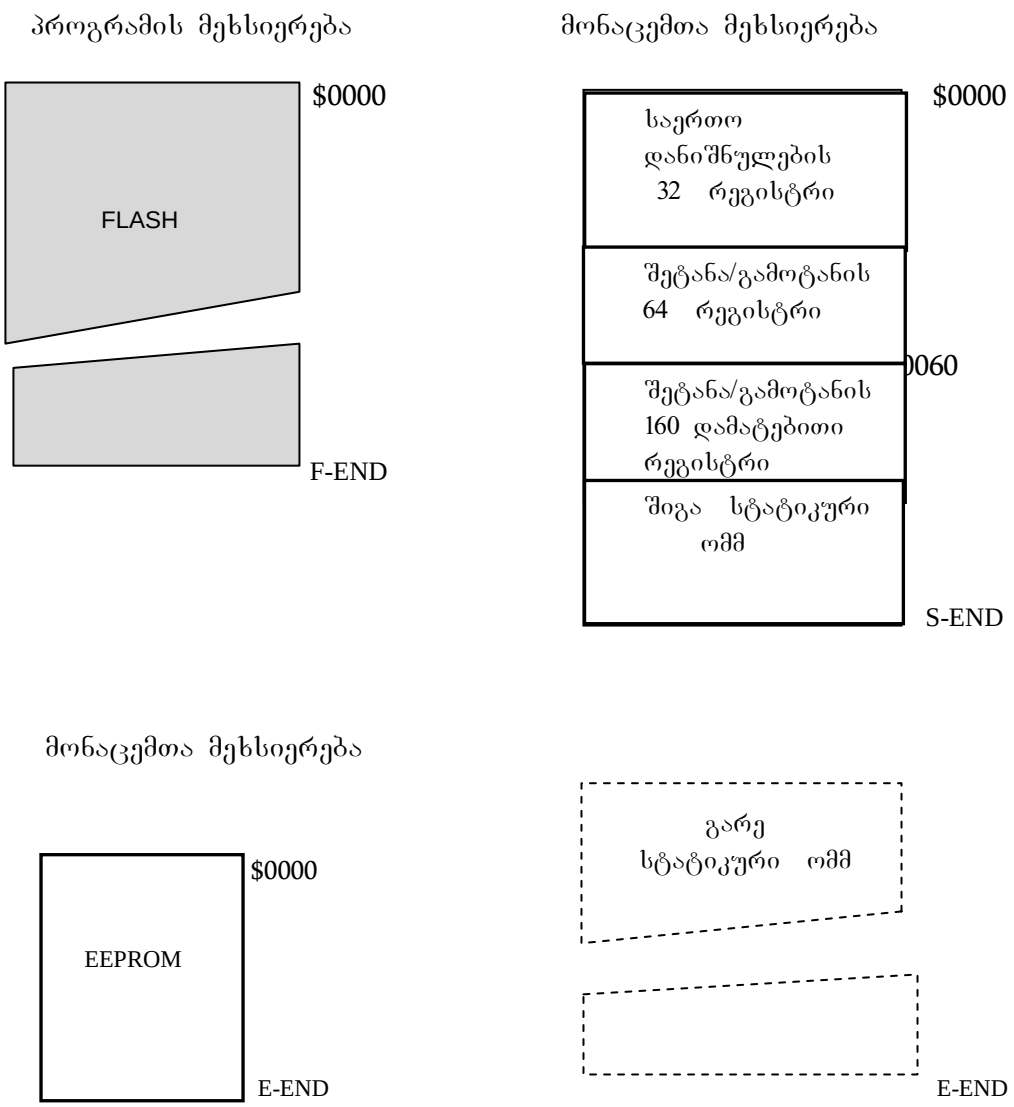
მიკროპროცესორი შეიცავს სხვადასხვა ტიპის და დანიშნულების მეხსიერებას:

- პროგრამების მეხსიერება, რომელშიც ჩაიწერება გამოყენებითი (ჩვენ მიერ შედგენილი) პროგრამები;
- მონაცემთა მეხსიერება, რომელშიც ინახება პროგრამისთვის საჭირო მონაცემები;
- მეხსიერება მუდმივ მონაცემების შენახვისათვის, რომლებიც პროგრამის შესრულების დროს არ იცვლება.

ყველა ტიპის მეხსიერება შედგება ბაიტიანი (8 თანრიგა) უჯრედებისაგან. თითოეულ უჯრედს ენიჭება მისამართი 0-დან N-მდე მეხსიერების ტევადობის შესაბამისად. მისამართის საშუალებით შესაძლებელია მეხსიერების საჭირო უჯრედთან მიმართვა მასში ჩაწერის ან ამოკითხვის მიზნით. მეხსიერების უჯრედების მისამართების ერთობლიობა ქმნის ე.წ. სამისამართო სივრცეს. კომპიუტერებში პროგრამები და მონაცემები ერთ მეხსიერებაშია ჩაწერილი და აქვთ ერთი საერთო სამისამართო სივრცე. წარმოდგენილ მიკროკონტროლერში გამოიყენება ე.წ. ჰარვარდის არქიტექტურა, რომლის დროსაც ყველა ტიპის მეხსიერებას აქვს თავისი სამისამართო სივრცე და მათთან მიმართვის სალტეები, რაც ცენტრალურ პროცესორის ერთდროული მუშაობის შესაძლებლობას იძლევა სხვადასხვა ტიპის მეხსიერებასთან, რის გამოც იზრდება მისი წარმა-დობა. მიკროკონტროლერ ATmega 128–ის მეხსიერების განზოგადებული სქემა ნაჩვენებია 1.2. სურათზე.

1.1.1. პროგრამების მეხსიერება

პროგრამების მეხსიერება განკუთვნილია მიკროკონტროლერის მართვის ბრძანებების შენახვისთვის. იგი ასევე ხშირად გამოიყენება კონსტანტების შენახვი-სთვისაც, რომლებიც არ იცვლება პროგრამების მუშაობის დროს. პროგრამების მეხსიერება არის პროგრამირებადი ელექტრულად წაშლადი მუდმივი მეხსიერების მოწყობილობა (ეწმმმ) (FLASH - მეხსიერება). ვინაიდან მიკროკონტროლერის ბრძანებების სიგრძე ერთი სიტყვის (16 ბიტი) ან მისი ჯერადია (მოთავსებულია ორ ან ორის ჯერად უჯრედში), პროგრამის მეხსიერებას გააჩნია 16 თანრიგიანი ორგანიზაცია. შესაბამისად, მეხსიერების მოცულობა 64კ (64×1024) 16 თანრიგიანი სიტყვაა. მიკროკონტროლერის პროგრამების მეხსიერება ლოგიკურად ორ არათანაბარ ნაწილად იყოფა – გამოყენებითი და ჩამტვირთავი პროგრამების არეები. ამ უკანასკნელში



სურ. 1.2. Atmega 128 მიკროპროცესორის მეხსიერტების სტრუქტურა

შეიძლება განთავსდეს სპეციალური პროგრამა (ჩამტვირთავი), რომელიც მიკროკონტროლერს აძლევს საშუალებას დამოუკიდებლად მოახდინოს გამოყენებითი პროგრამების პროგრამულ მეხსიერებაში ჩატვირთვა და ამოტვირთვა. ჩამტვირთავი პროგრამის რეალიზაცია და ამ არის გამოყენება განიხილება ქვემოთ. მისი გამოყენება არ არის აუცილებელი. თუ მიკროკონტროლერის თვითპროგრამირების შესაძლებლობა არ გამოიყენება, მაშინ გამოყენებითი პროგრამა შეიძლება განთავსებული იყოს მთლიანად პროგრამების მეხსიერებაში.

$0000 მისამართის მქონე უჯრედში განთავსებულია ე.წ. განულების ვექტორი. იგი არის გადასვლის ბრძანება, რომელშიც მითითებულია იმ უჯრედის მისამართი, საიდანაც იწყება ჩვენ მიერ ჩატვირთული პროგრამა (ეს პროგრამა შეიძლება განთავსებული იყოს მეხსიერების ნებისმიერ არეში). მიკროკონტროლერის კვებასთან მიერთების შემდეგ იგი იწყებს მუშაობას $0000 მისამართით ჩაწერილ ბრძანებასთან მიმართვით. აღნიშნული ბრძანების შესრულე- ბის შედეგად მეხსიერებაში სრულდება გადასვლა პროგრამაზე და მისი შესრულება.

$0002 მისამართის უჯრედიდან დაწყებული განთავსებულია წყვეტის ვექტორთა ცხრილი. იგი არის უჯრედების ერთობლიობა, რომლებშიც ჩაწერილია წყვეტის

ვექტორები. წყვეტის ვექტორი მონაწილეობს წყვეტის შესრულებაში. წყვეტა შესაძლებლობას აძლევს მიკროპროცესორს რეაგირება მოახდინოს რაიმე მოვლენაზე გარე ობიექტებში ან მიკროპროცესორის შიგნით. მოვლენის წარმოშობის შემთხვევაში ფორმირდება წყვეტის სიგნალი, რომლის საფუძველზე მიკროკონტროლერი გადაერთვება წყვეტის შესაბამის პროგრამის შესრულებაზე. ეს სრულდება წყვეტის ვექტორების საშუალებით. თითოეულ წყვეტის სიგნალს შეესაბამება რაიმე პროგრამა, რომლის მისამართიც მითითე- ბულია ერთ-ერთ წყვეტის ვექტორში. წყვეტის ვექტორი ასევე არის გადასვლის ბრძანება, რომელიც განთავსებულია წყვეტის ვექტორების ცხრილის ერთ-ერთ უჯრედში.

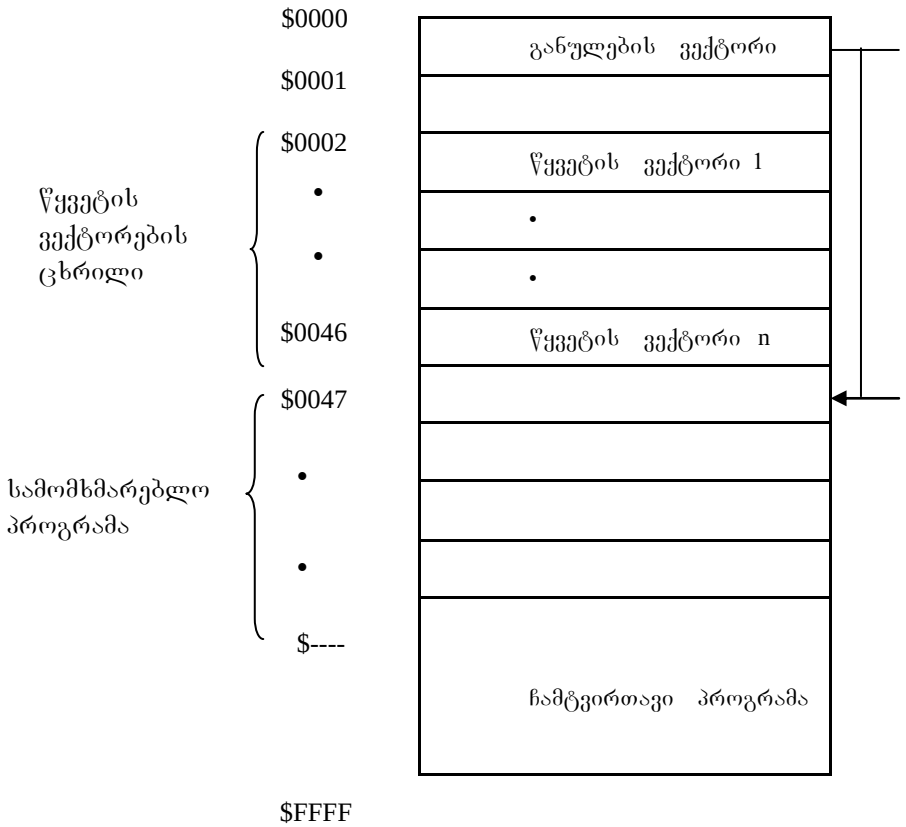
წყვეტის სიგნალის წარმოშობის შემთხვევაში მიკროკონტროლერი მიმართავს შესაბამის უჯრედს ცხრილში და შეასრულებს იქ ჩაწერილ გადასვლის ბრძანებას, რომლის შედეგადაც სრულდება გადასვლა ამ წყვეტის შესაბამის პროგრამაზე.

წყვეტის ვექტორების რაოდენობა განისაზღვრება გამოყენებული წყვეტის რაოდენობის მიხედვით.

წყვეტის ვექტორების ცხრილის და განულების ვექტორის ადგილმდებარე- ობა შეიძლება შეიცვალოს. ისინი შესაძლოა განთავსდეს არა მხოლოდ პროგრამული მეხსიერების დასაწყისში, როგორც ზემოთ არის აღწერილი, არამედ ჩატვირთვის არის დასაწყისში. ამის შესახებ ინფორმაციას განვიხილავთ მოგვიანებით.

იმ შემთხვევაში, თუ პროგრამაში წყვეტა არ გამოიყენება ან წყვეტის ვექტორის ცხრილი განთავსებულია ჩატვირთვის არეში, მაშინ ძირითადი პროგრამა შეიძლება დაიწყოს უშუალოდ $0002 მისამართიდან.

1.3.სურ.-ზე ნაჩვენებია ინფორმაციის განაწილება პროგრამების მეხსიერე- ბაში



სურ.1.3. ინფორმაციის განაწილება პროგრამის მეხსიერებაში

1.1.2. მონაცემთა მეხსიერება

მიკროკონტროლერ Atmega 128-ის მონაცემთა მეხსიერება დაყოფილია სამ ნაწილად: რეგისტრული მეხსიერება, ოპერატიული მეხსიერება (სტატიკური ომმ) და ენერგოდამოუკიდებელი მუდმივი მეხსიერების მოწყობილობა (EEPROM).

რეგისტრული მეხსიერება მოიცავს 32 საერთო დანიშნულების რეგისტრს (სდრ) და სამომხმარებლო შეტანა/გამოტანის რეგისტრებს (შგრ). შეტანა/გამოტანის რეგისტრების სიმრავლე მოიცავს სხვადასხვა დანიშნულების რეგისტრებს: მიკრო- კონტროლერის მართვის რეგისტრს, მდგომარეობის რეგისტრს, მიკროკონტრო- ლერში ჩაშენებული პერიფერიულ მოწყობილობებში შემავალ რეგისტრებს.

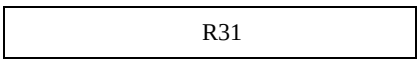
პროგრამის მონაცემთა შენახვისათვის, საერთო დანიშნულების რეგისტრებ- ის გარდა, შეიძლება გამოყენებულ იქნეს 4კბაიტის მოცულობის სტატიკური ომმ. შესაძლებლია მიკროკონტროლერს მიუერთოთ 64 კბაიტამდე მოცულობის გარე სტატიკური ომმ.

სხვადასხვა ინფორმაციის (მაკალიბრებელი კონსტანტები, სერიული ნომრები, გასაღებები და ა.შ) ხანგრძლივი დროით შენახვისთვის გამოყენება მუდმივი მეხსიერება EEPROM. მისი მოცულობა შეადგენს 4კბაიტს. ეს მეხსიერება განთავსებულია ცალკე სამისამართო სივრცეში, მასთან წვდომა შგრ-ს დახმარებით ხორციელდება.

1.1.3 საერთო დანიშნულების რეგისტრები

საერთო დანიშნულების ყველა რეგისტრი გაერთიანებულია სწრაფი წვდომის რეგისტრულ ფაილში, რომლის სტრუქტურა ნაჩვენებია 1.4.სურათზე. როგორც აღვნიშნეთ, 32 სდრ უშუალოდ დაკავშირებულია ალმ-თან, აქედან გამომდინარე ნებისმიერი სდრ შეიძლება გამოყენებული იყოს ყველა ბრძანებაში, როგორც ოპერანდების წყარო/მიმღები. ასეთი გადაწყვეტილება საშუალებას იძლევა ალმ-მა შეასრულოს ერთი ოპერაცია (ამოიკითხოს ოპერანდები რეგისტრული ფაილიდან, შეასრულოს ბრძანება და შედეგი ჩაწეროს რეგისტრულ ფაილში) ერთ მანქანურ ციკლში.

R0
R1
R2
…
R13
R14
R15
R16
R17
…
R26
R27
…
R30



სურ.1.4. საერთო დანიშნულების რეგისტრები

1.1.4. შეტანა/გამოტანის რეგისტრები

პირობითად ყველა შეტანა/გამოტანის რეგისტრი (შგრ) შეიძლება გაიყოს ორ ჯგუფად: მიკროკონტროლერების სამომსახურეო რეგისტრებად და რეგისტრებად, რომლებიც კონკრეტულ პერიფერიულ მოწყობილობას ემსახურებიან (მათ შორის შეტანა/გამოტანის პორტების რეგისტრები). ისინი შეადგენენ შეტანა/გამგამოტანის ძირითად 64 ბაიტიან და დამატებით 160 ბაიტიან სივრცეს. შეტანა/გამოტანის რეგისტრების ჩამონათვალი მოყვანილია 1.1.ცხრილში

ცხრილი 1.1. მიკროკონტროლერ Atmega 128–ის შეტანა/გამტანის რეგისტრები

დასახელება	ფუნქცია
UCSR1C	USART1-ის მართვისა / მდგომარეობის C რეგისტრი
UDR1	USART1-ის მონაცემთა რეგისტრი
UCSR1A	USART1-ის მართვისა და მდგომარეობის A რეგისტრი
UCSR1B	USART1-ის მართვისა და მდგომარეობის B რეგისტრი
UBRR1L	USART1-ის გადაცემის სიჩქარის რეგისტრი, უმცროსი ბაიტი
UBRR1H	USART1-ის გადაცემის სიჩქარის რეგისტრი , უფროსი ბაიტი
UCSR0C	USART0-ის მართვისა და მდგომარეობის C რეგისტრი
UBRR0H	USART0-ის გადაცემის სიჩქარის რეგისტრი, უფროსი ბაიტი
TCCR3C	T3 ტაიმერ / მთვლელის მართვის C რეგისტრი
TCCR3A	T3 ტაიმერ/ მთვლელის მართვის A რეგისტრი
TCCR3B	T3 ტაიმერ/ მთვლელის მართვის B რეგისტრი
TCNT3H	T3 ტაიმერ/ მთვლელის მთვლელირეგისტრის უფროსი ბაიტი
TCNT3L	T3 ტაიმერ / მთვლელის მთვლელი რეგისტრის უმცროსი ბაიტი
OCR3AH	T3 ტაიმერ /მთვლელის თანხვდომის A რეგისტრის უფროსი ბაიტი
OCR3AL	T3 ტაიმერ/ მთვლელის თანხვდომის A რეგისტრის უმცროსი ბაიტი
OCR3BH	T3 ტაიმერ /მთვლელის თანხვდომის B რეგისტრის უფროსი ბაიტი
OCR3BL	T3 ტაიმერ /მთვლელის თანხვდომის B რეგისტრის უმცროსი ბაიტი,
OCR3CH	T3 ტაიმერ /მთვლელის თანხვდომის C რეგისტრის უფროსი ბაიტი
OCR3CL	T3 ტაიმერის / მთვლელის თანხვდომის C რეგისტრის უმცროსი ბაიტი

ICR3H	T3 ტაიმერ /მთვლელის დაპყრობის რეგისტრი უფროსი ბაიტი
ICR3L	T3 ტაიმერ / მთვლელის დაპყრობის რეგისტრის უმცროსი ბაიტი
ETIMSK	ტაიმერ/მთვლელიდან წყვეტის ნიღბის დამატებითი რეგისტრი
ETIFR	ტაიმერ/მთვლელიდან წყვეტის ალმების დამატებითი რეგისტრი
TCCR1C	T1 ტაიმერ/ მთვლელის მართვის C რეგისტრი
OCR1CH	T1 ტაიმერ/ მთვლელის თანხვდომის C რეგისტრის უფროსი ბაიტი
OCR1CL	T1ტაიმერ/ მთვლელის თანხვდომის C რეგისტრის უმცროსი ბაიტი
TWCR	TWI მართვის რეგისტრი
TWDR	TWI მონაცემთა რეგისტრი
TWAR	TWI სამისამართო რეგისტრი
TWSR	TWI მდგომარეობის რეგისტრი
TWBR	TWI გადაცემის სიჩქარის რეგისტრი
OSCCAL	ტაქტური გენერატორის დაკალიბრების რეგისტრი
XMCRA	გარე მეხსიერების მართვის A რეგისტრი
XMCRB	გარე მეხსიერების მართვის B რეგისტრი
EICRA	გარე წყვეტების მართვის A რეგისტრი
SPMCR	პროგრამების მეხსიერების მართვის რეგისტრი
PORTG	G პორტის მონაცემების რეგისტრი
DDRG	G პორტის მონაცემების მიმართულების რეგისტრი
PING	G პორტის გამოსასვლელი
PORTF	F პორტის მონაცემების რეგისტრი
DDRF	F პორტის მონაცემთა მიმართულების რეგისტრი

(გაგრძელება)

დასახელება	ფუნქცია
SREG	მდგომარეობის რეგისტრი
SPH	სტეკის მაჩვენებლის უფროსი ბაიტი
SPL	სტეკის მაჩვენებლის უმცროსი ბაიტი
XDIV	ტაქტური სიხშირის გამყოფის მართვის რეგისტრი
RAMPZ	გვერდის ამორჩევის რეგისტრი
EICRB	გარე წყვეტების მართვის რეგისტრი
EIMSK	გარე წყვეტების ნიღაბის რეგისტრი
EIFR	გარე წყვედების ალმების რეგისტრი
TIMSK	ტაიმერ/მთვლელიდან წყვეტის ნიღბის რეგისტრი
TIFR	ტაიმერ/მთვლელიდან წყვეტის ალმების რეგისტრი
MCUCR	მიკროკონროლერის მართვის რეგისტრი
MCUCSR	მიკროკონტროლერის მართვისა და მდგომარეობის რეგისტრი
TCCR0	T0 ტაიმერ/ მთვლელის მართვის რეგისტრი
TCNT0	T0ტაიმერ/ მთვლელის მთვლელი რეგისტრი
OCR0	T0ტაიმერ/ მთვლელის თანხვდომის რეგისტრი
ASSR	ასინქრონული რეჟიმის მდგომარეობის რეგისტრი
TCCR1A	T1 ტაიმერ/ მთვლელის მართვის A რეგისტრი
TCCR1B	T1ტაიმერ/ მთვლელის მართვის B რეგისტრი
TCNT1H	T1 ტაიმერ/ მთვლელის მთვლელი რეგისტრის უფროსი ბაიტი
TCNT1L	T1ტაიმერ/ მთვლელის მთვლელი რეგისტრის უმცროსი ბაიტი
OCR1AH	T1ტაიმერ/ მთვლელის თანხვდომის A რეგისტრის უფროსი ბაიტი
OCR1AL	T1ტაიმერ /მთვლელის თანხვდომის A რეგისტრის უმცროსი ბაიტი,
OCR1BH	T1 ტაიმერ / მთვლელის თანხვდომის B რეგისტრის უფროსი ბაიტი
OCR1BL	T1ტაიმერ/ მთვლელის თანხვდომის B რეგისტრის უმცროსი ბაიტი
ICR1H	T1ტაიმერ/ მთვლელის დაპყრობის რეგისტრის უფროსი ბაიტი
ICR1L	T1ტაიმერ / მთვლელის დაპყრობის რეგისტრის უმცროს ბაიტი
TCCR2	T2ტაიმერ / მთვლელის მთვლელი რეგისტრი
OCR2	T2 ტაიმერ / მთვლელის თანხვდომის რეგისტრი

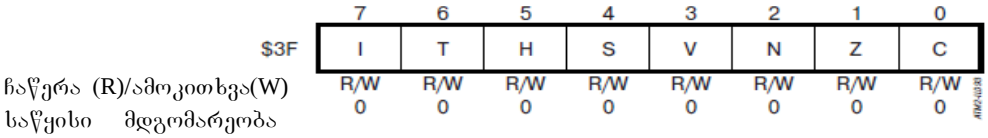
OCDR	შიგა გამართვის რეგისტრი
WDTCR	მოდარაჯე ტაიმერის მართვის რეგისტრი
SFIOR	სპეციალური ფუნქციების რეგისტრი
EEARH	EEPROM მისამართის რეგისტრის უფროსი ბაიტი
EEARL	EEPROM მისამართის რეგისტრის უმცროსი ბაიტი
EEDR	EEPROM მონაცემთა რეგისტრი
EECR	EEPROM მართვის რეგისტრი
PORTA	A პორტის მონაცემთა რეგისტრი
DDRA	A პორტის მონაცემთა მიმართულების რეგისტრი
PINA	A პორტის გამოსასვლელები
PORTB	B პორტის მონაცემთა რეგისტრი
DDRB	B პორტის მონაცემთა მიმართულების რეგისტრი
PINB	B პორტის გამოსასვლელები
PORTC	C პორტის მონაცემთა რეგისტრი
DDRC	C პორტის მონაცემთა მიმართულების რეგისტრი
PINC	C პორტის გამოსასვლელები
PORTD	D პორტის მონაცემთა რეგისტრი

(გაგრძელრბა)

DDRD	D პორტის მონაცემთა მიმართულების რეგისტრი
PIND	D პორტის გამოსასვლელები
SPDR	SPI მონაცემთა რეგისტრი
SPSR	SPI მდგომარეობის რეგისტრი
SPCR	SPI მართვის რეგისტრი
UDR0	USART0 მონაცემთა რეგისტრი
UCSR0A	USART0A მართვისა და მდგომარეობის რეგისტრი
OCSR0B	USART0 მართვისა და მდგომარეობის რეგისტრი
UBRR0L	USART0 გადაცემის სიჩქარის რეგისტრის უმცროსი ბაიტი
ACSR	ანალოგური კომპარატორის მართვისა და მდგომარეობის რეგისტრი
ADMUX	აცგ-ის მულტიპლექსორის მართვის რეგისტრი
ADCSRA	აცგ-ის მართვისა და მდგომარეობის რეგისტრი
ADCH	აცგ-ის მონაცემის უფროსი ბაიტის რეგისტრი
ADCL	აცგ-ს მონაცემის უმცროსი ბაიტის რეგისტრი
PORTE	E პორტის მონაცემთა რეგისტრი
DDRE	E პორტის მონაცემთა მიმართულების რეგისტრი
PINF	E პორტის გამოსასვლელები

შეტანა/გამოტანის რეგისტრებს შორის არის **რეგისტრი SREG** (მდგომარეო- ბის რეგისტრი), რომელიც ყველაზე უფრო ხშირად გამოიყენება პროგრამის შესრულების დროს. მისი თითოეული თანრიგი შეესაბამება რაიმე ალამს, რომელიც გვიჩვენებს მიკროკონტროლერის მიმდინარე მდგომარეობას. ალმების უმრავლესობა დგება ერთში ან ნულში განსაზღრული სიტუაციების შექმნის შემთხვევაში (ბრძანების შესრულების შედეგის შესაბამისად). რეგისტრის ყველა თანრიგთან წვდომა შესაძლებელია, როგორც წაკითხვის, ასევე ჩაწერის რეჟიმებში.

მიკროკონტროლერის განულების შემდეგ რეგისტრის ყველა თანრიგი ნულოვან მდგომარეობაში გადადის. რეგისტრის ფორმატი ნაჩვენებია 1.5. სურათზე, ხოლო მისი აღწერა 1.2. ცხრილში.



სურ.1.5. **SREG** რეგისტრის ფორმატი

1.1.5. მონაცემთა ენერგოდამოუკიდებელი მეხსიერება

მიკროკონტროლერ Atmega 128 აქვს 4 კილობაიტის მოცულობის ენერგოდამოუკიდებელი (EEPROM) მეხსიერება. EEPROM მეხსიერება განთავსებულია თავის სამისამართო სივრცეში. EEPROM მეხსიერებასთან მუშაობისთვის გამოიყენება შეტანა/გამოტანის სამი რეგისტრი: მისამართების რეგისტრი, მონაცემთა რეგისტრი და მართვის რეგისტრი.

თანრიგები	დასახელება	აღწერა
7	I	**წყვეტის საერთო ნებართვა**. ყველა წყვეტის ნების დართვისათვის წყვეტის ალამი ერთიანში უნდა იყოს დაყენებული. ცალკეული წყვეტის ნებართვა /აკრძალვა ხორციელდება წყვეტის ნიღბის რეგისტრების შესაბამის თანრიგებში ერთის ან ნულის ჩაწერით. თუ საერთო წყვეტის ალმის მნიშვნელობა ნულია, წყვეტა აკრძალულია ამ რეგისტრების თანრიგების მდგომარეობის მიუხედავად.. წყვეტის ალმის განულება ხორციელდება აპარატურულად, იმ დროს, როდესაც იწყება წყვეტის შესრულება, წყვეტის აღდგენა მოხდება RETI ბრძანების მეშვეობით მომდევნო წყვეტის დამუშავების ნებადართვისათვის.
6	T	**ბიტის ასლის შენახვა**. რეგისტრის ეს თანრიგი გამოიყენება ,როგორც წყარო ან მიმღები ბიტების ასლის აღების დროს. საერთო დანიშნულების რეგისტრიდან ნებისმიერი მითითებული თანრიგი შეიძლება ჩაიწეროს ამ თანრიგში ან დაყენდეს მოცემული თანრიგის შემცველობის შესაბამისად სპეციალური ბრძანების გამოყენებით.
5	H	**ნახევარ გადატანის ალამი.** ამ ალამის დაყენება მოხდება ერთიანის მდგომარეობაში იმ შემთხვევაში, თუ ადგილი ექნება გადატანას ბაიტის უმცროსი ნაწილიდან უფროს ნაწილში (მესამე თანრიგიდან მეოთხე თანრიგში) ან მოხდება სესხება ბაიტის უფროსი ნაწილიდან უმცროს ნაწილში) ზოგი- ერთი არითმეტიკული ოპერაციის შესრულების დროს.
4	S	**ნიშნის .ალამი** ამ ალამის დაყენება 1-ში მოხდება არითმეტიკული ოპერაციის შესრულების შემდეგ ,როდესაც შედეგი ნაკლებია ნულზე.
3	V	**დამატებით კოდში გადავსების ალამი.** ამ ალამის დაყენება მოხდება ერთიანში იმ შემთხვევაში, თუ ადგილი ექნება მიღებულ

		შედეგში თანრიგთა ბადის გადავსებას და რიცხვები წარმოდგენილია დამატებით კოდში.
2	N	**უარყოფითი მნიშვნელობის ალამი.** ამ ალმის დაყენება მოხდება ერთიანში იმ შემთხვევაში თუ შედეგის უფროსი 7-ე თანრიგი ერთიანშია, წინააღმდეგ შემთხვევაში ალამი ტოლი იქნება ნულის.
1	Z	**ნულის ალამი.** ამ ალმის დაყენება ერთიანში იმ შემთხვევაში მოხდება თუ ოპერაციის შედეგი ნულის ტოლია.
0	C	**გადატანის ალამი.** ამ ალმის დაყენება ერთიანში მოხდება იმ შემთხვევაში თუ ოპერაციის შესრულების დროს ადგილი ჰქონდა ბაიტის უფროსი თანრიგიდან გადატანას

ცხრილი 1.2. SREG მდგომარეობის რეგისტრის თანრიგები

EEPROM მისამართების რეგისტრი

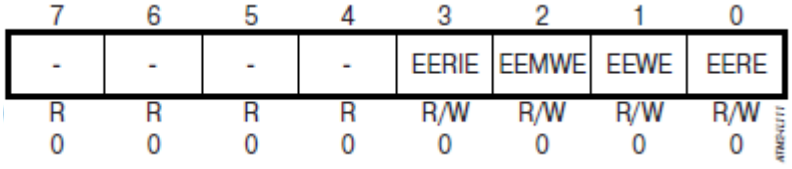
EEPROM მეხსიერების მისამართის EEAR(EEPROM Addres Register) რეგისტრი ფიზიკურად განთავსებულია ორ EEARH:EEARL შეტანა/გამოტანის რეგისტრში. ამ რეგისტრებში ჩაიტვირთება უჯრედის 12-თანრიგა მისამართი, რომელთანაც მოხდება მიმართვა. მისამართის რეგისტრში შესაძლებელია როგორც ჩაწერა, ასევე ამოკითხვა. ამასთან, EEARL რეგისტრში იწერება მისამართის უმცროსი 8 თანრიგი, ხოლო მისამართის უფროსი ოთხი თანრიგი – EEARH რეგისტრის უმცროს თანრიგებში.

EEPROM მონაცემების რეგისტრი

EEPROM მეხსიერების EEDR (EEPROM Data Register) არის 8 ბიტიან მონაცემთა რეგისტრი, რომელშიც ჩაწერის ოპერაციის დროს ჩაიტვირთება EEPROM-ში ჩასაწერი მონაცემები, ამოკითხვის ოპერაციის დროს ამ რეგისტრში განათავსდება EEPROM-დან ამოკითხული მონაცემები.

EEPROM მართვის რეგისტრი

EEPROM მეხსიერების მართვის რეგისტრი EECR (EEPROM Control Register) გამოიყენება EEPROM მეხსიერებასთან მიკითხვის მართვისათვის. ამ რეგისტრის ფორმატი ნაჩვენებია 1.6.სურათზე, ხოლო ამ რეგისტრის აღწერა წარმოდგენილია 1.3. ცხრილში.



წაკითხვა (R)/ჩაწერა(W)
საწყისი მნიშვნელობ

სურ.1.6. EECR მართვის რეგისტრის ფორმატი

თანრიგი	დასახელება	აღწერა
7.....4		გამოუყენებელი იკითხება როგორც “0”
3	EERIE	**EEPROM -დან წყვეტის ნების დართვა**. ეს თანრიგი მართავს წყვეტის გენერირებას,რომელიც ფორმირდება EEPROM-ში ჩაწერის ციკლის დამთავრების შემდეგ. (თუ SREG რეგისტრის I თანრიგი ერთიანის მდგომარეობაშია). EEWE თანრიგის ნულოვან მდგომა-

		რეობის ყოფნის დროს წყვეტის გენერაცია მუდმივად მიმდინარეობს.
2	EEMWE	**EEPROM ჩაწერის ნებადართვის მართვა.** ამ თანრიგის მდგომარეობა განსაზღვრავს EEWE ჩაწერის ნებადართვის ალამის ფუნქციონი- რებას. თუ მოცემული თანრიგი დაყენებულია ერთიანის მდგომარე- ობაში, EEWE-ში ერთიანის ჩაწერისას ანხორციელებს მონაცემების ჩაწერას EEPROM-ში. წინააღმდეგ შემთხვევაში EEW ერთიანში დაყენება არავითარ ეფექტს არ იძლევა. EEMWE პროგრამული დაყენების შემდეგ მისი ჩამოგდება აპარატურულად ხორციელდება.
1	EEWE	**EEPROM ჩაწერის ნებადართვა.** ამ თანრიგის ერთიანში დაყენების შემდეგ განხორციელდება მონაცემების ჩაწერა EEPROM (თუ EEMWE ერთიანშია).
0	EERE	**EEPROM წაკითხვის ნებადართვა.** ამ თანრიგის ერთიანში დაყენების შემდეგ სრულდება მონაცემების წაკითხვა **EEPROM-დან.** წაკითხვის დასრულების შემდეგ ეს თანრიგი ავტომატურად განულდება.

ცხრილი 1.3. EECR რეგისტრის თანრიგები

EEPROM მეხსიერებაში ერთი ბაიტის ჩასაწერად საჭიროა შემდეგი ეტაპების შესრულება:

1. მონაცემის EPROM მეხსიერებაში ჩაწერისათვის მზადყოფნის მოლოდინი (მოლოდინი მანამ, სანამ არ განულდება EECR რეგისტრის EEWE ალამი);
2. მონაცემის FLASH მეხსიერებაში პროგრამის ჩაწერის დასრულების მოლოდინი (მოლოდინი გაგრძელდეს მანამ არ მოხდება SPMC რეგისტრის SPMEN ალამის განულება);
3. ჩაიტვირთოს EERD რეგისტრში მონაცემთა ბაიტი, ხოლო საჭირო მისამართი - EEAR რეგისტრში;
4. EECR რეგისტრის EEEMWE ალამის ერთიანში დაყენება;
5. EECR რეგისტრის EEWE თანრიგში ლოგიკური ერთიანის ჩაწერა.

მეორე პუნქტის შემოტანა განაპირობა იმან, რომ EEPROM მეხსიერებაში ჩაწერა შეუძლებელია შესრულდეს ერთდროულად FLASH მეხსიერებაში ჩაწერასთან ერთად. ამიტომ EEPROM მეხსიერებაში ჩაწერის წინ უნდა დავრწმუნდეთ იმაში, რომ დასრულებულია FLASH მეხსიერების დაპროგრამება. თუ პროგრამაში არ არის ჩამტვირთავი, მაშინ მიკროკონტროლერი ვერ შეცვლის მეხსიერებაში პროგრამის შემცველობას, მეორე ბიჯი შეიძლება გამოტოვებული იყოს.

ჩაწერის ციკლის დამთავრების შემდეგ EEWE თანრიგი ავტომატურად განულდება, რის შემდეგაც პროგრამას შეუძლია შეასრულოს შემდეგი ბაიტის ჩაწერა.

EEPROM ჩაწერის დროს შეიძლება შეიქმნას გარკვეული პრობლემები, რომლებიც წყვეტებით არის გამოწვეული:

1. თუ წყვეტა წარმოიქმნა EEPROM ერთი ბაიტის ჩაწერის პროცედურის ჩამონათვალის 4 და 5 ეტაპებს შორის, მაშინ EEPROM ჩაწერის პროცედურა შეწყდება, რადგანაც EEMWE ნულდება წყვეტის დამუშავების დროს.
2. თუ წყვეტის ქვეპროგრამის დამუშავება დაიწყო იმ დროს, როდესაც მიმდინარეობდა EEPROM-ში ჩაწერა, მოხდება მიკითხვა ამ მეხსიერებასთან, შეიცვლება

EEPROM-ის სამისამართო და მონაცემთა რეგისტრების შემცველობა. რის შედეგად პირველი ჩაწერის პროცესი შეწყდება.

ზემოთ ჩამოთვლილ პრობლემებს თავი რომ ავარიდოთ რეკომენდებულია ავკრძალოთ ყველა წყვეტა (გავანულოთ SREG რეგისტრის I ბიტი) EEPROM მეხსიერებაში ჩაწერის პროცესის 2....5 ეტაპების დროს.

წაკითხვის ოპერაციის შესრულების წინ ასევე აუცილებელია შევამოწმოთ EEWE ალამის მდგომარეობა. საქმე იმაშია, რომ სანამ სრულდება EEPROM-ში ჩაწერის ოპერაცია (EEWE ალამი დაყენებულია 1-ში), არ შეიძლება მოხდეს წაკითხვა EEPROM მეხსიერებიდან და მისამართების რეგისტრის შემცველობის შეცვლა. მას შემდეგ, როდესაც საჭირო მისამართი ჩაიტვირთება EEAR რეგისტრში, აუცილებელია EECR რეგისტრის EERE თანრიგის დაყენება ერთიანის მდგომარეობაში. EEDR რეგისტრში მონაცემთა ჩაწერის შემდეგ შესრულდება ამ თანრიგის აპარატურული განულება.

1.2. ბრძანებათა მთვლელი

ბრძანებათა მთვლელი არის რეგისტრი, რომლის დანიშნულებაა პროგრამების მეხსიერებაში მყოფი ბრძანებების მისამართების ავტომატური ფორმირება. პროგრამიდან ის პირდაპირ მიუწვდომელია. ბრძანებათა მთვლელის შემცველობა დამოკიდებულია პროგრამული მეხსიერების მოცულობაზე. Atmega 128 - 16 თანრიგიანია (რაც შესაძლებლობას იძლევა ჩაიწეროს მასში 64 კილომდე ორბაიტიანი უჯრედის მისამართი).

პროგრამის ნორმალური შესრულების დროს (როდესაც ბრძანებები ერთმანეთის მიმდევრობით სრულდება) ბრძანებათა მთვლელის შემცველობა ავტომატურად იზრდება ერთით ან ორით (დამოკიდებულია შესრულებულ ბრძანებაზე). ამით ფორმირდება ბრძანების მისამართი, რომელიც მომდევნო მანქანურ ციკლში უნდა შესრულდეს. ეს თანამიმდევრობა იცვლება გადასვლის ბრძანების შესრულების, ქვეპროგრამის გამოძახებისა და ქვეპროგრამიდან უკან დაბრუნების ან წყვეტის წარმოქმნის დროს.

კვების ჩართვის ან მიკროკონტროლერის განულების შემდეგ ბრძანებათა მთვლელში ჩაიტვირთება სასტარტო მისამართი $0000. როგორც წესი ამ მისამართით ჩაწერილია უპირობო გადასვლის ბრძანება პროგრამის დასაწყისში გადასვლისთვის. ამ ბრძანების შესრულების პროცესში პროგრამულ მთვლელში ჩაიტვირთება სამომხმარებლო პროგრამის საწისი მისამართი საიდანაც იწყება პროგრამის შესრულება.

წყვეტის წარმოქნის დროს ბრძანებათა მთვლელში ჩაიტვირთება შესაბამის წყვეტის ვექტორის მისამართი. თუ წყვეტა გამოიყენება პროგრამაში, მაშინ წყვეტის ვექტორის მისამართზე უნდა შესრულდეს წყვეტის დამმუშავებელ ქვეპროგრამაზე გადასვლის ბრძანება. ამ შემთხვევაშიც ბრძანების შესრულების პროცესში მთვლელში ჩაიწერება წყვეტის პროგრამის საწყისი მისამართი და ამის შემდეგ იგი დაიწყებს ამ პროგრამის ბრძანებების მისამართების დაფორმირებას.

1.3.სტეკი

სტეკი არის დამამახსოვრებელი მოწყობილობა, რომელშიც მონაცემთა ჩაწერა ან ამოკითხვა სრულდება მიმდევრობით. მიკროპროცესორ Atmega128-ში სტეკისთვის გამოიყენება ომმ-ის უჯრედების ნაწილი. სტეკის უჯრედების დამისამართება

ხდება სტეკის მაჩვენებლის საშუალებით. სტეკის მაჩვენებლის როლში გამოიყენება შეყვანა/გამოყვანის წყვილი რეგისტრი SPH:SPL. კვების ძაბვის მიწოდების (ან განულების) შემდეგ ამ რეგისტრებში ჩაიწერება ნულები. პროგრამის შესრულების წინ აუცილებელია მოვახდინოთ სტეკის მაჩვენებლის ინიციალიზაცია და ჩავწეროთ მასში ომმ-ის ზედა მისამართი. ეს მისამართი არის სტეკის საწყისი მისამართი.

სტეკში მონაცემთა ჩაწერის შემთხვევაში იგი იწერება სტეკის მაჩვენებელში არსებული მისამართის მქონე უჯრედში. ამის შემდეგ სტეკის მაჩვენებლის შემცველობა მცირდება ორით (ვინაიდან მიკროპროცესორში სტეკი გამოყენებულია ბრძანების 16 თანრიგა მისამართის ჩასაწერად ომმ-ის ორ ბაიტიან უჯრედში). მომდევნო ჩაწერა სრულდება სტეკის ორ უჯრედში სტეკის მაჩვენებელში დაფორმირებული ახალი მისამართით. ამის შემდეგ სტეკის მაჩვენებლის მნიშვნელობა კვლავ მცირდება და ა.შ. ბოლო მონაცემთა ჩაწერის შემდეგ სტეკის მაჩვენებელში ფორმირდება პირველი თავისუფალი უჯრედის მისამართი, რომელსაც სტეკის წვერო ეწოდება. ამოკითხვის შემთხვევაში პირიქით სტეკის მაჩვენებლის შემცველობა იზრდება ორით და უჩვენებს ბოლოს ჩაწერილ უჯრედის მისამართს, საიდანაც ხდება ამოკითხვა. ამგვარად, ამოკითხვა სრულდება მეზობელი უჯრედებიდან მიმდევრობით, მისამართების გადიდებით.

სტეკი გამოიყენება ქვეპროგრამის გამოძახების ან წყვეტის პროგრამაზე გადასვლის დროს ე.წ. დაბრუნების მისამართის დასამახსოვრებლად, რომლის საშუალებით სრულდება ქვეპროგრამიდან ძირითად პროგრამაზე დაბრუნება.

სტეკი მისაწვდომია პროგრამულად. სტეკთან მუშაობისთვის ბრძანებათა ნაკრებში არის ორი ბრძანება: სტეკში შეტანის (PUSH) და სტეკიდან ამოტვირთვის (POP).

თავი 2

ტაქტირება, ენერგომოხმარების შემცირების რეჟიმი და განულება

საერთო ცნებები

მიკროკონტროლერში შემავალი კომპონენტების მუშაობის სინქრონიზაცია სრულდება სატაქტო იმპულსების საშუალებით. სატაქტო იმპულსების მიმდევრობა მიეწოდება ცალკეულ ბლოკებს, რომელთა საშუალებით სრულდება მათზე დაკისრებული სამუშაო. სატაქტო სიგნალები ხასიათდებიან თანამიმდევრობის სიხშირით, გაზომილი კილო- ან მეგა- ჰერცებში. რაც უფრო მაღალია სატაქტო სიგნალების სიხშირე მით უფრო სწრაფად მუშაობს მიკროკონტროლერის ცალკეული ბლოკი. სატაქტო სიხშირის არჩევა დამოკიდებულია მიკროკონტროლერისადმი წაყენებულ მოთხოვნებზე.

მიკროკონტროლერ Atmrga 128-თვის შეიძლება გამოვიყენოთ განსხვავებული სატაქტო სიგნალების წყარო, რომელთაც შეუძლიათ სხვადასხვა სიხშირის სატაქტო სიგნალების დაფორმირება. უპირველეს ყოვლისა, ეს არის კვარცის გენერატორი მიერთებული გარე რეზონატორთან. ასევე ტაქტირებისთვის შეიძლება იყოს გამოყენებული მარტივი RC გენერატორი, როგორც შიგა (მაკალიბრებელი), ასევე გარე RC წრედით, ტაქტირებისათვის შეიძლება აგრეთვე გამოვიყენოთ სინქრონიზაციის გარე სიგნალებიც.

მიკროკონტროლერს გააჩნია ენერგომოხმარების შემცირების რამდენიმე (6-მდე) რეჟიმი, ე.წ. "ძილის" რეჟიმები. ეს რეჟიმები საშუალებას იძლევა შემცირდეს

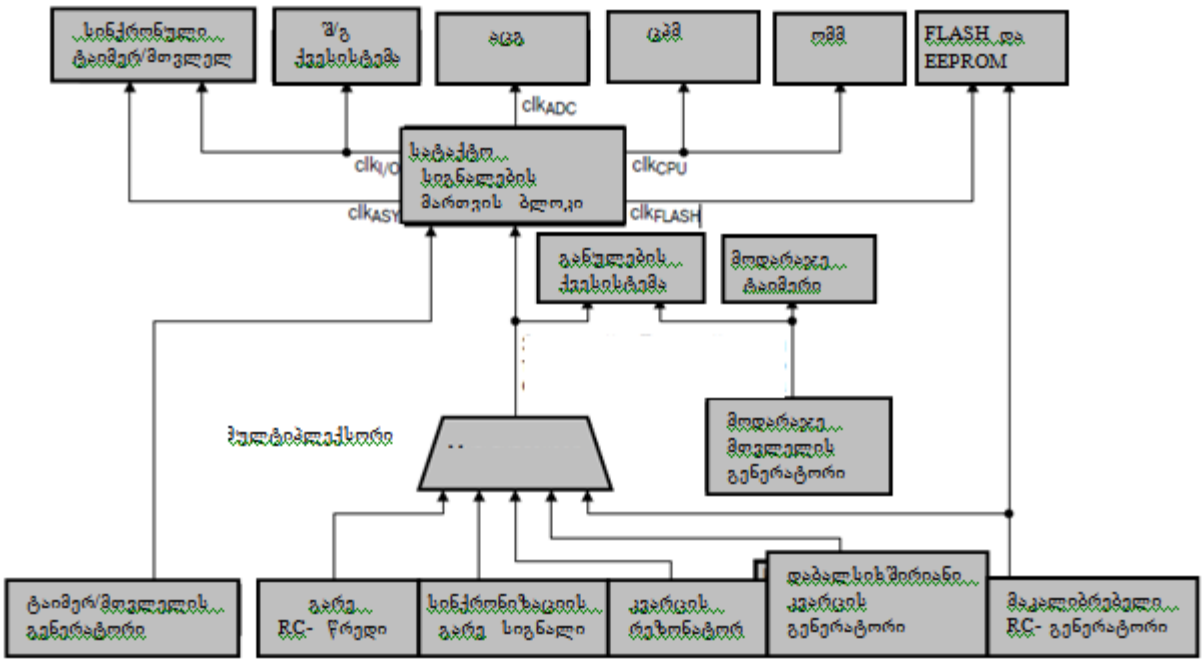
მიკროკონტროლერის ენერგომოხმარება მისი უმოქმედობის პერიოდში. ენერგო-მოხმარების შემცირების რეჟიმებში გადასვლა ხორციელდება SLEEP ბრძანებით. მიკროკონტროლერის "ძილის" რეჟიმიდან გამოსვლის დროს სრულდება მისი რეინიციალიზაცია (განულება).

მიკროკონტროლერის განულება ხორციელდება არა მარტო მისი "გაღვიძების", არამედ რიგი მოვლენების დადგომის შემთხვევაში. ამ მოვლენებს მიეკუთვნება: RESET (განულების) შესასვლელზე დაბალი დონის სიგნალის მიწოდება, მიკროკონტროლერის კვების ძაბვასთან ჩართვა, მოდარაჯე ტაიმერის ამოქმედება.

2.1. სატაქტო გენერატორი

მიკროკონტროლერის სინქრონიზაციის გამარტივებული სქემა ნაჩვენებია 2.1. სურათზე. როგორც სურათიდან ჩანს, სისტემური ტაქტური სიგნალის საფუძველზე ფორმირდება დამატებითი სიგნალები, რომლებიც გამოიყენება მიკროკონტროლერის მოდულებისა და ბლოკების ტაქტირებისათვის:

- Clk_{CP} - ცენტრალური პროცესორის სატაქტო სიგნალი, გამოიყენება მიკრო-კონტროლერის ბლოკების ტაქტირებისთვის, რომლებიც პასუხისმგებელია მიკროკონტროლერის ბირთვის მუშაობაზე (რეგისტრული ფაილი, მონაცემთა მეხსიერება და სხვა). ამ სიგნალის გამორთვის შემთხვევაში ცენტრალური პროცესორის მოწყობილობა (ცპმ) წყვეტს თავის მუშაობას, ყველა გამოთვლები ჩერდება;
- $Clk_{I/O}$ – შეტანა/გამოტანის ქვესისტემის სატაქტო სიგნალს იყენებს პერიფერიული მოწყობილობების უმრავლესობა, როგორიცაა ტაიმერ/მთვლელები და ინტერფეისული მოდულები;



სურ.2.1. მიკროკონტროლერის ტაქტირების სქემა

მუშაობის რეჟიმი	CKSEL 0…3
გენერატორ კვარცის ან კერამიკული რეზონატორით	1111.....1010
დაბალი სიხშირის კვარცის გენერატორი	1001
გენერატორი გარე RC წრედით	1000-0101
გენერატორი შიდა RC წრედით*	0100-0001
ტაქტირება გარე სინქრონიზაციის სიგნალით	0000
* რეჟიმი უთქმელობით	

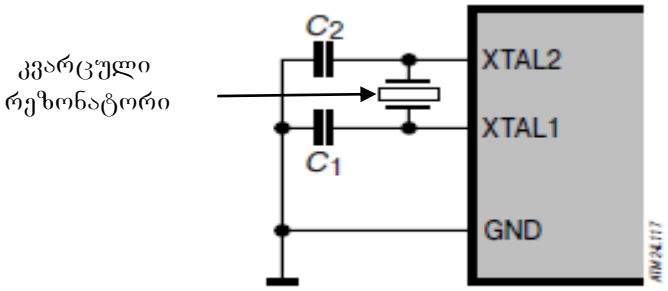
- Clk_{FLASH} – პროგრამის FLASH - მეხსიერების მართვის სატაქტო სიგნალი. როგორც წესი ეს სიგნალი ფორმირდება ცენტრალური პროცესორის სატაქტო სიგნალთან ერთად;
- Clk_{ASY} - ასინქრორული ტაიმერ/მთვლელის სატაქტო სიგნალი. ტაქტირება ხორციელდება უშუალოდ გარე კვარცის რეზონატორიდან.
- Clk_{ADC} - აცგ მოდულის სატაქტო სიგნალი. აღნიშნული სატაქტო სიგნალი საშუალებას იძლევა განხორციელდეს ანალოგური სიგნალების გარდაქმნა ციფრულ ფორმატში ცენტრალურ პროცესორის მოწყობილობისა და შეტანა/გამოტანის ქვესისტემების გაჩერების შემთხვევაში.

როგორც მოყვანილი 2.1.სურ.-დან ჩანს, მიკროკონტროლერის შემადგენლობაში შედის სხვდასხვა შესაძლებლობის მქონე რამდენიმე სატაქტო გენერატორი, რომელთაგან მიკროპროკონტროლერისადმი წაყენებული კონკრეტული პირობების შესაბამისად შეგვიძლია გამოვიყენოთ ერთ-ერთი. სატაქტო გენერატორის ამორჩევა ხდება CKSEL რეგისტრის 0-3 თანრიგებში გარკვეული კოდის ჩაწერით. თითოეული გენერატორის არჩევისათვის საჭირო კოდი ნაჩვენებია 2.1. ცხრილში.

ცხრილი 2.1. სატაქტო გენერატორის ამორჩევა

ტაქტური გენერატორი გარე რეზონატორით

რეზონატორი უერთდება მიკროკონტროლერის XTAL1 და XTAL2 გამომყვანებს, რომლებიც ნაჩვენებია 2.2. სურ-ზე. ეს გამომყვანებია შესაბამისად ტაქტური გენერატო-ის შესასვლელი და გამოსასვლელი.



სურ.2.2. კვარცული რეზონატორის მიერთება მიკროკონტროლერთან

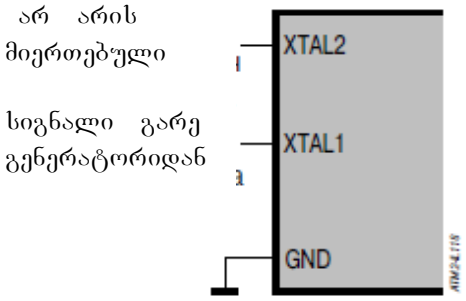
კვარცული რეზონატორი ძაბვის მიწოდების შემდეგ გამოსასვლელებზე აფორმირებს სინუსოიდურ ძაბვის სიგნალებს, რომლებიც მიეწოდება XTAL1, XTAL2 შესასვლელებით მიკროპროკონტროლერის შიგა სქემას, სადაც ფორმირდება სატაქტო სიგნალები. კვარცული რეზონატორები სხვადასხვა სიხშირისაა, ამიტომ შესაბამისი რეზონატორის შერჩევით შეგვიძლია მივიღოთ სატაქტო იმპულსების სასურველი სიხშირე.

დაბალსიხშირიანი კვარცული გენერატორი

რეჟიმი გათვალისწინებულია 32768 ჰციან კვარცული რეზონატორის "საათის კვარცის" სიხშირეზე გამოყენებისათვის. როგორც ყველა გარე რეზონატორები, იგი ჩაირთვება მიკროკონტროლერის XTAL1 და XTAL2 გამომყვანებთან.

ტაქტირება გარე სინქრონიზაციის სიგნალით

სიგნალი გარე წყაროდან მიეწოდება XTAL1 შესასვლელს, რომელიც ნაჩვენებია 2.3.სურ.-ზე.

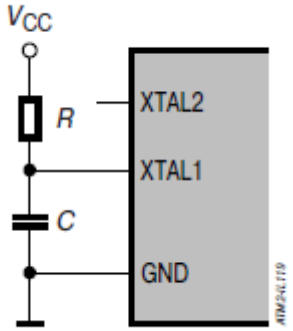


სურ.2.3. გარე სინქრონიზაციის სიგნალის მიერთება მიკროკონტროლერთან

გარე RC წრედი

იმ შემთხვევაში, როდესაც არ არის მოთხოვნა მაღალ დროის სიზუსტეზე, შეიძლება გამოვიყენოთ მარტივი RC გენერატორი. ამასთან გარე RC წრედი

მიერთებულია XTAL1 გამომყვანთან, როგორც ეს ნაჩვენებია 2.4. სურ-ზე. RC წრედში ფორმირდება იმპულსები კონდენსატორის დამუხტვა-განმუხტვის გზით. იმპულსების სიხშირე და ამპლიტუდა განისაზღვრება C კონდენსატორის და R რეზისტორის შერჩევით. წრედის კონდესატორის ტევადობა უნდა იყოს 22 პფ (min), რეზისტორის წინაღობა უნდა შეირჩეს 3.3.....100კომ დიაპაზონში. ტაქტური სიგნალის სიხშირე განისაზღვრება ფორმულით $f \approx 1/3$ RC. აღნიშნული სიგნალები მიეწოდება XTAL1 შესასვლელით მიკროპროცესორის შიგა სქემას, რომელიც გარდასახავს წრედიდან მიწოდებულ სიგნალებს მიკროკონტრო- ლერისთვის მისაღებ ფორმაში.



სურ.2.4. გარე RC წრედის მიერთება მიკროკონტროლერთან

ჩაშენებული გენერატორი შიგა RC წრედით

ეს გენერატორი ჩაშენებულია მიკროკონტროლერში RC წრედთან ერთად და გარე წრედების მიერთებას არ საჭიროებს.

შიგა RC გენერატორს შეუძლია იმუშაოს რამდენიმე ფიქსირებულ სიხშირეზე. გენერატორის მუშა სიხშირე განისაზღვრება CKSEL რეგისტრის 0..3 თანრიგებში

შესაბამისი კოდის ჩაწერით, რაც ნაჩვენებია 2.2. ცხრილში.

ცხრილი 2.2. შიგა RC გენერატორის სიხშირის ნუსხა

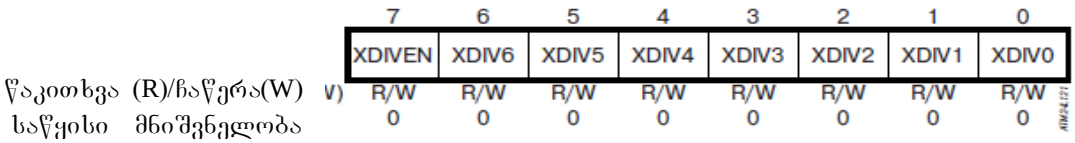
CKSEL 0…3	სიხშირე (მგჰც)
0001*	1.0
0010	2.0
0011	4.0
0100	8.0
* რეჟიმი უთქმელობით	

ტაქტური სიხშირის მართვა

მიკროკონტროლერ Atmega 128 აქვს შესაძლებლობა საჭიროების შემთხვევაში პროგრამულად შეამციროს სატაქტო გენერატორიდან მიღებული სიგნალების

სიხშირე. უდავოა, რომ სატაქტო სიხშირის შემცირება იწვევს Clk_{CPU}; $Clk_{I/O}$; Clk_{FLAH}; Clk_{ADC}, და ა.შ. სიგნალების სიხშირეების შემცირებას. ე.ი. მიკროკონტროლერის ყველა პერიფერიული მოწყობილობების მუშაობა შენელდება. სიხშირის დაყოფა ხდება სქემაში, რომელსაც წინაგამყოფს უწოდებენ.

მიკროკონტროლერ Atmega 128 ტაქტური სიგნალების წინაგამყოფის მართვისთვის განკუთვნილია შეტანა/გამოტანის XDIV რეგისტრი. ამ რეგისტრის ფორმატი ნაჩვენებია 2.5.სურ.-ზე.



სურ.2.5. XDIV რეგისტრის ფორმატი

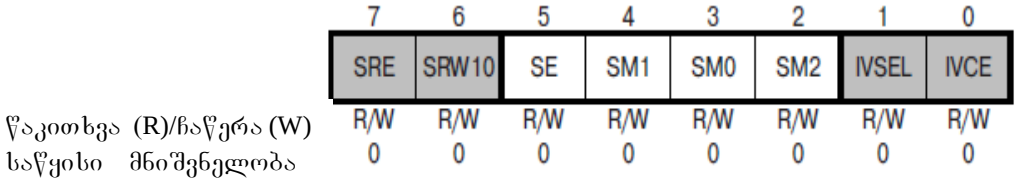
რეგისტრის უფროსი თანრიგი (XDIVEN) ემსახურება წინაგამყოფის ჩართვა/გამორთვას, ხოლო დანარჩენი თანრიგები (XDIV 0,,,6) განსაზღვრავს მიკრო-კონტროლერის ტაქტურ სიხშირეს. თუ აღვნიშნავთ XDIV0….6 თანრიგების შემცველობას d-თი, ტაქტური სიხშირის დამოკიდებულება ამ თანრიგების მდგომა-რეობასთან განისაზღვრება გამოსახულებით FCLK=გენერატორის სიხშირე /(129-d).

XDIV0…6 თანრიგების ცვლილება შესაძლებელია მხოლოდ XDIVEN თანრიგის განულებულ შემთხვევაში. თუ ეს თანრიგი ერთიანის მდგომარეობაშია, მაშინ მიკროკონტროლერის ტაქტური სიხშირე განისაზღვრება ზემოთ წარმოდგენილი გამოსახულებით. XDIVEN თანრიგის განულების შემთხვევაში, XDIV0..6 თანრიგების მნიშვნელობები იგნორირებული იქნება.

2.2. შემცირებული ენერგომოხმარების რეჟიმები

როგორც ზევით იყო აღნიშნული, ენერგიის დაზოგვის მიზნით მიკრო-კონტროლერში გათვალისწინებულია ე.წ. “ძილის” მდგომარეობაში გადასვლა, როდესაც ზოგიერთ მოწყობილობას შეუწყდება სატაქტო სიგნალების მიწოდება და ჩერდება. ამ მდგომარეობიდან გამოსვლა ხდება რაიმე მოვლენასთან დაკავშირებული წყვეტის სიგნალის მოსვლით, რის შედეგადაც მოწყობილობა გააგრძელებს მუშობას. მიკროკონტროლერში გამოიყენება ენერგოდაზოგვის რამდენიმე რეჟიმი, რომლებიც განსხვავდება მიკროკონტროლერის “ძილის” რეჟიმში გადასული პერიფერიული მოწყობილობების რაოდენობით.

“ძილის” რეჟიმის არჩევისათვის გამოიყენება MCUCR შეტანა/გამოტანის რეგისტრი, რომლის ფორმატი ნაჩვენებია 2.6.სურ-ზე რეჟიმის არჩევაში მონაწილეობს ოთხი თანრიგი, რომელთა დანიშნულება მოყვანილია 2.3. ცხრილში (სურათზე ეს თანრიგები აღნიშნულია ღია ფერით).



სურ.2.6. MCUCR რეგისტრის ფორმატი

ცხრილი2.3. MCUCR რეგისტრის თანრიგები "ძილის " რეჟიმის მართვისათვის

თანრიგის დანიშნულება	აღწერა
SE	**შემცირებულ ენერგომოხმარების რეჟიმში გადასვლის ნებადართვა.** ამ თანრიგის ერთიანში დაყენება ნებას რთავს მიკროკონტროლერს გადავიდეს შემცირებულ ენერგომოხმარების რეჟიმში. გადართვა ხორციელ- დება SLEEP ბრძანების მეშვეობით. თუ SE თანრიგი განულებულია ბრძანება არ სრულდება
SM2.....SM0	**შემცირებული ენერგომოხმარების რეჟიმის ამორჩევა.** ამ თანრიგების შემცველობა განსაზღვრავს, თუ რომელ რეჟიმში გადავა მიკრო- კონტროლერი SLEEP ბრძანების შესრულების შემდეგ.

ენერგომოხმარების შემცირების ნებისმიერ რეჟიმზე გადასვლა ხორციელდება SLEEP ბრძანების მეშვეობით. ამასთან SE ალამი ერთიანის მდგომარეობაში უნდა იყოს დაყენებული. მიკროკონტროლერის "ძილის" რეჟიმში გაუთვალისწინებული გადასვლის თავიდან ასაცილებლად რეკომენდებულია ეს ალამი დაყენებული იყოს SLEEP ბრძანების უშუალოდ შესრულების წინ. რეჟიმი, რომელზედაც გადადის მიკროკონტროლერი SLEEP ბრძანების შესრულების შემდეგ განისაზღვრება SM2.....SM0 თანრიგების მდგომარეობით.

შესაბამისობა ამ თანრიგების შემცველობასა და ენერგომოხმარების შემცირების რეჟიმებს შორის მოცემულია 2.4. ცხრილში.

ცხრილი 2.4. შემცირებული ენერგომოხმარების რეჟიმის ამორჩევა

SM2	SM1	SM0	რეჟიმები
0	0	0	Idle
0	0	1	AADC Noise Reduction
0	1	0	Power Down
0	1	1	Power save
1		0	დარეზერვირებულია
1	0	1	დარეზერვირებულია
1	1	0	Standby
1	1	1	Extended Standby*
* ეს რეჟიმი შესაძლებელია მხოლოდ კვარცის გენერატორის სატაქტო სიგნალების წყაროდ გამოყენების დროს			

“ძილის” რეჟიმიდან გამოსვლა სრულდება წყვეტის ან განულების შედეგად.

წყვეტის გენერაციის დროს მიკროკონტროლერი გადადის მუშა რეჟიმში, ასრულებს წყვეტის დამუშავების ქვეპროგრამას და მისი დასრულების შემდეგ განაახლებს პროგრამის შესრულებას. ენერგოდაზოგვის რეჟიმის შესრულების განმავლობაში შგრ, ომმ, სდრ შემცველობები არ იცვლება.

Idle (მოლოდინის რეჟიმი)

Idle რეჟიმში **Clk_{CPU}** და Clk_{FLASH} სატაქტო სიგნალების ფორმირება არ ხდება. მიკრო-კონტროლერის ცენტრალური პროცესორის მოწყობილობა ჩერდება, ხოლო ყველა დანარჩენი პერიფერიული მოწყობილობები (ინტერფეისული მოდული, ტაიმერ/მთვლელი, ანალოგური კომპარატორი, აცგ, მოდარაჯე ტაიმერი), ასევე წყვეტის ქვესისტემა, აგრძელებს ფუნქციონირებას. ამიტომ Idle რეჟიმიდან გამოსვლა შესაძლებელია როგორც გარე, ასევე შიგა წყვეტით. თუ ნებადართულია აცგ-ს მუშაობა, მაშინ გარდამ- სახი დაიწყებს გარდაქმნას “ძილის” რეჟიმში გადასვლისთანავე.

Idle რეჟიმის ძირითადი უპირატესობა არის ის, რომ იგი სწრაფ რეაგირებას ახდენს მოვლენებზე, რომლებიც იწვევს მიკროკონტროლერის “გამოღვიძებას”. სხვა სიტყვებით რომ ვთქვათ პროგრამის შესრულება იწყება მაშინვე, როდესაც Idle რეჟიმიდან გადასვლა ხდება მუშა რეჟიმში.

ADC Noise Reduction (აცგ ხმაურის შემცირების რეჟიმი)

მოცემული რეჟიმის გამოყენება შესაძლებელია იმ მოდელებში, რომლის შემადგენლობშიაც აცგ მოდულია. ამ რეჟიმში მიკროკონტროლერის ცენტრალური პროცესორული მოწყობილობა და შეტანა/გამოტანის ქვესისტემა წყვეტს მუშაობას (გამოირთვება სატაქტო სიგნალები clk_{cpu}, clk_{FLASH} და $clk_{I/O}$), ხოლო აცგ, გარე წყვეტის დამუშავების ქვესისტემა, მოდარაჯე ტაიმერი და TWI მოდულის მისამართის შედარების ბლოკი აგრძელებს მუშაობას. აქედან გამომდინარე აცგ შესასვლელებზე მცირდება ხელშეშლა, რომელსაც იწვევს მიკროკონტროლერის შეტანა/გამოტანის სისტემის მუშაობა, რაც თავის მხრივ გვაძლევს საშუალებას გავზარდოთ გარდაქმნის სიზუსტე. თუ აცგ ჩართულია მაშინ გარდაქმნა იწყება უშუალოდ ამ “ძილის” რეჟიმში გადასვლის შემდეგ.

მიკროკონტროლერის დაბრუნება მუშა მდგომარეობაში შეიძლება განხორციელდეს განულების შედეგად (აპარატურულად, მოდარაჯე ტაიმერიდან, კვების ძაბვის შემცირების აღმომჩენი BOD სქემიდან) ან ზოგიერთი წყვეტის სიგნალის გენერაციის შედეგად.

Pover Down (მიკრომოხმარების რეჟიმი)

Pover Down რეჟიმი გამორთავს ყველა შიგა ტაქტურ სიგნალს, შესაბამისად ფუნქციონირებას წყვეტს მიკროკონტროლერის ყველა სისტემა, რომლებიც სინქრონულ რეჟიმში მუშაობს. იმის გამო, რომ მიკროკონტროლერის სატაქტო გენერატორი წყვეტს მუშაობას **Pover Down** რეჟიმში, მიკროკონტროლერის "გამოღვიძების" გამომწვევი მოვლენასა და მის მუშაობის განახლებას შორის საჭიროა გარკვეული დრო, რომლის განმავლობაშიც მიკროპროცესორის სატაქტო გენერატორი გადის სამუშაო რეჟიმზე.

Pover Down რეჟიმიდან გამოსვლა შესაძლებელია ან განულების შედეგად (აპარატურულად, მოდარაჯე ტაიმერიდან), ანდა გარე წყვეტის გენერაციის შედეგად.

Pover save (ეკონომიური რეჟიმი)

ეს რეჟიმი იდენტურია **Pover Down** რეჟიმის, მხოლოდ ერთი გამონაკლისით: თუ მიკროკონტროლერის ტაიმერ/მთვლელი, რომელიც მხარს უჭერს ე.წ. ასინქრონულ რეჟიმში მუშაობას (მისი ტაქტირება სრულდება ცალკე გენერატორიდან), მაშინ ის იმუშავებს მიკროკონტროლერის "ძილის" რეჟიმშიც. ამიტომ, **Pover save** რეჟიმიდან გამოსვლა შესაძლებელია არა მხოლოდ **Pover Down** რეჟიმში განხილული ხდომილების შედეგად, ასევე ტაიმერ/მთვლელიდან წყვეტის სიგნალითაც, რა თქმა უნდა იმ შემთხვევაში, თუ კი წყვეტა ნებადართულია.

Standby

Standby რეჟიმი მთლიანად იდენტურია **Pover save** რეჟიმის, იმ განსხვავებით რომ ტაქტური გენერატორი აგრძელებს ფუნქციონირებას.

Extended Standby (მოლოდინის გაფართოებული რეჟიმი)

Extended Standby რეჟიმი **Standby** რეჟიმის ანალოგიურია.

2.3. მიკროკონტროლერის განულება

მიკროკონტროლერის რეინციალიზაციას ანუ ე.წ. "განულება"-ს გადაჰყავს მიკროკონტროლერი საწყის მდგომარეობაში. განულება შეიძლება შესრულდეს შემდეგი მიზეზით:

- მიკროკონტროლერთან ძაბვის მიერთება;
- დაბალი პოტენციალის მიწოდება RESET გამომყვანზე - აპარტატული განულება;
- მოდარაჯე მთვლელისაგან;
- კვების ძაბვის შემცირება განსაზღვრული სიდიდეზე ქვევით;
- JTAG ინტერფეისით მიწოდებული ბრძანების საშუალებით.

ნებისმიერი ზემოთ ჩამოთვლილი მიზეზის წარმოშობის შემთხვევაში შეტანა/გამოტანის ყველა რეგისტრში ჩაიწერება მათი საწყისი მნიშვნელობები, ხოლო ბრძანების მთვლელში ჩაიწერება განულების ვექტორის მისამართი. ამ მისამართზე

იმყოფება სამომხმარებლო პროგრამაზე გადასვლის ბრძანება JMP. თუ პროგრამაში წყვეტა არ გამოიყენება, მაშინ პროგრამის შესრულება დაიწყება განულების ვექტორის მისამართიდან. განულების ვექტორი განთავსებულია პროგრამული მეხსიერების დასაწყისში $0000 მისამართზე.

მიკროკონტროლერის განულების სქემის ლოგიკა შემდეგია: მოვლენის დადგომის დროს, რომელსაც მივყავართ მიკროკონტროლერის განულებამდე, ფორმირდება შიგა განულების სიგნალი. ერთდროულად გაიშვება განულების დაყოვნების ფორმირების ტაიმერი. განსაზღრული დროის შუალედის გასვლის შემდეგ შიგა განულების სიგნალი მოიხსნება და იწყება პროგრამის შესრულება. მიკროკონტროლერი გვაძლევს საშუალებას განვსაზღვროთ მიზეზი, რის შედეგაც განხორციელდა მიკროკონტროლერის განულება. აღნიშნული მდგომარეობის განსაზღვრისათვის გამოიყენება მიკროკონტროლერის მართვისა და მდგომარეობის **MCUCSR** რეგისტრი. ამ რეგისტრში განთავსებულია ალმების ნაკრები, რომელთა მდგომარეობა დამოკიდებულია იმ მოვლენაზე, რომელმაც გამოიწვია მოწყობილობის განულება. MCUCSR რეგისტრის ფორმატი ნაჩვენებია 2.7,სურ-ზე (თანრიგები, რომლებიც არ მიეკუთვნება განულების ქვესისტემას, გამოყოფილია რუხი ფერით). ალმების აღწერა მოყვანილია 2.5. ცხრილში

	7	6	5	4	3	2	1	0
	X	X	X	JTRF	WDRF	BORF	EXTRF	PORF
წაკითხვა (R)/ჩაწერა (W)	X	X	X	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობები	0	0	0	0	0	0	0	0

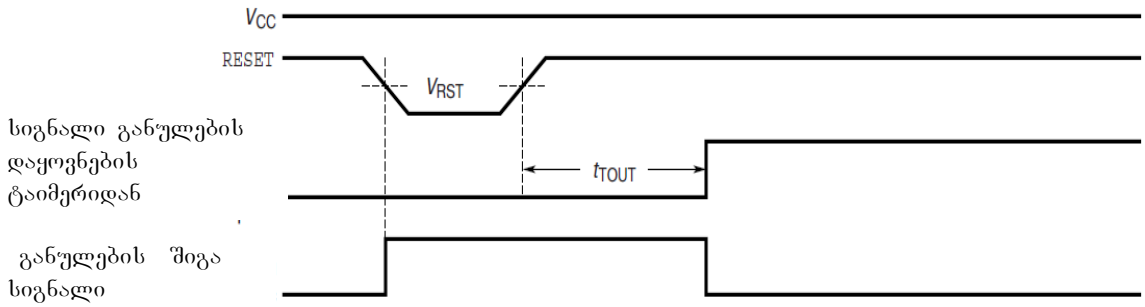
სურ..2.7. CUCSR რეგისტრის ფორმატი

განულება კვების ჩართვის დროს

მიკროკონტროლერის შემადგენლობაში შედის კვების ჩართვის დროს განულების სქემა POR (Power on Reset). ეს სქემა მიკროკონტროლერს იჭერს განულების მდგომარეობაში, სანამ კვების ძაბვა არ გადააჭარბებს რომელიღაც V_{POT} ზღვრულ მნიშვნელობას. როდესაც კვების ძაბვა აღწევს V_{POT} მნიშვნელობას POR სქემა ჩართავს განულების დაყოვნების ტაიმერს. იგი ითვლის t_{TOU} დაყოვნების დროს, რომლის დასრულების შემდეგ შიგა განულების სიგნალი მოიხსნება და მოხდება მიკროკონტროლერის გაშვება.

აპარატურული განულება

მიკროკონტროლერის აპარატურული (ანუ გარე) განულება ხორციელდება RESET გამომყვანზე დაბალი დონის სიგნალის მიწოდებით. მიკროპროცესორი რჩება განულების მდგომარეობაში მანამ, სანამ RESET გამომყვანზე არის დაბალი დონის სიგნალი. როდესაც RESET გამომყვანზე ძაბვა მიაღწევს თავის ზღრულ V_{RST} მნიშვნელობას, ჩაირთვება განულების დაყოვნების ტაიმერი. t_{TOUT} დაყოვნების შემდეგ შიგა განულების სიგნალი მოიხსნება და მოხდება მიკროკონტროლერის გაშვება. აპარატურული განულების დროითი დიაგრამა ნაჩვენებია 2.8.სურ-ზე.



სურ. 2.8. აპარატურული განულების დროითი დიაგრამა

ცხრილი 2.5. განულების წყაროების ალმების **CUCSR** რეგისტი

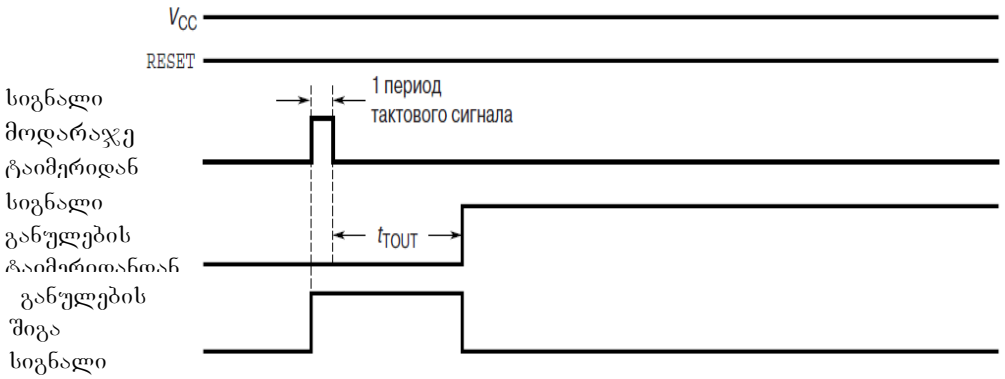
ალმების დანიშნულება	აღწერა
JTRF	**JTAG – განულების ალამი.** ერთიანში დაყენება ხდება, **JTAG** ინტერფეისით მიწოდებული "AVR RESET" ბრძანებით. თანრიგის განულება ხორციელდება კვების გამორთვით ან უშუალოდ ლოგიკური ნულის ჩაწერით ამ თანრიგში.
WDRF	**განულების ალამი მოდარაჯე ტაიმერიდან.** ერთიანში დგება თუ განულების წყარო მოდარაჯე ტაიმერია. თანრიგის განულება ხორციელდება კვების გამორთვის ან უშუალოდ ლოგიკური ნულის ჩაწერით ამ თანრიგში.
BORF	**განულების ალამი ძაბვის შემცირების გამო.** ერთიანში დგება BOD სქემიდან. თანრიგის განულება ხორციელდება კვების გამორთვის შემთხვევაში ან უშუალოდ ლოგიკური ნულის ჩაწერით ამ თანრიგში.
EXRTF	**აპარატურული განულების ალამი .** დგება ერთიანში, როცა განულება განხორციელდება განულების RESET გამომყვანზე დაბალი დონის სიგნალის მიწოდების დროს. თანრიგი განულდება კვების გამორთვის შემთხვევაში ან უშუალოდ ლოგიკური ნულის ჩაწერით ამ თანრიგში.
PORF	**განულების ალამი კვების ჩართვისას.** დგება ერთიანში ,როდესაც მიკროკონტროლერს მიეწოდება კვება. თანრიგის განულება ხორციელდება მხოლოდ უშუალოდ ლოგიკური ნულის ჩაწერით ამ თანრიგში.

შენიშვნა: BOD –ძაბვის შემცირების აღმომჩენი სქემა

განულება მოდარაჯე ტაიმერიდან

მოდარაჯე ტაიმერიდან (თუ ის ჩართულია) გენერირდება განულების მოკლე დადებითი იმპულსი, რომლის ხანგრძლივობა მიკროკონტროლერის ტაქტური სიგნალის ერთი პერიოდის ტოლია. აღნიშნული იმპულსის ფრონტის კლებადობის დროს ფორმირდება განულების შიგა სიგნალი და ჩაირთვება განულების დაყოვნების ტაიმერი. T_{TOUT} დაყოვნების შემდეგ განულების შიგა სიგნალი მოიხსნება და ხორციელ

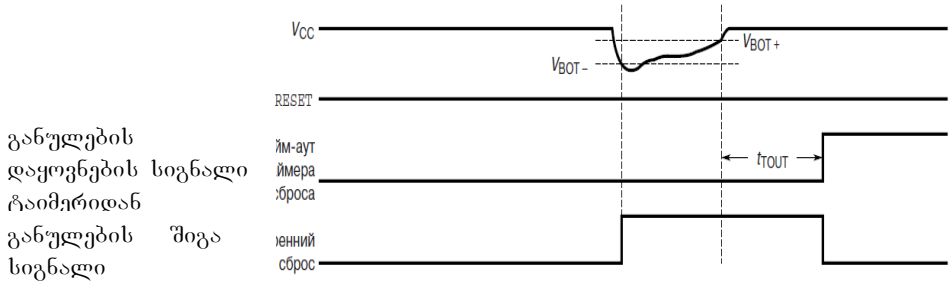
დება მიკროკონტროლერის გაშვება. მოდარაჯე ტაიმერიდან განულების დროითი დიაგრამა ნაჩვენებია 2.9.სურ.-ზე.



სურ.2.9. მოდარაჯე ტაიმერიდან განულების დროითი დიაგრამა

განულება კვების ძაბვის შემცირების დროს

მიკროკონტროლერ Atmega 128 თავის შემადგენლობაში გააჩნია BOD (Brown Out Detection) სქემა , რომელიც თვალყურს ადევნებს კვების წყაროს ძაბვის სიდიდეს. თუ ამ სქემის მუშაობა ნებადართულია ,მაშინ კვების ძაბვის შემცირების დროს რომელიღაც ნიშნულზე დაბლა, მას გადაჰყავს მიკროკონტროლერი განულე- ბის მდგომარეობაში. ხოლო როდესაც კვების ძაბვა კვლავ გაიზრდება ზღვრულ მნიშვნელობამდე, გაიშვება განულების დაყოვნების ტაიმერი. t_{TOUT} დაყოვნების ფორმირების შემდეგ შიგა დაყოვნების სიგნალი მოიხსნება და ხორციელდება მიკროკონტროლერის გაშვება. დროითი დიაგრამა, რომელიც შეესაბამება BOD სქემიდან განულების პროცესს ნაჩვენებია 2.10.სურ.-ზე.



სურ.2.10. BOD სქემიდან განულების დროითი დიაგრამა

BOD სქემის მართვისათვის გამოიყენება ორი მაკონფიგურებელი უჯრედი. სქემის ჩართვა/გამორთვას მართავს მაკონფიგურებელი უჯრედი BODEN სქემის მუშაობის ნებართვისათვის ეს უჯრედი უნდა იყოს დაპროგრამებული (ჩაიწეროს “0”). ამოქმედების ზღვარი განისაზღვრება მაკონფიგურებელი უჯრედით BODLEVEL. თუ ამ უჯრედში ჩაწერილია ლოგიკური ერთიანი, მაშინ ამოქმედების ზღვარი ტოლია 2,7 ვ. ხოლო თუ მასში ჩაწერილია ლოგიკური “0” (მისი დაპროგრამების შემდეგ), ამოქმედე-

ბის ზღვარი ტოლია 4,0 ვ. სქემის ამოქმედება განხორციელდება მხოლოდ იმ შემთხვევაში, როცა კვების ძაბვის ჩავარდნის ხანგრძლივობა მეტი იქნება რომელიღაც სიდიდეზე. ჩავარდნის ხანგრძლივობის მინიმალური მნიშვნელობა 2 მკწმ სქემის ნებისმიერი რეჟიმის მუშაობის დროს.

განულების სქემის მართვა

განულების სქემის მართვა მდგომარეობს t_{TOUT} განულების სიგნალის დაყოვნების ხანგრძლივობის დაყენებაზე. ამისათვის გამოიყენება იგივე კონფიგურაციის უჯრედები, რომლებიც განსაზღვრავს მიკროკონტროლერის ტაქტური გენერატორის მუშაობის რეჟიმს.

თავი 3

წყვეტა

წყვეტას უწოდებენ მოქმედებას, რომლის დროსაც მიკროკონტროლერი წყვეტს პროგრამის ნორმალურ მიმდინარეობას პრიორიტეტული ამოცანის შესრულების მიზნით, რომელიც განისაზღვრება მიკროკონტროლერის შიგა ან გარე მოვლენების საფუძველზე. წყვეტის გამომწვევი მოვლენის დადგომის დროს ფორმირდება წყვეტის მოთხოვნის სიგნალი, რის საფუძველზე მიკროკონტროლერი ინახავს სტეკში ბრძანების მთვლელის შემცველობას (ანუ პროგრამის მომდევნო ბრძანების მისამართს, რომლის წინაც მოხდა წყვეტა) და ჩატვირთავს მასში შესაბამის წყვეტის ვექტორის მისამართს. ამ მისამართზე, როგორც წესი, განთავსებულია წყვეტის დამუშავების ქვეპროგრამაზე უპირობო გადასვლის ბრძანება. ეს ბრძანება თავის მხრივ შეიცავს აღნიშნული წყვეტის პროგრამის საწყის მისამართს, რომელიც ჩაიწერება ბრძანების მთვლელში. შედეგად მიკროკონტროლერი დაიწყებს წყვეტის პროგრამის შესრულებას. წყვეტის დამუშავების პროგრამაში ბოლო ბრძანება უნდა იყოს სტეკიდან მისამართის დაბრუნების ბრძანება RETI, რომელიც აღადგენს ადრე შენახულ ბრძანების მთვლელის შემცველობას და უზრუნველყოფს ძირითად პროგრამაში დაბრუნებას.

რამდენადაც წყვეტის წყაროს წარმოადგენენ მიკროკონტროლერის სხვადასხვა პერიფერიული მოწყობილობები, იმდენად წყვეტის რაოდენობას განსაზღვრავს კონკრეტული მოდელი. Atmega 128-ში შესაძლებელია გამოყენებული იყოს 34 სახვადასხვა წყვეტა.

3.1.წყვეტის ვექტორების ცხრილი

მიკროკონტროლერ Atmega 128-ს გააჩნია პრიორიტეტული წყვეტის მრავალდონიანი სისტემა. პროგრამის მეხსიერების უმცროსი მისამართები, დაწყებული $0002 მისამართიდან, განკუთვნილია წყვეტის ვექტორის ცხრილისათვის. ყოველ წყვეტას ამ ცხრილიდან შეესაბამება მისამართი, რომელიც ჩაიტვირთება ბრძანებათა მთვლელში წყვეტის წარმოქმნის შემთხვევაში. ცხრილში ვექტორის მდგომარეობა ასევე განსაზღვრავს შესაბამისი წყვეტის პრიორიტეტს: რაც მცირეა მისამართი მით მაღალია წყვეტის პრიორიტეტი (მისი მომსახურება შესრულდება პირველ რიგში). წყვეტის ვექტორის ზომა 2 ბაიტია. წყვეტის ქვეპროგრამების დამუშავებაზე გადასვლისათვის გამოიყენება JMP ბრძანება. ქვევით წარმოდგენილ 3.1. ცხრილში მოყვანილია მიკროკონტროლერ Atmega 128 წყვეტის ვექტორების ჩამონათვალი.

თუ მიკროკონტროლერის მუშაობის დროს წყვეტა არ არის გათვალისწინებული, მაშინ წყვეტის ვექტორის ცხრილების ადგილზე შესაძლებელია განთავსებული იყოს ძირითადი პროგრამის ნაწილი.

3.2.წყვეტის დამუშავება

წყვეტის გლობალური ნებადართვის/აკრძალვისათვის განკუთვნილია SREG რეგისტრის I ალამი. წყვეტის ნებადართვის დროს ის უნდა იყოს დაყენებული ერთიანის მდგომარეობაში, ხოლო აკრძალვის დროს - განულებულ მდგომარეობაში. წყვეტის ინდივიდუალური ნებადართვა ან აკრძალვა (შენიღბვა) ხორციელდება

ცალკეულ მოდულებში წყვეტის ნიღბის რეგისტრის შესაბამისი თანრიგის დაყენება/განულებით, რომელიც განიხილება ქვევით.

ცხრილი 3.1. Atmega 128 მიკროკონტროლერის წყვეტის ვექტორების ცხრილი

წყარო	აღწერა	მისამართი
INT0	გარე წყვეტა 0	$0002
INT1	გარე წყვეტა 1	$0004
INT2	გარე წყვეტა 2	$0006
INT3	გარე წყვეტა 3	$0008
INT4	გარე წყვეტა 4	$000A
INT5	გარე წყვეტა 5	$000C
INT6	გარე წყვეტა 6	$000E
INT7	გარე წყვეტა 7	$0010
TIMER2COMP	T2 ტაიმერ/მთვლელ-ის თანხვდენა	$0012
TIMER2OVR	T2 ტაიმერ/მთვლელ-ის გადავსება	$0014
TIMER1CAPT	T1 ტაიმერ/მთვლელ-ის დაპყრობა	$0016
TIMER1COMPA	T1 ტაიმერ/მთვლელის თანხვდენა “A ”	$0018
TIMER1COMPB	T1 ტაიმერ/მთვლელ-ის თანხვდენა “B ”	$001A
TIMER1OVR	T1 ტაიმერ/მთვლელ-ის გადავსება	$001C
TIMER0 COMP	T0 ტაიმერ/მთვლელ-ის თანხვდენა	$001 E
TIMER0 OVR	T0 ტაიმერ/მთვლელ-ის გადავსება	$0020
SPI,STC	SPI- ო გადაცემა დასრულებულია	$0022
USART0,RX	USART0, მიღება დასრულებულია	$0024
USAT0,UDRE	USART0, მონაცემების რეგისტრი ცარიელია	$0026
USAT0.TX	USART1, გადაცემა დასრულებულია	$0028
ADC	აცგ გარდაქმნა დასრულებულია	$002A
EE _RDY	EEPROM,მზადყოფნაშია	$002C
ANA _COMP	ანალოგური კომპარატორი	$002E
TIMER1COMPC	T1 ტაიმერ/მთვლელ-ის თანხვდომა “C”	$0030
TIMER3CAPT	T3 ტაიმერ/მთვლელ-ის დაპყრობა	$0032
TIMER3CAPTA	T3 ტაიმერ/მთვლელ-ის თანხვდომა “A”	$0034
TIMER3CAPTB	T3 ტაიმერ/მთვლელ-ის თანხვდომა “B”	$0036
TIMER3CAPTC	T3 ტაიმერ/მთვლელ-ის თანხვდომა “C”	$0038
TIMER3OVF	T3 ტაიმერ/მტვლელი-ის გადავსება	$003A
USART1,RX	USART1 ,გადაგზავნა დასრულებულია	$003C
USART1,UDRE	USART0, მონაცემების რეგისტრი ცარიელია	$003E
USART1,TX	USART1, გადაცემა დასრულებულია	$0040
TWI	გადავსება TWI მოდულიდან	$0042
SPM_RDY	მზადყოფნა SPM	$0044

წყვეტის წარმოქმნის დროს SREG რეგისტრის I ალამი აპარატურულად განულდება, რითაც აიკრძალება მომდევნო წყვეტის დამუშავება. მაგრამ წყვეტის დამუშავების ქვეპროგრამაში ეს ალამი შეიძლება ხელახლა დაყენდეს ერთიანის მდგომარეობაში ჩართული წყვეტის ნების დართვისათვის. წყვეტის დამუშავების ქვეპროგრამიდან დაბრუნების შემთხვევაში (RETI ბრძანების შესრულების დროს) I ალამი დაყენდება აპარატურულად.

ყველა არსებული წყვეტები შესაძლებელია დაიყოს ორ ტიპად: პირველი ტიპის წყვეტა გენერირდება ზოგიერთი მოვლენის დადგომის დროს, რომლის შედეგადაც ხდება წყვეტის ალმის დაყენება. იმ შემთხვევაში თუ წყვეტა ნებადართულია, მაშინ ბრძანებათა მთვლელში ჩაიტვირთება შესაბამისი წყვეტის ვექტორის მისამართი. ამავე

დროს განულდება წყვეტის ალამი აპარატურულად. წყვეტის ალმის განულება ასევე შეიძლება განხორციელდეს პროგრამულად ლოგიკური ერთიანის ჩაწერით რეგისტრის შესაბამისი ალმის თანრიგში.

საჭიროა გვახსოვდეს, რომ წყვეტის დამუშავების პროგრამის გამოძახების დროს SREG მდგომარეობის რეგისტრის შემცველობა არ შეინახება. ამიტომ მომხმარებელმა დამოუკიდებლად უნდა შეინახოს ამ რეგისტრის შემცველობა წყვეტის დამუშავების პროგრამაში გადასვლის დროს (საჭიროების შემთხვევაში) და აღადგინოს მისი მნიშვნელობა RETI ბრძანების გამოძახების წინ.

თუ წყვეტების საერთო აკრძალვის შემთხვევაში გენერირდა ერთი ან რამდენიმე წყვეტის მოთხოვნა, ხდება მათი დაფიქსირება შესაბამის ალმების დაყენებით, იმ მიზნით, რომ წყვეტის ნების დართვის შემდეგ წყვეტები შესრულდეს მათი პრიორიტეტის მიხედვით.

მიკროკონტროლერში გამოყენებულია წყვეტების მომსახურების რიგითობის პრინციპი რამდენიმე წყვეტის მოთხოვნის ერთდროული წარმოშობის შემთხვევაში. ამ მიზნით წყვეტების მოთხოვნებს ენიჭებათ მომსახურების სხვადასხვა პრიორიტეტი. უფრო მაღალი პრიორიტეტის მქონე მოთხოვნა მომსახურებული იქნება ნაკლები პრიორიტეტის მოთხოვნის წინ.

წყვეტის მოთხოვნები შეიძლება დაგენერირდეს გარე სქემებში და მიეწოდოს მიკროკონტროლერს გამომყვანების საშუალებით ან მიკროკონტროლერის შიგნით არსებულ ბლოკებში. ყოველ მოთხოვნისათვის გამოყოფილია თანრიგები (წყვეტის ალმები), რომლებშიც ფიქსირდება წყვეტის მოთხოვნები. წინამდებარე თავში განიხილება გარე წყვეტა, შიგა წყვეტები კი განხილული იქნება შემდგომ თავებში, ცალკეული მოდულების შესწავლის დროს.

3.3.გარე წყვეტა

გარე წყვეტის მოთხოვნა მიეწოდება მიკროკონტროლერს სხვადასხვა გარე სქემიდან რვა გამომყვანის საშუალებით, რომლებიც აღინიშნება INT0-INT7-თ.

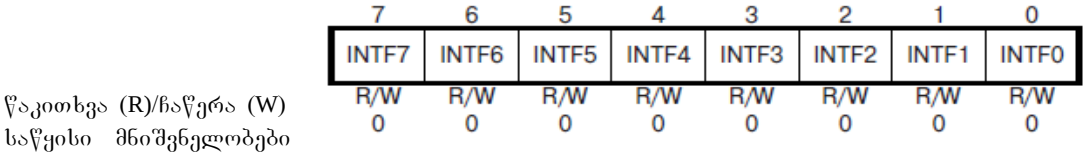
Atmega128 გარე წყვეტის ნებადართვის/აკრძალვისათვის განკუთვნილია EIMSK (External Interrupt MaSK – გარე წყვეტების ნიღბები) ნიღბების რეგისტრი, რომელიც შედის შეტანა/გამოტანის რეგისტრების შემადგენლობაში. ამ რეგისტრის ფორმატი ნაჩვენებია 3.1.სურ.-ზე.

	7	6	5	4	3	2	1	0
	INT7	INT6	INT5	INT4	INT3	INT2	INT1	INT0
წაკითხვა (R)/ჩაწერა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	0	0	0	0	0	0	0	0

სურ.3.1. EIMSK რეგისტრის ფორმატი

რეგისტრის ყოველი INT_n თანრიგი ასრულებს იმ წყვეტის ნების დართვას/აკრძალვას, რომლის ნომერი შეესაბამება თანრიგის ნომერს. თუ რეგისტრის n-ური თანრიგში ჩაწერილია ერთიანი და SREG I ალამი ერთიანის მდგომარეობაშია, მაშინ INTn გამომყვანებიდან წყვეტა ნებადართულია.

Atmega128 გარე წყვეტის ინდიკაციისათვის განკუთვნილია EIFR (External Interrupt Flag Register-გარე წყვეტების ალმების რეგისტრი). ამ რეგისტრის ფორმატი ნაჩვენრბია 3.2.სურ.-ზე.



სურ.3.2. EIFR რეგისტრის ფორმატი

მიკროკონტროლერის INT_n გამოსასვლელზე წყვეტის მოთხოვნის შემთხვევაში შესაბამისი ალამი $INTF_n$ (რეგისტრის n-ური თანრიგი) დაყენდება ერთიანის მდგომარეობაში. ალამი ჩამოიგდება აპარატურულად წყვეტის ქვეპროგრამის დამუშავების დროს ან მასში ლოგიკური ერთიანის ჩაწერით,

თავი 4

შეტანა/გამოტანის პორტები

გარე მოწყობილობებთან კავშირისათვის მიკროკონტროლერი შეიცავს შეტანა/გამოტანის პორტებს.

მიკროკონტროლერის თითოეულ პორტს აქვს განსაზღვრული რაოდენობის გამომყვანები, რომლებიც უზრუნველყოფს ციფრული სიგნალების მიღებას და გადაცემას. მონაცემთა გადაცემის მიმართულების განსაზღვრა შეტანა/გამოტანის ნებისმიერი კონტაქტის გავლით შეიძლება განხორციელდეს დროის ნებისმიერ მომენტში პროგრამულად.

Atmega 128 აქვს ექვსი (A,B,C,D,E,F) 8 თანრიგიანი შეტანა/გამოტანის პორტი და ერთი ხუთ თანრიგიანი შეტანა/გამოტანის G პორტი. სულ შეტანა/გამოტანის კონტაქტების რაოდენობა 53.

შეტანა/გამოტანის პორტების რეგისტრები

თითოეული პორტი შეიცავს სამ შეტანა/გამოტანის რეგისტრს, რომელთა საშუალებით ხორციელდება პორტებთან წვდომა: PORTx პორტის მონაცემის რეგისტრი, საიდანაც გაიცემა წინასწარ ჩაწერილი მონაცემები პორტის გამომყვანებზე; DDRx გადაცემის მიმართულების რეგისტრი, რომლის მდგომარეობა განსაზღვრავს გადაცემის მიმართულებას; PINx პორტის შეტანის რეგისტრი, სადაც იწერება გამომყვანებიდან მიღებული მონაცემები. რეგისტრის ნამდვილი დასახელება მიიღება <x>-ის ნაცვლად პორტის დასახელების ჩასმით. შესაბამისად, A პორტისათვის - PORTA, DDRA, PINA, B პორტისათვის – PORTB, DDRB, PINB. და ასე შემდეგ. რამდენადაც PINx რეგისტრის დახმარებით ხორციელდება წვდომა პორტის გამომყვანების სიგნალების ფიზიკურ მნიშვნელობებთან, მისგან შესაძლებელია მხოლოდ წაკითხვა. დანარჩენი ორ რეგისტრში შესაძლებელია როგორც ჩაწერა, ასევე წაკითხვა.

შეტანა/გამოტანის პორტების კონფიგურირება

თითოეულ პორტის გამოსასვლელს შეესაბამება თითო თანრიგი შეტანა/გამოტანის სამი რეგისტრიდან: PORTxn(PORTx), DDRxn(DDRx) და PINxn (PINx). რეგისტრის თანრიგების დასახელება მიიღება <n> სიმბოლოს მაგივრად თანრიგის ნომრის ჩასმით. მაგალითად, A პორტის 0-თანრიგი - როგორც PORTA0. პორტის გამომყვანის რიგითი ნომერი შეესაბამება ამ პორტის რეგისტრის თანრიგის რიგით ნომერს. როდესაც პორტის თანრიგების რაოდენობა რვაზე ნაკლებია, ისინი იწერებიან უმცროს თანრიგებში. რეგისტრების გამოუყენებელი უფროსი თანრიგები შეღწევადია მხოლოდ წაკითხვისათვის და მასში ჩაწერილია “0”.

DDRx რეგისტრის DDRxn თანრიგი განსაზღვრავს მონაცემების გადაცემის მიმართულებას შეტანა/გამოტანის კონტაქტების გავლით. თუ ეს თანრიგი დაყენებულია “ერთიანის “ მდგომარეობაში, მაშინ პორტის n-ური გამომყვანი არის გამოსასვლელი, ხოლო როცა დაყენებულია “ნულში”, მაშინ -შესასვლელი.

თუ გამომყვანი ფუნქციონირებს როგორც გამოსასვლელი (DDxn=1), ეს თანრიგი განსაზღვრავს პორტის n-ური გამოსასვლელის მდგომარეობას (მონაცემების გამოტანას). როცა გამომყვანი რეგისტრის PORTxn თანრიგი "ერთიანშია", გამომყვანზე იქნება მაღალი დონის ძაბვა, "ნულიანის" შემთხვევაში გამომყვანზე იქნება დაბალი დონის ძაბვა.

როცა გამომყვანი ფუნქციონირებს როგორც შესასვლელი (DDRxn=0), მონაცემები შეიტანება PINxn რეგისტრის თანრიგში პროგრამის მიერ შემდგომი წაკითხვისათვის.

თავი 5

ტაიმერ / მთვლელები

საერთო ცნობები

მიკროკონტროლერი Atmega 128 შეიცავს საერთო დანიშნულების ოთხ ტაიმერ/მთვლელს, რომლებიც აღინიშნება როგორც T0,T1,T2,T3. თითოეულ მათგანს შეუძლია სხვადასხვა ფუნქციის შესრულება.

TO ტაიმერ/მთვლელს გააჩნია ფუნქციების მინიმალური ანაკრები. იგი შეიძლება გამოყენებული იყოს დროითი ინტერვალების აღრიცხვისა და გაზომვისათვის ან როგორც გარე მოვლენების მთვლელი, განივიმპულსური მოდულიაციის (გიმ) სიგნალების გენერაციისათვის. ასევე მას აქვს შესაძლებლობა იმუშაოს ასინქრონულ რეჟიმში, როგორც რეალური დროის საათი..

ტაიმერ/მთვლელი T1 ასევე შესაძლებელია გამოყენებული იყოს დროითი ინტერვალის ათვლისათვის და როგორც გარე მოვლენების მთვლელი. ამის გარდა, მას შეუძლია დაიმახსოვროს თავისი მდგომარეობა გარე სიგნალის საშუალებით. როგორც ტაიმერ/მთვლელ T0-ს მას შეუძლია მუშაობა როგორც განივ-იმპულსური მოდულატორი.

ტაიმერ/მთვლელ T2 მთლიანად ანალოგიურია ტაიმერ/მთვლელ TO-ს. ერთ-ერთ მათგანს შეუძლია მუშაობა ასინქრონულ რეჟიმში, ხოლო მეორეს გარე მოვლენების მთვლელის რანგში.

ტაიმერ/მთვლელ T3 ფუნქციური შესაძლებლობით იდენტურია ტაიმერ/მთვლელ T1-ს.

მიკროკონტროლერის შემადგენლობაში შედის ასევე მოდარაჯე ტაიმერი, რომელიც არის აუცილებელი ატრიბუტი ყველა თანამედროვე მიკროკონტროლერებისათვის. ეს ტაიმერი გვაძლევს საშუალებას თავი ავარიდოთ პროგრამის არასანქციურ დაციკვლას, რომელიც წარმოიქმნება ამა თუ იმ მიზეზის გამო.

5.1.ტაიმერ /მთვლელის შემადგენლობა

თითოეული ტაიმერ/მთვლელის ბლოკი შეიცავს სამ შეტანა/გამოტანის რეგისტრს, რომელთა საშუალებით ხორციელდება მიკროკონტროლერის ცენტრალური ბირთვიდან მიმართვა მონაცემთა ჩაწერის ან ამოკითხვის მიზნით. ეს რეგისტრებია: **$TCNT_X$** რეგისტრი, რომელიც არის რევერსიული მთვლელი. მოდულის მუშაობის რეჟიმიდან გამომდინარე შეიძლება შესრულდეს მთვლელი რეგისტრის შემცველობის განულება, ინკრიმენტი (გაზრდა) ან დეკრემენტი (შემცირება) ტაიმერ/მთვლელის ყოველ სატაქტო სიგნალით clk_{to}(clk_{t2}); **$TCCR_X$** - მართვის რეგისტრი, რომელშიც იწერება მთვლელის მუშაობის პარამეტრების განმსაზღვრელი კოდი; **OCR_X** რეგისტრი შედის შედარების ბლოკის შემადგენლობაში. ტაიმერ/მთვლელის მუშა რეჟიმის დროს სრულდება უწყვეტად (ყოველ მანქანურ ციკლში) ამ რეგისტრის შედარება $TCNT_X$ რეგისტრთან. x ნიშნაკი მიუთითებს მთვლელზე, რომლის შემადგენლობაშიც შედის რეგისტრი. მაგალითად, T0 მთვლელისათვის გვექნება TCNT0, TCCR0, OCR0 და ა.შ.

T1,T3 მთვლელების შემადგენლობაში ასევე შედიან ე.წ. დაპყრობის **ICR1(ICR3**) რეგისტრები, რომელთა დანიშნულებაა შეინახონ დროის განსაზღრულ მომენტში ტაიმერ/მთვლელის შემცველობა.

მთვლელებში შესაძლებელია ადგილი ჰქონდეს სხვადასხვა ხდომილებას, რომლის დადგომის შედეგად გენერირდება შესაბამისი წყვეტის სიგნალი: ტაიმერ/მთვლელის გადავსება, როდესაც TCNT მთვლელში მაქსიმალური კოდის ჩაწერის შემდეგ მიწოდებულ სატაქტო სიგნალს იგი გადაჰყავს ნულის მდგომარეობაში. მის შედეგად გენერირებული წყვეტის სიგნალი აღინიშნება როგორც **TOV**; მთვლელის შემცველობის OCR რეგისტრთან თანხვედრის შემთხვევას აღნიშნავენ როგორც OC; მთვლელის შემცველობის შენახვის (დაპყრობის) ფაქტს - როგორც IC.

5.2.წყვეტა ტაიმერ/მთვლელებიდან

როგორც ზევით იყო აღნიშნული, ტაიმერ/მთვლელებში მთელი რიგი ხდომილებების დადგომის შემთხვევაში გენერირდება წყვეტის მოთხოვნის სიგნალები, რომელთა საფუძველზე მიკროკონტროლერი უნდა გადაირთოს შესაბამისი სამომსახურო პროგრამის შესრულებაზე. ამავე დროს ხდება მათი დაფიქსირება ალმების სახით. T0,T1,T2 ტაიმერებიდან წყვეტის სიგნალის მოსვლის ინდიკაციისათვის განკუთვნილია რეგისტრი TIFR (Timer/Counter Interrupt Flag Register-ტაიმერ/მთვლელიდან წყვეტის ალმების რეგისტრი).

ამ რეგისტრის ფორმატი ნაჩვენებია 5.1.სურ.-ზე, ხოლო მისი თანრიგების აღწერა 5.1.ცხრილში.

	7	6	5	4	3	2	1	0
	OCF2	TOV2	OCF1A	ICF1A	ICF1B	TOV1	OCF0	TOV0
წაკითხვა (R)/ჩაწერა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობები	0	0	0	0	0	0	0	0

სურ. 5.1. TIFR რეგისტრის ფორმატი

T1 და T3 ტაიმერ/მთვლელიდან წყვეტის ინდიკაციისთვის განკუთვნილია ETIFR რეგისტრი, რომელიც მოთავსებულია შეყვანა/გამოყვანის დამატებით რეგისტრებში. ამ რეგისტრის ფორმატი მოცემულია 5.2.სურ.-ზე, ხოლო მისი თანრიგების აღწერა - 5.2.ცხრილში. რაიმე ხდომილების დადგომის შემთხვევაში TIFR (ETIFR) რეგისტრების შესაბამისი თანრიგი დაყენდება “ერთიანის” მდგომარეობაში. წყვეტის დამუშავების ქვეპროგრამის დაწყების მომენტიდან ისინი აპარატურულად განულდება. ნებისმიერი ალამი აგრეთვე შეიძლება გავანულოთ პროგრამული გზით, თუ მასში ჩავწერთ “1”.

ცხრილი 5.1. TIFR რეგისტრის თანრიგები

დანიშნულება	აღწერა
OCF2	T2 ტაიმერ/მთვლელიდან “თანხვედრის” წყვეტის ალამი
TOV2	T2 ტაიმერ/მთვლელიდან გადავსების წყეტის ალამი
ICF1	T1 ტაიმერ/მთვლელიდან “დაპყრობის” წყვეტის ალამი
OCF1A	T1 ტაიმერ/მთვლელიდან “ A თანხვედრის” წყვეტის ალამი
OCF1B	T1 ტაიმერ/მთვლელიდან “ B თანხვედრის” წყვეტის ალამი
TOV1	T1 ტაიმერ/მთვლელიდან გადავსების წყვეტის ალამი
OCF0	T0 ტაიმერ/მთვლელიდან “თანხვედრის” წყვეტის ალამი
TOV0	T0 ტაიმერ/მთვლელიდან გადავსების წყვეტის ალამი

	7	6	5	4	3	2	1	0
	-	-	ICF3	OCF3A	OCF3B	TOV3	OCF3C	OCF1C
ჩაწერა (R)/წაკითხვა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	0	0	0	0	0	0	0	0

სურ.5.2. ETIFR რეგისტრის ფორმატი

ცხრილი 5.2. ETIFR რეგისტრის თანრიგები

თანრიგები	დასახელება	აღწერა
7,6		გამოუყენებელი თანრიგები. ამოიკითხება როგორც “0”
5	ICF3	წყვეტის ალამი, როდესაც ადგილი აქვს <დაპყრობას> T3 ტაიმერ/მთვლელ-ში.
4	OCF13A	წყვეტის ალამი, როდესაც დგილი აქვს < A თანხვდენას> T3 ტაიმერ/მთვლელში
3	OCF3B	წყვეტის ალამი, როდესაც ადგილი აქვს < B თანხვდენას> T3 ტაიმერ/მთვლელ -ში
2	TOV3	წყვეტის ალამი, როდესაც ადგილი აქვს გადგავსებას T3 ტაიმერ/მთვლელში
1	OCF3C	წყვეტის ალამი, როდესაც ადგილი აქვს < C თანხვდენას > T3 ტაიმერ/მთვლელში
0	OCFIC	წყვეტის ალამი, როდესაც ადგილი აქვს < C თანხვდენას > T1 ტაიმერ/მთვლელში

TO,T1,T2 ტაიმერ/მთვლელებიდან წყვეტის ნებადართვა/აკრძალვისათვის გამოიყენება რეგისტრი TIMSK (Timer/Counter Interrupt MaSK Registr-ტაიმერ/მთვლელებიდან წყვეტების ნიღბის რეგისტრი).

ამ რეგისტრის ფორმატი ნაჩვენებია 5.3.სურ.-ზე, ხოლო მისი თანრიგების აღწერა - 5.3.ცხრილში.

	7	6	5	4	3	2	1	0
	OCIE2	TOIE2	TOCIE1	OCIE1A	OCIEB	TOIE1	OCIEO	TOIEO
წაკითხვა (R)/ჩაწერა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	0	0	0	0	0	0	0	0

სურ.5.3. TIMSK რეგისტრის ფორმატი

საწყის მდგომარეობაში TIMSK რეგისტრის ყველა თანრიგში ნულია ჩაწერილი.

ცხრილი 5.3. TIMSK რეგისტრის თანრიგების ფუნქციური დანიშნულება

დასახელება	აღწერა
OCIE2	T2 ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამი <თანხვდენის> შემთხვევაში.
TOIE2	T2 ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამი გადავსების შემთხვევაში.
TCIE1	T1 ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამი <დაჭრობის> შემთხვევაში.
OCIE1A	T2ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამი < A თანხვდენის > შემთხვევაში.
OCIEB	T2 ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამია <B თანხვდენის > შემთხვევაში.
TOIE1	T1 ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამია გადავსების შემთხვევაში.
OCIEO	TO ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამია <თანხვდენის> შემთხვევაში.
TOIEO	TO ტაიმერ/მთვლელის წყვეტის ნებადართვის ალამია გადავსების შემთხვევაში.

მიკროკონტროლერ Atmega 128 T1 ,T3 ტაიმერ/მთვლელიდან წყვეტების ნებისდართვა/აკრძალვისათვის განკუთვნილია კიდევ ერთი ETIMSK (Enaible Timer/Counter interrupt MaSk Register- ტაიმერ/მთვლელებიდან წყვეტის ნიღბის რეგისტრი) რეგისტრი, რომელიც განთავსებულია შეტანა/გამოტანის რეგისტრების დამატებით სივრცეში. ამ რეგისტრის ფორმატი ნაჩვენებია 5.4.სურ..-ზე, მისი თანრიგების აღწერა მოცემულია 5.4.ცხრილში.

	7	6	5	4	3	2	1	0
	-	-	TICIE3	OCIE3A	OCIE3B	TOIE3	OCIE3C	OCIE1C
წაკითხვა (R)/ ჩაწერა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	0	0	0	0	0			

სურ.5.4. ETIMSK რეგისტრის ფორმატი

ცხრილი 5.4. ETIMSK რეგისტრის თანრიგების ფუნქციური დანიშნულება

თანრიგები	დასახელება	აღწერა
7,6	-	გამოუყენებელი თანრიგებია. ამოიკითხება როგორც "ნული"
5	TICIE3	წყვეტის ნებადართვის ალამი, როდესაც ადგილი აქვს <დაპყრობას> T3 ტაიმერ/მთვლელში
4	OCIE3A	წყვეტის ნებადართვის ალამი, როდესაც ადგილი აქვს < A თანხვდენას > T3 ტაიმერ/მთვლელში
3	OCIE3B	წყვეტის ნებადართვის ალამი, როდესაც ადგილი აქვს < B თანხვდენას > T3 ტაიმერ/მთვლელში
2	TOIE3	წყვეტის ნებადართვის ალამი, როდესაც ადგილი აქვს გადავსებას T3 ტაიმერ/მთვლელში
1	OCIE3C	წყვეტის ნებადართვის ალამი, როდესაც ადგილი აქვს < C თანხვდენას > T3 ტაიმერ/მთვლელში
0	OCIE1C	წყვეტის ნებადართვის ალამი, როდესაც ადგილი აქვს < C თანხვდენას > T1 ტაიმერ/მთვლელში.

5.3.ტაიმერ/მთვლელების გამომყვანების დანიშნულება

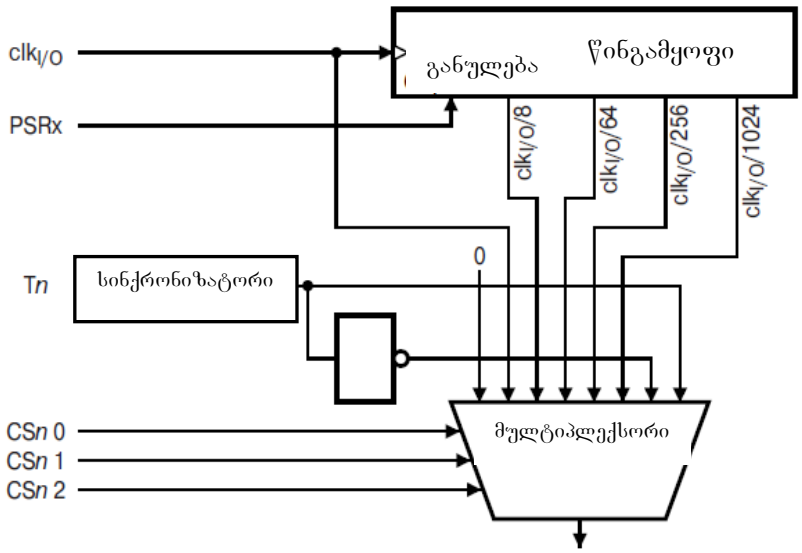
თითოეული ტაიმერ/მთვლელი იყენებს მიკროკონტროლერის ერთ ან მეტ გამომყვანს, როგორც წესი ეს გამომყვანები წარმოადგენს საერთო დანიშნულების შეტანა/გამოტანის პორტების სადენებს, ხოლო ფუნქციები რომლებსაც ასრულებს აღნიშნული გამომყვანები ტაიმერ/მთვლელებთან ერთობლივი მუშაობის დროს, წარმოადგენს მათ ალტერნატულ ფუნქციებს. მიკროკონტროლერის ყველა გამომყვანები, რომელებიც გამოიყენებიან ტაიმერ/მთვლელების მიერ მოცემულია 5.5. ცხრილში. აქვეა მითითებული ამ გამომყვანების ფუნქციები.

ცხრილი 5.5. გამომყვანები, რომლებიც გამოიყენება საერთო დანიშნულების ტაიმერ / მთვლელების მიერ

დასახელება	გამომყვანები	აღწერა
T0	-	TO ტაიმერის გარე სიგნალის შესასვლელი
OOC0	PB4	TO ტაიმერის შედარების სქემის გამოსასვლელი
T1	PD6	T1 ტაიმერის გარე სიგნალის შესასვლელი
ICPI	-	T1ტაიმერ დაპყრობის შესასვლელი
OOCIA	PB5	T1ტაიმერის შედარების სქემის გამოსასვლელი
OOCIB	PB6	
OCIC	PB7	
T2	PD7	T2ტაიმერის გარე სიგნალის შესასვლელი
OC2	PB7	T2ტაიმერის შედარების სქემის გამოსასვლელი
T3	PE6	T3 ტაიმერ გარე სიგნალის შესასვლელი
ICP3	PE7	T3ტაიმერ დაპყრობის შესასვლელი
OC3A	PE3	T3ტაიმერ შედარების სქემის გამოსასვლელი
OC3B	PE4	
OC3C	PE5	
TOSC1	PG4	შესასვლელი რეზონატორის მიერთებისთვის
TOSC2	PE3	გამოსასვლელი რეზონატორის მიერთებისთვის

5.4. ტაიმერ/მთვლელების სიხშირის წინაგამყოფები

სხვადასხვა ამოცანის მოთხოვნიდან გამომდინარე საჭიროა მთვლელის თვლის სიჩქარის ცვლილება, რაც შეიძლება განხორციელდეს სატაქტო იმპულსების სიხშირის ცვლით. მაღალი სიხშირის სატაქტო იმპულსების შემთხვევაში მთვლელი ითვლის უფრო სწრაფად და პირიქით. ამ მიზნით გამოიყენება ე.წ. წინაგამყოფები, რომლებიც განკუთვნილია ტაიმეტ/მთვლელების clk_{T0}, clk_{T1}, clk_{T2}, clk_{T3} ტაქტური სიგნალების ფორმირებისთვის. ტაიმერ/მთვლელების წინაგამყოფის გამარტივებული სტრუქტურული სქემა ნაჩვენებია 5.5.სურ-ზე.



T_n ტაიმერ/მთვლელის სატაქტო სიგნალი clk_{Tn}

n=1,2,3 (PSRx=PSR321);

სურ. 5.5. წინგამყოფის სტრუქტურული სქემა

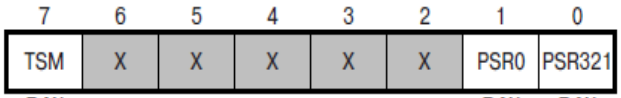
როგორც სტრუქტურული სქემიდან ჩანს ბლოკის შემადგენლობაშია 10 თანრიგიანი წინაგამყოფი და გამომყვანი მულტიპლექსორი (ტაქტური სიგნალების ამომრჩევი). წინგამყოფს შესასვლელზე მიეწოდება ძირითადი სატაქტო სიგნალები $clk_{I/O}$, რომლებიც შეიძლება დაფორმირდეს ან სისტემური ტაქტური $clk_{I/O}$ სიგნალის საფუძველზე, ან ასინქრონულ რეჟიმში სიგნალით, რომელიც მიიღება დამატებითი კვარცული რეზონატორიდან. გადართვა სინქრონულ და ასინქრონულ მუშაობის რეჟიმებს შორის ხორციელდება ASSR რეგისტრის AS თანრიგის მეშვეობით (იხ.ქვევით)

ამ სიგნალების საფუძველზე სიხშირის წინგამყოფი გამოსასვლელებზე აფორმირებს სხვადასხვა სიხშირის სატაქტო სიგნალებს $clk_{I/O}/8$, $clk_{I/O}/64$, $clk_{I/O}/256$, $clk_{I/O}/1024$. მულტიპლექსორის საშუალებით მთვლელს მიეწოდება ტაქტური იმპულსები ერთ-ერთი აღნიშნული გამოსასვლელიდან CS_n0, CS_n1, CS_n0 შესასვლელებზე მიწოდებული კოდის მიხედვით, რომელიც იწერება სათანადო მთვლელის $TCCR_n$ რეგისტრში.

საჭიროა გავითვალისწინოთ, რომ T1,T2,T3 ტაიმერ/მთვლელები იყენებენ ერთი და იგივე წინაგამყოფს, ხოლო T0 ტაიმერი – ცალკე წინაგამყოფს. ტაქტური სიგნალების მართვა ცალკეული ტაიმერ/მთვლელისათვის ხორციელდება ინდივიდუალურად.

წინაგამყოფის მართვა

გარდა ტაიმერ/მთვლელის სატაქტო სიგნალების მართვისა მიკროკონტროლერი იძლევა საშუალებას, განახორციელოს წინაგამყოფის განულება და ასევე მისი გაჩერება. ამისთვის ის იყენებს SFIOR სპეციალური ფუნქციების რეგისტრს. ამ რეგისტრის ფორმატი ნაჩვენებია 5.6.სურ.-ზე (თანრიგები, რომლებიც არ გამოიყენებიან ტაიმერ/მთვლელის წინაგამყოფის მართვის დროს ნახაზზე აღნიშნულია X- ით).



წაკითხვა(R)/ჩაწერა (W)
საწყისი მნიშვნელობები

PSR0—T0 მთვლელის განულება.
PSR321- T1,T2,T3 მთვლელების განულება

სურ.5.6. SFIOR რეგისტრის ფორმატი

ტაიმერ/მთვლელის წინაგამყოფის განულებისთვის გამოიყენება SFIOR რეგისტრის PSRx თანრიგები. ამ რეგისტრის თანრიგებში ერთიანის ჩაწერის შემთხვევაში შესაბამისი ტაიმერ/მთვლელების წინაგამყოფები გადადის საწყის მდგომარეობაში. კიდევ ერთხელ უნდა აღვინიშნოთ, რომ წინაგამყოფი შეიძლება გამოყენებულ იქნეს რამდენიმე ტაიმერ/მთვლელის მიერ და შესაბამისად განულება განხორციელდება ყველა იმ ტაიმერ/მთვლელებში, რომლებიც იყენებენ მას.

მიკროკონტროლერის ყველა წინაგამყოფის გაჩერება ხორციელდება SFIOR რეგისტრის TSM თანრიგში ერთიანის ჩაწერით. წინგამყოფის შემდგომი გაშვება ხორციელდება TSM თანრიგში ნულიანის ჩაწერით. მითითებული ფუნქცია შეიძლება გამოყენებული იყოს ტაიმერ/ მთვლელის სინქრონიზაციისათვის. ტაიმერ/მთვლელების გაჩერების შემდეგ შესაძლებელია მათი ინიციალიზაცია (საჭირო პარამეტრების დაყენება). ხელახალი გაშვების შემთხვევაში ყველა ტაიმერ/მთვლელი დაიწყებს მუშაობას ერთდროულად.

5.5.მუშაობის რეჟიმები

ტაიმერ/მთვლელები შესაძლებელია მუშაობდეს 4 სხვდასხვა რეჟიმში. მთვლელი ტაიმერის რეჟიმის არჩევა ხდება $TCCR_n$ მართვის რეგისტრის WGM_n0:WGM_n1 თანრიგებში კოდის ჩაწერით. 5.6.ცხრილში მოცემულია კოდების შესაბამისობა ტაიმერის რეჟიმებთან.

ცხრილი 5.6 ტაიმერ/მთვლელების მუშაობის რეჟიმები

რეჟიმის ნომერი	WGM_{n1} (CTC_n)	WGM_{n1} (PWM_n)	T_n ტაიმერ/მთვლელ მუშაობის რეჟიმი
0	0	0	Normal
1	0	1	Phase correqt PWM
2	1	0	CTC
3	1	1	Fast PWM

შენიშვნა: n=0 ან 2

Normal რეჟიმი

ტაიმერ/მთვლელის მუშაობის ეს რეჟიმი ყველაზე უფრო მარტივი რეჟიმია. ამ რეჟიმში მთვლელი რეგისტრი ფუნქციონირებს როგორც ამჯამავი მთვლელი. ყოველი სატაქტო იმპულსი ახდენს მთვლელი რეგისტრის შემცველობის გაზრდას ერთით. $FF მაქსიმალური მნიშვნელობის მიღწევის შემდეგ (იგი ჩაწერილია თექვსმეტობით სისტემაში, რაც ნიშნავს ორობით კოდში ყველა თანრიგში ერთიანს) იგი ისევ იწყებს თვლას $00-დან. ამავე დროს ფორმირდება გადავსების სიგნალი და ერთიანის მდგომარეობაში დგება T0VFn გადავსების ალამი.

ტაიმერ/მთვლელის თვლის პროცესში ხდება მისი შედარება OCRn შედარების რეგისტრში ჩაწერილ კოდთან. მნიშვნელობების თანხვედრის შემთხვევაში TIFR რეგისტრის OCFn წყვეტის ალამი დგება ერთის მდგომარეობაში და თუ TIMASK რეგისტის OCIEn თანრიგში ჩაწერილია ერთი (ამ წყვეტის ნების დართვა) გენერირდება წყვეტის სიგნალი.

მთვლელი რეგისტრის და შედარების რეგისტრის ტოლობის შემთხვევაში OCFn ალმის დაყენებასთან ერთად შესაძლებელია შეიცვალოს მიკროკონტროლერის OC გამომყვანის მდგომარეობა. ხოლო, მისი ცვლილება თუ როგორი სახით მოხდება, განისაზღვრება TCCR რეგისტრის COMn1:COMn0 თანრიგების მიხედვით, 5.7.ცხრილის შეესაბამისად.

ცხრილი 5.7. Normal რეჟიმში OC გამომყვანის მართვა

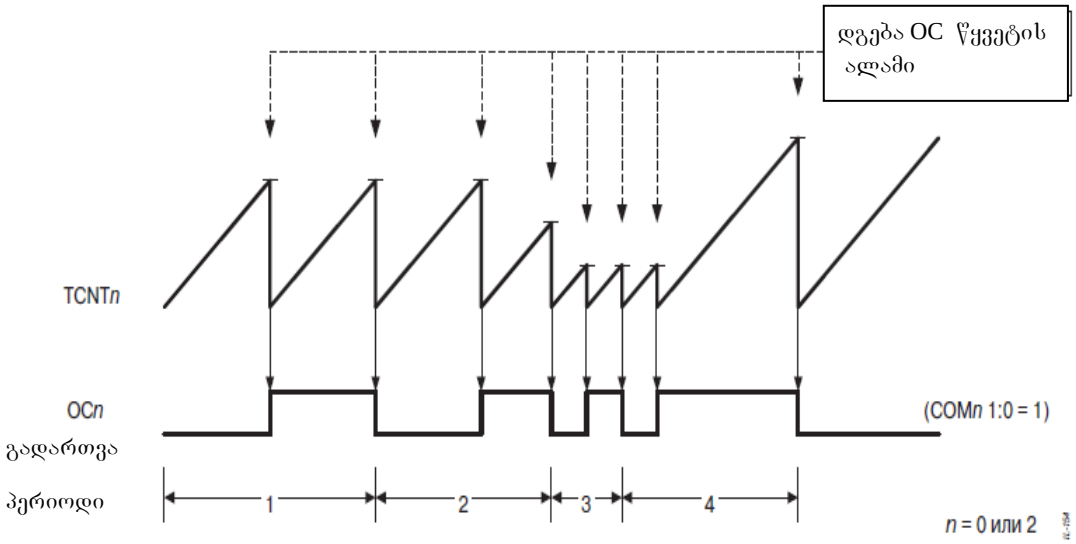
COMn1	COMn0	აღწერა
0	0	ტაიმერ/მთვლელი გამორთულია OCn გამომყვანიდან
0	1	გამომყვანის მდგომარეობა იცვლება საწინააღმდეგოდ
1	0	გამომყვანზე დგება ნული
1	1	გამომყვანზე დაგება ერთი

შენიშვნა: n = 0 ან 2

საჭიროების შემთხვევაში OC გამომყვანის მდგომარეობა შეიძლება იძულებით შეიცვალოს TCCR მართვის რეგისტრის FOC თანრიგში ლოგიკური ერთიანის ჩაწერით. ამ შემთხვევაში წყვეტა არ გენერირდება.

CTC რეჟიმი (თანხვდომის დროს განულება)

ამ რეჟიმში მთვლელი რეგისტრი ისევე ფუნქციონირებს, როგორც ჩვეულებრივი ამჯამავი მთვლელი, რომლის ინკრიმენტი ხორციელდება ყოველი ClkTn ტაქტური იმპულსით. მაგრამ, მთვლელი რეგისტრის მაქსიმალური შესაძლებელი მნიშვნელობა განისაზღვრება შედარების OCR რეგისტრის შემცველობით. შედარების რეგისტრში ჩაწერილი მნიშვნელობის მიღწევის შემდეგ თვლა გრძელდება <$00> მნიშვნელობიდან. იმავე clkT სატაქტო იმპულსზე, რომელზეც ხორციელდება მთვლელი რეგისტრის განულება, TIFR რეგისტრის TOVFn წყვეტის ალამი დგება ერთიანში. ამ რეჟიმის დროითი დიაგრამა ნაჩვენებია 5.7.სურ.-ზე.



სურ. 5.7. CTC რეჟიმის დროითი დიაგრამა

როგორც დიაგრამიდან ჩანს შედარების კონსტანტის ცვლა იწვევს OC გამოსასვლელზე სიგნალის ხანგრძლივობის ანუ პერიოდის (დროითი შუალედი მეზობელ სიგნალებს შორის) ცვლას, რაც ნიშნავს გამომავლი სიგნალების სიხშირის ცვლას. მოცემული სიხშირით სიგნალების გენერაციისათვის აუცილებელია COMn1:COMn0 თანრიგებში ჩაიწეროს “01” მნიშვნელობა (OC გამომყვანის მდგომარეობის გადართვა) (იხ. ცხრილი 5.7.). სიგნალის გენერაციის სიხშირე განსაზღრული იქნება გამოსახულებით $f_{OCn} = =f_{clk\_I/O}$ /2N(1+ +OCR_n), სადაც N არის წინაგამყოფის გაყოფის კოეფიციენტს.

განხილულ რეჟიმს, ისე, როგორც მომდევნო ორ რეჟიმს ეწოდება ***განივიმპულსური*** *მოდულაციის (გიმ) რეჟიმი.*

Fast PWM რეჟიმი

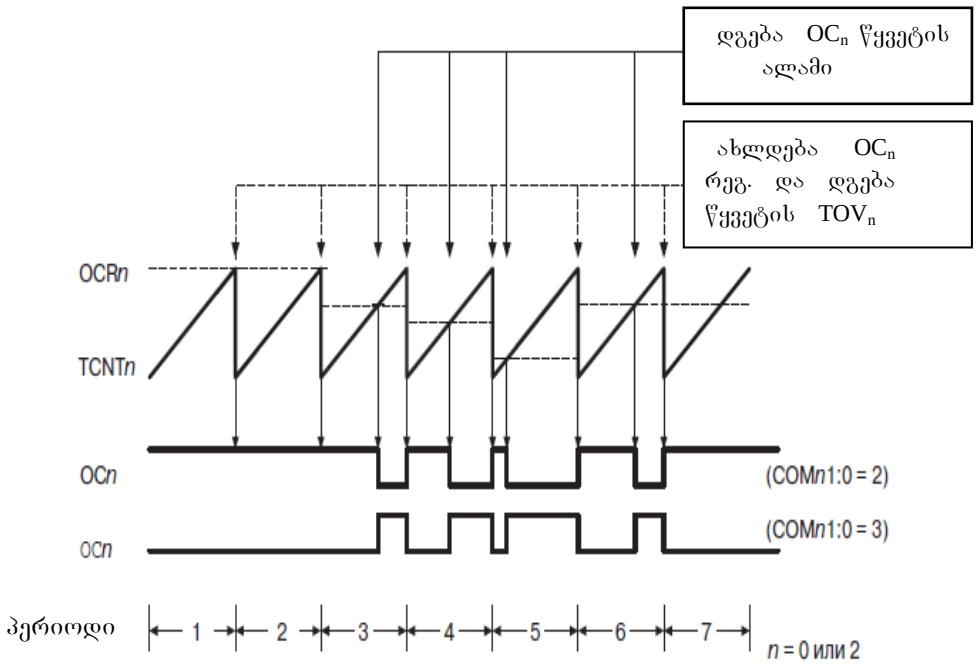
Fast PWM რეჟიმი (სწრაფი განივიმპულსური მოდულიაცია) გვაძლევს საშუალებას გენერირება განვახორციელოთ მაღალსიხშირული განივიმპულსური მოდულიაციით. იმის გამო, რომ ხდება სიგნალის მაღალი სიხშირით გენერაცია, მოცემული რეჟიმი

წარმატებით შეიძლება გამოყენებული იყოს სიმძლავრეების რეგულირების დროს, ციფრულ-ანალოგურ გარდამქმნელებში და ა.შ.

მთვლელი რეგისტრი ამ რეჟიმში ფუნქციონირებს როგორც ამჯამავი მთვლელი, რომლის ინკრიმენტი ხორციელდება ყოველი სატაქტო სიგნალის იმპულსით. მთვლელის მდგომარეობა იცვლება $00 –დან $FF- მდე, რის შემდეგ მთვლელი რეგისტრი ნულდება და ციკლი მეორდება, იმ შემთხვევაში როდესაც მთვლელი რეგისტრი მიაღწევს თავის მაქსიმალურ მნიშვნელობას ხორციელდება TIFR რეგისტრში TOV_n წყვეტის ალმის დაყენება. როდესაც მთვლელი რეგისტრის შემცველობა OCR_n შედარების რეგისტრს გაუტოლდება, მაშინ TIFR რეგისტრში დაყენდება OCF_n ალამი.

ამ რეჟიმში შედარების სქემის მუშაობის თავისებურება არის OCR_n რეგისტრში ჩაწერის დროს ორმაგი ბუფერიზაცია, რომელიც მდგომარეობს იმაში, რომ OCRn-ში ჩასაწერი რიცხვი თავდაპირველად შეინახება სპეციალურ ბუფერულ რეგისტრში, ხოლო შედარების რეგისტრის შემცველობის ცვლილება ხორციელდება მთვლელის მაქსიმალურ $FF მნიშვნელობის მიღწევის შემდეგ. ასეთი გადაწყვეტილების მიღებამ გამორიცხა მოდულატორის გამოსასვლელზე არასიმეტრიული იმპულსური სიგნალების (შეფერხებების) წარმოქმნა, რომელსაც ექნებოდა ადგილი შედარების რეგისტრში უშუალოდ ჩაწერის დროს.

მიკროკონტროლერის OC_n გამომყვანის ფუნქციონირება განისაზღვრება ასევე $TCCR_n$ რეგისტრის COMn1:COMn0 თანრიგების შემცველობით (როგორც 5.7.ცხრილიში იყო ნაჩვენები). ამ რეჟიმში ტაიმერ მთვლელის მუშაობის დროითი დიაგრამა ნაჩვენებია 5.8.სურ.-ზე.

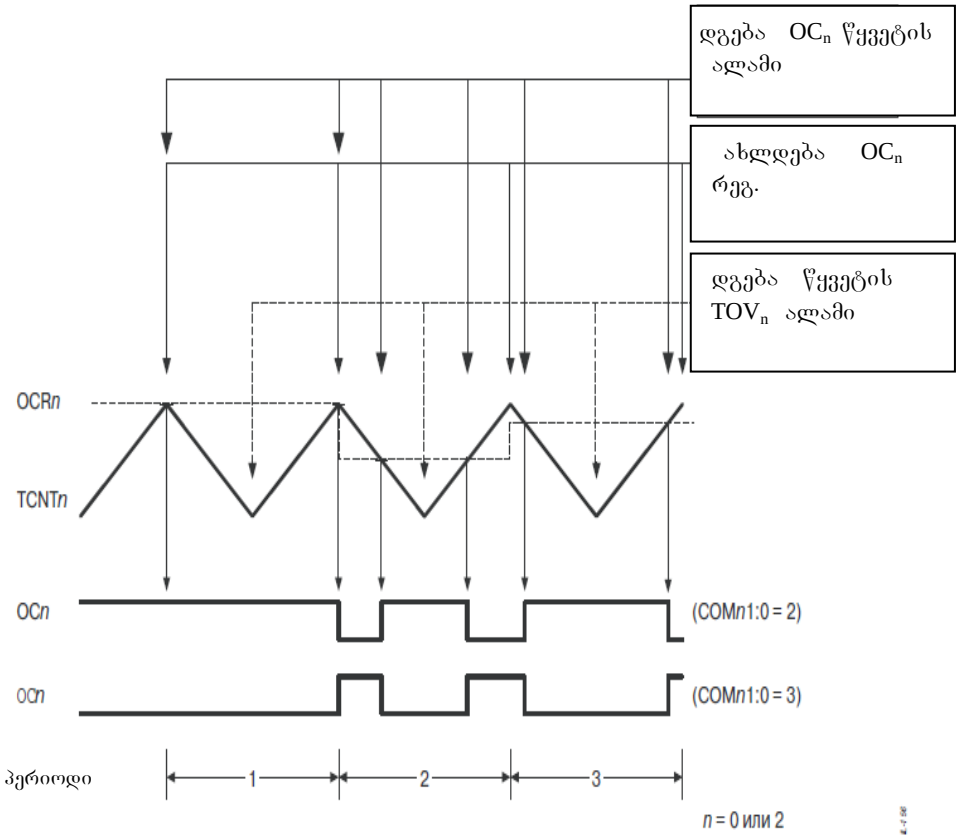


სურ.5.8. Fast PWM რეჟიმის დროითი დიაგრამა

Phase Correct PWM რეჟიმი

Phase Correct PWM (განივიმპულსური მოდულაცია ზუსტი ფაზით), როგორც Fast PWM რეჟიმი განკუთვნილია განივიმპულსური მოდულაციის სიგნალების გენერაციისათვის. ამ რეჟიმში მთვლელი რეგისტრი ფუნქციონირებს, როგორც რევერსიული მთვლელი, რომლის მდგომარეობის შეცვლა ხორციელდება ყოველი Clk ტაქტური სიგნალით. დასაწყისში მთვლელის მდგომარეობა იცვლება **$**00 –დან **$**FF- მდე, ხოლო შემდგომ პირიქით - **$**FF-დან **$**00-მდე: როდესაც მთვლელი მიაღწევს თავის მაქსიმალურ **$**FF მნიშვნელობას, მოხდება თვლის მიმართულების შეცვლა, მთვლელის მინიმალური **$00** მნიშვნელობის მიღწევის დროს ასევე მოხდება თვლის მიმართულების შეცვლა და ერთდროულად, TIFR რეგისტრის TOVn (წყვეტა ტაიმერის გადავსების დროს} ალამის დაყენება.

თვლის რეგისტრის და შედარების OCRn რეგისტრის შემცველობის ტოლობის შემთხვევაში მოხდება TIFRn რეგისტრის OCFn (წყვეტა თანხვედრის დროს) ალმის დაყენება და იცვლება OCn გამომყვანის მდგომარეობა. თუ როგორი სახით იცვლება მისი მდგომარეობა განისაზღვრება TCCRn რეგისტრის COMn1:COMn0 თანრიგების შემცველობით. (იხ. 5.7.ცხრილი). მთვლელის მუშაობის დროითი დიაგრამა ნაჩვენებია 5.9.სურ.-ზე).



სურ.5.9. Phase Correct PWM რეჟიმის დროითი დიაგრამა

ამ რეჟიმში ასევე რეალიზებულია OCRn რეგისტრში ჩაწერის ორმაგი ბუფერი ზაცია, როგორც წინა რეჟიმში იყო ახსნილი.

განხილულ რეჟიმში გენერირებული სიგნალის სიხშირე განისაზღვრება გამოსახულებით: f_{oc}=Fclk$_{I/O}$/512N, სადაც N- წინაგამყოფის გაყოფის კოეფიციენტია.

ასინქრონული რეჟიმი

T0 ტაიმერ/მთვლელს, სხვა მთვლელებისგან განსხვავებით, გარდა სინქრონული რეჟიმისა, შეუძლია მუშაობა ასინქრონულ რეჟიმში. ასინქრონული რეჟიმი ითვალისწინებს ტაიმერთან ცალკე სატაქტო სიგნალების წყაროდან ტაქტირებას. სიგნალის სიხშირის მიმწოდებლად შეიძლება გამოყენებული იყოს, როგორც კვარცული რეზონატორი, მიერთებული მიკროკონტროლერის TOSC1 და TOSC2 გამომყვანებთან, ასევე სიგნალი გარე სქემიდან, რომელიც მიეწოდება TOSC1 გამომყვანზე. ეს შესაძლებლობას იძლევა ვამუშაოთ ტაიმერი, ამოცანის მოთხოვნიდან გამომდინარე, სხვადასხვა სიხშირეზე. მაგალითად, როგორც რეალური დროის საათი. მიუხედავად იმისა, რომ T0 ტაიმერ /მთვლელის სატაქტო გენერატორი აწყობილია 32768 ჰერც სიხშირეზე, კვარცული რეზონატორის სიხშირე ან სიგნალი გარე სქემიდან შესაძლებელი იყოს 0-დან 256 კჰც საზღვრებში.

ტაიმერ/მთვლელის ასინქრონულ რეჟიმში გადართვისათვის განკუთვნილია ASSR რეგისტრი. აღნიშნული რეგისტრის ფორმატი ნაჩვენებია 5.10.სურ.-ზე. მისი ცალკეული თანრიგების აღწერა 5.8.ცხრილში.

	7	6	5	4	3	2	1	0
	-	-	-	-	ASO	TCN0UB	OCR0UB	TCR0UB
წაკითხვა (R) / ჩაწერა(W)	R	R	R	R	R/W	R	R	R
საწყისი მნიშვნელობები	0	0	0	0	0	0	0	0

სურ.5.10. ASSR რეგისტრის ფორმატი

5.6.T0 , T2 ტაიმერ/მთვლელები

ტაიმერ/მთვლელების T0, T2 შემადგენლობაში არის სამი შეტანა/გამოტანის რეგისტრი: TCNTO (TCNT2) მთვლელი რეგისტრი, TCCR0 (TCCR2) მართვის რეგისტრი და OCR0(OCR2) შედარების რეგისტრი. T0-ს დამატებული აქვს ASSR რეგისტრი, რომელიც ემსახურება ტაიმერ/მთვლელის მოდულის მართვას ასინქრონულ რეჟიმში. ორივე მთვლელის ყველა რეგისტრი 8 თანრიგაა. T0/T2 ტაიმერ/მთვლელებმა შეიძლება განახორციელონ წყვეტის გენერაცია მთვლელი რეგისტრის გადავსების ან მთვლელი რეგისტრისა და შედარების რეგისტრის ტოლობის დროს. ამ ორი წყვეტის ალმები განთავსებულია TIFR რეგისტრში. ხოლო ამ წყვეტების ნებადართვა/აკრძალვა სრულდება TIMSK რეგისტრის შესაბამისი ნებადართვა/აკრძალვის ალმების დაყენებით, როგრც ზევით იყო აღნიშნული.

ტაიმერ/მთვლელების TCNT0 (TCNT2) რეგისტრები შედის შესაბამისი ბლოკების შემადგენლობაში და წარმოადგენენ რევერსიულ მთვლელს. მოდულის მუშაობის რეჟიმიდან გამომდინარე სრულდება მთვლელი რეგისტრის შემცველობის განულება,

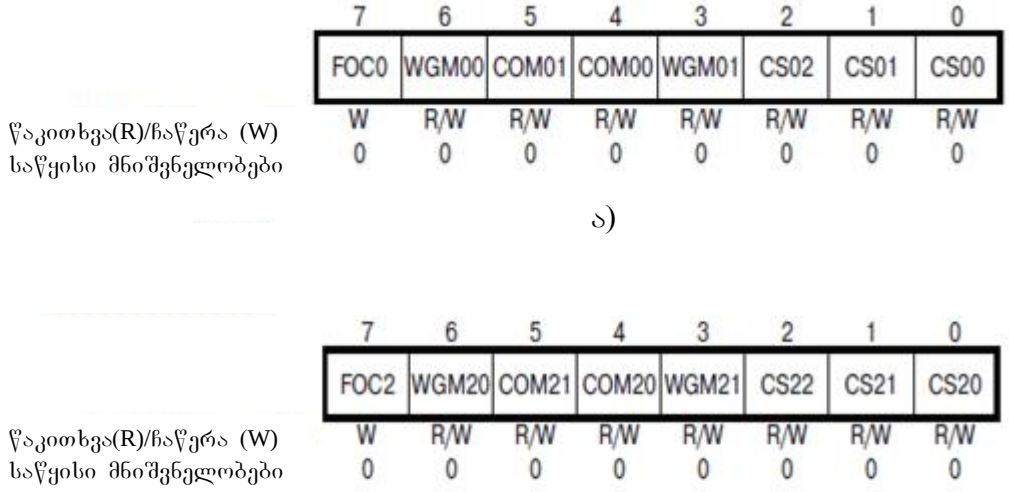
ინკრიმენტი ან დეკრემენტი ტაიმერ/მთვლელის ყოველ სატაქტო clk_{to}(clk_{t2}) სიგნალით. რეგისტრები მიკითხებადი არიან დროის ნებისმიერ მომენტში, როგორც ჩაწერის, ისე წაკითხვის ოპერაცია მთვლელ რეგისტრში ახდენს შედარების ბლოკის მუშაობის ბლოკირებას ტაიმერ/მთვლელის ტაქტური სიგნალის ერთი პერიოდის განმაულობაში..

ცხრილი 5.8. ASSR რეგისტრის ცალკეული თანრიგების მნიშვნელობები ასინქრონულ რეჟიმის დროს

თანრიგები	დასახელება	აღწერა
7.....4	–	დარეზერვირებულია, წაიკითხება ნული
3	AS_n	**სამუშაო რეჟიმის გადართვა**. თუ თანრიგი ერთიანშია, T_n ტაიმერ/მთვლელის წინგამყოფის შესასვლელზე მიეწოდება იმპულსები ტაიმერ/მთვლელის კვარცული გენერატორიდან (ასინქრონულუ რეჟიმში). ამ რეჟიმში TOSC1 და TOSC2 გამომყვანები გამოიყენება კვარცული რეზონატორის მიერთებისთვის და შესაბამისად არ შეიძლება გამოყენებული იყოს როგორც საერთო დანიშნულების შეტანა/ გამოტანის კონტაქტები. თუ თანრიგი განულებულია, მაშინ წინაგამყოფის შესასვლელს მიეწოდება მიკროკონტროლერის შიგა ტაქტური სიგნალი. ამ შემთხვევაში TOSC1 და TOSC2 გამომყვანები საერთო დანიშნულების შეტანა/გამოტანის შესასვლელებია. ამ თანრიგის მდგომარეობის შეცვლის დროს TCNT,OCR და TCCR შემცველობა შეიძლება დაზიანდეს.
2	TCN_nUB	**$TCNT_n$ რეგისტრის მდგომარეობის აღდგენა.** $TCNT_n$ რეგისტრში მნიშვნელობის ჩაწერისათვის ეს თანრიგი უნდა იყოს დაყენებული ერთიანის მდგომარეობაში. ხოლო ამ რეგისტრში გადაგზავნილი მნიშვნელობის ჩაწერის შემდეგ ალამი აპარატურულად განულდება. **$TCNTU_n$** განულები სშემდეგ $TCNT_n$ რეგისტრი მზადაა ახალი მნიშვნელობის ჩაწერისათვის. $TCNT_n$ რეგისტრი ჩაწერისას თუ დაყენებულია $TCNTU_n$ ალამი შეიძლება მოხდეს რეგისტრში ადრე ჩაწერილი შემცველობის დაზიანება და წყვეტის გენერაცია.
1	OCR_nUB	**OCR_n რეგისტრის მდგომარეობის აღდგენა.** OCR_n რეგისტრში მნიშვნელობის ჩაწერისთვის ხდება OCR_nUB ალმის ერთიანში დაყენება.ხოლო აღნიშნულ რეგისტრში მნიშვნელობის ჩაწერის შემდეგ სრულდება ამ ალამის აპარატურული ჩამოგდება. რაც იმის მაუწყებელია, რომ რეგისტრი OCR_n მზად არის ახალი მნიშვნელობის ჩასაწერად. რეგისტრში ჩაწერისას თუ დაყენებულია OCR_nUB ალამი შეიძლება გამოიწვიოს რეგისტრში ადრე ჩაწერილი შემცველობის დაზიანება და წყვეტის გენერაცია.
0	TCR_nUB	**$TCCR_n$ რეგისტრის მდგომარეობის აღდგენა.** $TCCR_n$ რეგისტრში მნიშვნელობის ჩაწერისთვის ხდება TCR_nUB ალმის ერთიანში დაყენება. ხოლო აღნიშნულ რეგისტრში მნიშვნელობის ჩაწერის შემდეგ სრულდება ამ ალამის აპარატურული განულება, რაც იმის მაუწყებელია, რომ $TCCR_n$ რეგისტრი მზად არის ახალი მნიშვნელობის ჩასაწერად. რეგისტრში ჩაწერისას თუ დაყენებულია TCR_nUB ალამი შეიძლება გამოიწვიოს რეგისტრში ადრე ჩაწერილი შემცველობის დაზიანება და წყვეტის გენერაცია.

რეგისტრები OCR0(OCR2) შედის შედარების მოდულის შემადგენლობაში. ტაიმერ/მთვლელის მუშა რეჟიმის დროს სრულდება უწყვეტად (ყოველ მანქანურ ციკლში) რეგისტრის შედარება TCNT0(TCNT2) რეგისტრთან. იმ შემთხვევაში, თუ ამ ორი რეგისტრის შემცველობა ერთმანეთის ტოლია, მაშინ მომდევნო მანქანურ ციკლში მოხდება TIFR რეგისტრის OCF0(OCF2) ალმის დაყენება. რის საფუძველზეც სრულდება წყვეტის გენერაცია (თუ წყვეტა ნებადართულია). გარდა ამისა, ამ ხდომილების დროს შეიძლება შეიცვალოს მიკროკონტროლერის OC0(OC2) გამომყვანის მდგომარეობა (უნდა მოხდეს მისი დაკონფიგურირება როგორც გამოსასვლელის DDRx რეგისტრის შესაბამისი თანრიგის ერთანში დაყენებით).

TCCR0 (TCCR2) განკუთვნილია ტაიმერ/მთვლელის მოდულის მართვისათვის. ამ რეგისტრის ფორმატი ნაჩვენებია 5.11.სურ.-ზე



სურ.5.11. TCCR0 (ა) და TCCR2 (ბ) რეგისტრების ფორმატები

აღნიშნულ რეგისტრებში სრულდება იმ რეჟიმების დაყენება, რომლებიც ზევით იყო განხილული. კერძოთ, სატაქტო სიგნალების სიხშირის არჩევა (CSn0-CSn0-2 თანრიგებით), განივიმპულსური მოდულიაციის რეჟიმის არჩევა (WGMn0-WGMn1 თანრიგებით), OCn შედარების სქემის გამოსასვლელის მდგომარეობის ცვლილების არჩევა თანხვედრის შემთხვევაში (COMn0-COMn1 თანრიგებით). რეგისტრის თანრიგების დანიშნულების უფრო დეტალური აღწერა მოცემულია 5.9.ცხრილში.

5.7. T1 და T3 ტაიმერ/მთვლელები

Atmega 128-ს აგრეთვე გააჩნია ტაიმერ/მთვლელები T1 და T3. ტაიმერ/მთვლელები T1 და T3 ისევე, როგორც ტაიმერ/მთვლელები T0 და T2 შეიძლება გამოვიყენოთ დროითი ინტერვალის ფორმირებისთვის, გარე მოვლენების რაოდენობის აღრიცხვისათვის, სიგნალების ფორმირებისთვის და განივიმპულსური მოდულიაციის სიგნალების

გენერირებისათვის. დამატებით ტაიმერ/მთვლელებს T1 და T3 აქვს უნარი გარე სიგნალის ზემოქმედებით შეინახოს თავისი მიმდინარე მდგომარეობა ცალკე შეტანა/გამოტანის რეგისტრში.

ორივე ტაიმერ/მთვლელების შემადგენლობაში შედის შემდეგი შეტანა/გამოტანის რეგისტრები:

- 16 თანრიგიანი TCNT1(TCNT3) მთვლელი რეგისტრი;
- 16 თანრიგიანი ICR1(ICR3) დაპყრობის რეგისტრი;
- სამი 16 თანრიგიანი OCR1A, OCR1B, OCR1C (OCR3A , OCR3B, OCR3C) შედარების რეგისტრი;
- სამი 8 თანრიგიანი TCCR1A, TCCR1B, TCCR1C (TCCR3A, TCCR3B, TCCR3C) მართვის რეგისტრი;

ცხრილი 5.9. TCCR0 (TCCR2) რეგისტრის ფორმატი

<table>
<tr><th>თანრიგი</th><th>დასახელება</th><th>აღწერა</th></tr>
<tr><td>7</td><td>FOC_n</td><td>OC_n გამომყვანის მდგომარეობის იძულებითი ცვლილება (Normal და CTC რეჟიმები) .იმ შემთხვევაში როდესაც ერთიანია ჩაწერილი ამ თანრიგში OC_n გამოსასვლელის მნიშვნელობა იცვლება იმის შესაბამისად თუ რა მდგომარეობაშია COM_n1: COM_n2 თანრიგები. წყვეტის გენერირება ამ შემთხვევაში არ ხორციელდება და ტაიმერის განულება (CTC რეჟიმში) არ ხდება. Fast PWM და Phase Correct PWM რეჟიმებში ეს თანრიგი უნდა ჩამოიგდოს ნულში. თანრიგის წაკითხვის დროს ყოველთვის უბრუნდება ნულოვან მდგომარეობას.</td></tr>
<tr><td rowspan="5">6,3</td><td rowspan="5">WGM_n1: WGM_n0</td><td colspan="4">ტაიმერ/მთვლელის მუშობის რეჟიმი. ეს თანრიგები განსაზღვრავს ტაიმეტრ/მთვლელის მუშა რეჟიმს შემდეგნაერად:</td></tr>
<tr><td>რეჟიმის ნომერი</td><td>WGM_n1</td><td>WGM_n0</td><td>T_n ტაიმერ/მთვლელის მუშაობის რეჟიმი</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>Normal</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>Phase correct PWM</td></tr>
<tr><td>2</td><td>1</td><td>0</td><td>CTC (განულება თანხვდენის დროს)</td></tr>
<tr><td>3</td><td>1</td><td>1</td><td>Fast PWM</td></tr>
<tr><td>5,4</td><td>COM_n1: COM_{n2}</td><td colspan="4">თანხვდენის ბლოკის მუშაობის რეჟიმი. ეს თანრიგები განსაზღვრავს OC_n გამომყვანის მოქმედებას, როდესაც ადგილი აქვს თანხვდენას. გამომყვანის მნიშვნელობაზე ამ თანრიგების ზემოქმედება განისაზღვრება ტაიმერ/მთვლელის მუშა რეჟიმით.</td></tr>
<tr><td>2....0</td><td>CS_n2.... CS_n0</td><td colspan="4">ტაქტური სიგნალების მართვა. ეს თანრიგები განსაზღვრავს მიკროპროცესორის ტაქტური სიგნალების წყაროს.</td></tr>
</table>

ყოველი 16 თანრიგიანი რეგისტრი ფიზიკურად განთავსებულია შეტანა/გამოტანის ორ რეგისტრში, რომლებიც აღინიშნება TCNT1-თვის როგორც TCNT1H (უფროსი ბაიტი): TCNT1L (უმცროსი ბაიტი). TCNT3-თვის - TCNT3H:TCNT3L.

T1 და T3 ტაიმერ/მთვლელებს შეუძლიათ განახორციელოს წყვეტის გენერაცია შემდეგი მოვლენების დადგომის დროს:

- მთვლელი რეგისტრის გადავსების შემთხვევაში;
- მთვლელი რეგისტრის და შედარების რეგისტრის ტოლობის შემთხვევაში (ყოველი შედარების ბლოკისათვის თითო წყვეტა);
- დაპყრობის რეგისტრში მთვლელი რეგისტრის შენახვის შემთხვევაში

T1 და T3 ტაიმერ/მთვლელების ყველა წყვეტების ალმები მოთავსებულია TIFR და ETIFR რეგისტრებში. ხოლო ამ წყვეტების ნებადართვა/აკრძალვა ხორციელდება TIMSK და ETIMSK რეგისტრების შესაბამისი ალმების დაყენება/ჩამოყრით (იხ. ზემოთ განხილული წყვეტა ტაიმერ/მთვლელებში).

OCR1A/ OCR1B/ OCR1C (OCR3A/ OCR3B/ OCR3C) შედიან შედარების ბლოკის შემადგენლობაში. ტაიმერ/მთვლელის მუშაობის დროს უწყვეტად (ყოველ მანქანურ ციკლში) ხორციელდება ამ რეგისტრების შედარება TCNT1(TCNT3) რეგისტრებთან. შედარების დროს, თუ ეს რეგისტრები ერთმანეთის ტოლი აღმოჩნდება, მაშინ მომდევნო მანქანურ ციკლში დაყენდება TIFR რეგისტრის შესაბამისი OCF1A/ OCF1B/ OCF1C (OCF3A/ OCF3B/ OCF3C) ალამი და ხორციელდება წყვეტის გენერაცია (თუ წყვეტა ნებადართულია). ამ მოვლენის დადგომის დროს ადგილი უნდა ჰქონდეს მიკროკონტროლერის OCF1A/ OCF1B/ OCF1C(OCF3A/ OCF3B/ OCF3C) გამომყვანების მდგომარეობის შესაძლო ცვლილებას

იმისათვის, რომ ტაიმერ/მთვლელმა შეძლოს რომელიმე ამ გამომყვანის მართვა უნდა მოხდეს მისი კონფიგურირება, როგორც გამოსასვლელის (DDRx რეგისტრის შესაბამი თანრიგი უნდა დაყენდეს " ერთიანის" მდგომარეობაში).

დაპყრობის ICR1(ICR3) რეგისტრი შედის დაპყრობის ბლოკის შემადგენლობაში, რომლის დანიშნულებაა შეინახოს განსაზღრული დროის მომენტში ტაიმერ/მთვლელის მდგომარეობა ICR1(ICR3) დაპყრობის რეგისტრში. ეს მოქმედება შეიძლება შესრულდეს მიკროკონტროლერის ICP1(ICP3) გამოსასვლელზე სიგნალის მიწოდებით ან (T1 ტაიმერ/მთვლელისათვის) ანალოგური კომპარატორიდან. ერთდროულად დაპყრობის რეგისტრში ჩაწერისას ხდება TIFR რეგისტრის ICF1 ალამის დაყენება (ETIFR რეგისტრში ICF3 ალამის დაყენება) და ფორმირდება მოთხოვნა წყვეტის გენერაციაზე. ნებადართვა წყვეტაზე ხორციელდება TIMSK რეგისტრის ICIE1 თანრიგის ან ETIMSK რეგისტრის ICIE3 თანრიგის "ერთიანში" დაყენებით.

ტაიმერ/მთვლელის მართვისათვის გამოიყენება მართვის სამი რეგისტრი: TCCR1A (TCCR3A), TCCR1B (TCCR3B), TCCR1C(TCCR3C). ამ რეგისტრების ფორმატი მოყვანილია 5.12...5.14.სურ.-ზე, ხოლო აღნიშნული თანრიგების აღწერა მოცემულია 5.10....5.12.ცხრილში.

	7	6	5	4	3	2	1	0
	COMnA1	COMnA0	COMnB1	COMnB0	COMnC1	COMnC0	WGMn1	WGMn0
წაკითხვა (R)/ ჩაწერა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობები	0	0	0	0	0	0	0	0

სურ.5.12. TCCR1A (TCCR3A) რეგისტრის ფორმატი

ცხრილი 5.10. TCCR1A(TCCR3A) რეგისტრის თანრიგები

თანრიგი	დასახელება	აღწერა
7,6	COMnA1: COMnA0	**შედარების ბლოკის მუშაობის რეჟიმი**. ეს თანრიგები განსაზღვრავს OCnx გამომყვანის ფუნქციონირებას "თანხვდენის" შემთხვევის დროს. ამ თანრიგე- ბის შემცველობის გავლენა გამომყვანის მდგომარე- ობაზე დამოკიდებულია ტაიმერ/მთვლელის მუშაო- ბის რეჟიმზე.
5,4	COMnB1: COMnB0	
3,2	COMnC1: COMnC0	
0,1	WGMn1A: WGMn0A	**ტაიმერ/მთვლელის მუშაობის რეჟიმი.** TCCRnB რეგისტრის WGMn3: WGMn2 თანრიგებთან ერთად განსაზღვრავენ Tn ტაიმერ/მთვლელის მუშაობის რეჟიმს (იხ.ცხრილი 5.6)

მოყვანილი ცხრილის მიხედვით, TCCR1A(TCCR3A) რეგისტრების უფროსი თანრიგები განკუთნილია შედარების სქემების OC გამოსასვლელების მდგომარეობის დასაპროგრამებლად. როგორც ზევით ითქვა, თითოეული აღნიშნული მთვლელი შეიცავს სამ შედარების რეგისტრს OCRnA, OCRnB, OCRnC. მთვლელის შედარება ხდება სამივე რეგიტრთან, რომელთა შედარების სქემებს აქვთ ცალცალკე OC გამოსასვლელები და მათი მდგომარეობის დაპროგრამება ტოლობის შემთხვევაში სრულდება აღნიშნული რეგისტრის თანრიგებში COMnA1: COMnA0; COMnB1: COMnB0; COMnC1: COMnC0.

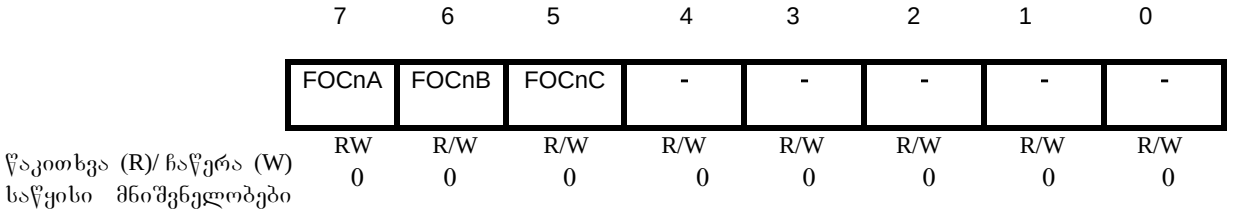
	7	6	5	4	3	2	1	0
	ICNCn	ICESn	-	WGMn3:	WGMn2	CSn2.	CSn1	CSn0
წაკითხვა (R)/ ჩაწერა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
საწყისი მნიშვნელობები	0	0	0	0	0	0	0	0

სურ.5.13. TCCR1B (TCCR3B) რეგისტრის ფორმატი

ცხრილი 5.11. TCCR1B (TCCR3B) რეგისტრის თანრიგები

თანრიგი	დასახელება	აღწერა
7	ICNCn	**დაპყრობის ბლოკის ხელშეშლის დათრგუნვის მართვის სქემა.** თუ თანრიგი ჩამოგდებულია - ნულშია, მაშინ ხელშეშლის დათრგუნვის სქემა გამორთულია (დაპყრობა ხორციელდება . პირველი აქტიური ფრონტით) თუ თანრიგი დაყენებულია ერთიანში, ხელშეშლის დათრგუნვის სქემა ჩართულია, მაშინ დაპყრობა ხორციელდება მხოლოდ ოთხ ერთნაირი ამორჩევით აქტიური სიგნალის ფრონტის შესაბამისად.
6	ICESn	**დაპყრობის სიგნალის აქტიური ფრონტის ამორჩევა.** თუ ICESn თანრიგი ჩამოგდებულია ნულში, მაშინ დაპყრობის რეგისტრში მთვლელი რეგისტრის შენახვა ხორციელდება სიგნალის კლებად ფრონტზე. თუ თანრიგი ერთიანშია მაშინ დაპყრობის რეგისტრში მთვლელი რეგისტრის შენახვა ხორციელდება სიგნალის მზარდი ფრონტით. მთვლელი რეგისტრის შენახვისას ერთდროულად ხდება TIFR(ETIFR) რეგისტრში ICFn წყვეტის ალმის დაყენება.
5	-	**გამოუყენებელია. . იკითხება როგორც ნოლი**
4,3	WGMn3: WGMn2	**ტაიმერ/მთვლელის მუშაობის რეჟიმი. T**CCRnA რეგისტრის WGMn1: WGMn0 თანრიგებთან ერთად განისაზღვრება Tn ტაიმერ/მთვლელის მუშაობის რეჟიმი (ცხრილი 5.11)
2....0	CSn2.... CSn0	**ტაქტური სიგნალების მართვა**. ეს თანრიგები განსაზღვრავს მიკროკონტროლერის ტაქტურ სიგნალების წყაროს.

უმცროსი ორი თანრიგი (WGMnA1: WGMnA0) განკუთვნილია მთვლელის თვლის რეჟიმის არჩევისათვის, ზევით განხილული 5.6.ცხრილიის თანახმად.



სურ. 5.14. TCCR1C(TCCR3C) რეგისტრის ფორმატი

ცხრილი 5.12. TCCR1C(TCCR3C) რეგისტრის თანრიგები

თანრიგი	დასახელება	აღწერა
7	FOCnA	**OCnx გამომყვანის იძულებითი მდგომარეობის შეცვლა.** იმ შემთხვევაში როდესაც FOCnx თანრიგში ერთიანია ჩაწერილი, მაშინ OCnx გამომყვანის მდგომარეობა იცვლება TCCRnA რეგისტრის COMn1x:COMn0x თანრიგების დაყენების შესაბამისად. ამ შემთხვევაში წყვეტა არ გენერირდება და ტაიმერის განულება (CTC რეჟიმში) არ ხდება. ეს ფუნქციები ხელმისაწვდომია მხოლოდ სამ რეჟიმში, რომლებიც არ გამოიყენება განივიმპულსური სიგნალების გენერაციის დროს. თანრიგის წაკითხვის დროს ყოველთვის ბრუნდება ნულში.
6	FOCnB	
5	FOCnC	
4.....0	–	არ გამოიყენება, იკითხება როგორც ნული

შენიშვნა: n=1 ან 3;

მიმართვა 16 თანრიგიან რეგისტრებთან

ყოველი 16 თანრიგიანი ტაიმერ/მთვლელების რეგისტრები ფიზიფურად განლაგებულია ორ რვა თანრიგიან რეგისტრში. შესაბამისად მასთან მიმართვისთვის საჭიროა შესრულდეს წაკითხვის ან ჩაწერის ორი ოპერაცია. იმისათვის, რომ 16 თანრიგიან რეგისტრში ორივე ბაიტის ჩაწერა ან ამოკითხვა განხორციელდეს ერთდროულად, ყოველ ტაიმერ/მთვლელს გააჩნია რვა თანრიგიანი TEMP რეგისტრი, განკუთვნილი უფროსი ბაიტის მნიშვნელობის შენახვისათვის (ეს რეგისტრი გამოიყენება მხოლოდ პროცესორის მიერ და პროგრამულად მიუწვდომელია).

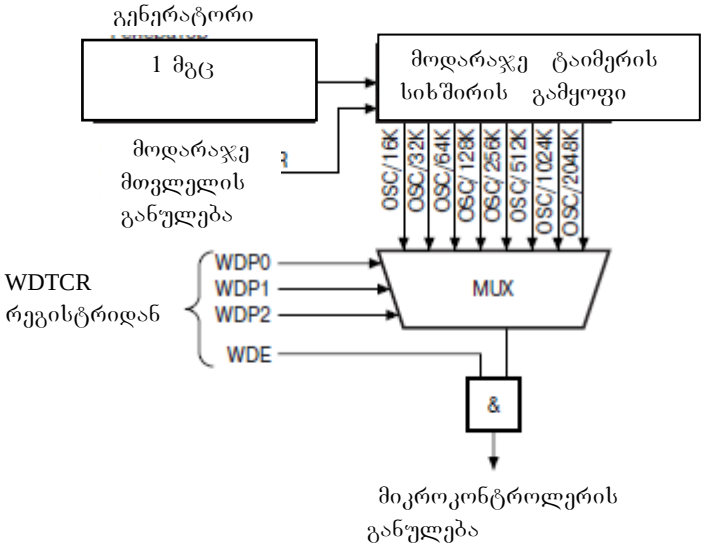
16 თანრიგიანი რეგისტრის ჩაწერის ციკლის შესრულებისთვის აუცილებელია პირველად ჩაიტვირთოს უფროსი ბაიტი TEMP რეგისტრში. უმცროსი ბაიტის მომდევნო ჩაწერის დროს ის გაერთიანდება TEMP რეგისტრის შემცველობასთან და ორივე ბაიტი ერთდროულად (ერთი და იგივე მანქანურ ციკლში) ჩაიწერება 16 თანრიგიან რეგისტრში.

იმ შემთხვევაში, როდესაც სრულდება ტაიმერ/მთვლელის 16 თანრიგიან რეგისტრთან მიკითხვის ციკლი, წყვეტა აკრძალული უნდა იყოს. წინააღმდეგ

შემთხვევაში თუ წყვეტა განხორციელდება ორ ბრძანებას შორის 16 თანრიგიან რეგისტრთან მიმართვისას, და წყვეტის ქვეპროგრამის დამუშავების დროს ასევე მოხდება მიმართვა ტაიმერ/მთვლელის ერთერთ 16 თანრიგიან რეგისტრთან, TEMP რეგისტრის შემცველობა შეიცვლება, რის გამოც მმართველი პროგრამის 16 თანრიგიან რეგისტრთან მიმართვის დროს შედეგი იქნება არასწორი.

5.8.მოდარაჯე ტაიმერი

მიკროკონტროლერ Atmega128-ს შემადგენლობაში არის აგრეთვე მოდარაჯე ტაიმერი, რომლის დანიშნულებას წარმოადგენს პროგრამის შესრულების პროცესში დაიცვას მიკროკონტროლერი უწესივრობისაგან. მოდარაჯე ტაიმერის სტრუქტურული სქემა მოცემულია 5.15.სურ.-ზე.

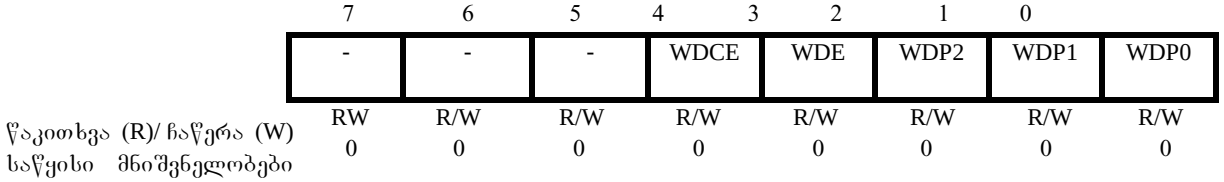


სურ.5.15. მოდარაჯე მთვლელის სტრუქტურა

მოდარაჯე ტაიმერს გააჩნია დამოუკიდებელი ტაქტური გენერატორი, ეს იმით არის განპირობებული, რომ ის მუშაობს იმ დროსაც, როცა მიკროკონტროლერი იმყოფება ნებისმიერ მიძინებულ რეჟიმში. აღნიშნული გენერატორის სიხშირის ძირითადი მნიშვნელობა ტოლია 1 მგჰც-ის, როცა კვების ძაბვა V_{CC}=+5.0 ვ.

თუ მოდარაჯე ტაიმერი ჩართულია, მაშინ გარკვეული დროითი შუალედის შემდეგ, რომელიც ტოლია მისი პერიოდის (თვლის დროის ინტერვალი 0-დან მაქსიმალურ მნიშვნელობამდე), ის ასრულებს მიკროკონტროლერის განულებას. იმისთვის, რომ თავიდან ავიცილოთ მიკროკონტროლერის განულება პროგრამის ნორმალური შესრულების დროს, აუცილებელია მოდარაჯე ტაიმერი პერიოდულად განულდეს გარკვეული დროითი შუალედის შემდეგ, რომელიც ნაკლებია მის პერიოდზე. მოდარაჯე ტაიმერის განულება სრულდება WDR ბრძანების მეშვეობით.

მოდარაჯე ტაიმერის მართვისთვის გამოიყენება WDTCR რეგისტრი. ამ რეგისტრის ფორმატი მოყვანილია 5.16.სურ.-ზე, ხოლო მისი თანრიგების აღწერა მოცემულია 5.12.ცხრილში. მოდარაჯე ტაიმერის ჩართვა/გამორთვისთვის გამოიყენება WDTCR რეგისტრის WDE და WDCE ორი თანრიგი. WDCE თანრიგში 1-ის ჩაწერით მოდარაჯე



სურ.5.16. WDTCR რეგისტრის ფორმატი

ცხრილი 5.12. WDTCR რეგისტრის თანრიგები

თანრიგები	დასახელება	აღწერა
7.....5	-	დარეზერვირებულია და ამოიკითხება "ნული"
4	WDCE	მოდარაჯე ტაიმერის მდგომარეობის შეცვლის ნებადართვა. (ნებადართვა მოდარაჯე ტაიმერის გამორთვაზე)
3	WDE	მოდარაჯე ტაიმერის ნებადართვა ("1"– ჩართულია)
2	WDP2	მოდარაჯე ტაიმერის წინაგამყოფის სიხშირის გაყოფის კოეფიციენტი
1	WDP1	
0	WDP0	

მთვლელი გადადის მუშა მდგომარეობაში, WDE თანრიგის საშუალებით კი ხდება მისი ჩართვა/გამორთვა (1-ის ჩაწერით იწყებს თვლას, 0-ის ჩაწერით – წვეტს თვლას).

მოდარაჯე ტაიმერის თვლის პერიოდი განისაზღვრება WDTCR რეგისტრის WDP2.....WDP0 თანრიგებში შესაბამისი კოდის ჩაწერით, რომლებიც მოცემულია 5.13.ცხრილში.

ცხრილი 5.13. მოდარაჯე ტაიმერის თვლის პერიოდები

WDP2	WDP1	WDP0	გენერატორის ტაქტების რაოდენობა	ტაიმ-აუტის დადგომის პერიოდი (ტიპობრივი მნიშვნელობები) [მწ]	
				V_{CC}=3.0 ვ	V_{CC}=5.0 ვ
0	0	0	16K(16384)	17	16
0	0	1	32K(32768)	34	33
0	1	0	64K(65536)	69	65
0	1	1	128K(131072)	140	130
1	0	0	256K(262144)	270	260
1	0	1	512K(524288)	550	520
1	1	0	1024K(1048576)	1100	1000

იმისათვის, რომ გამოირიცხოს მიკროკონტროლერის წინასწარი დაუგეგმავი განულება, მოდარაჯე ტაიმერის პერიოდის ცვლილების დროს, აუცილებელია WDP2....WDP0 თანრიგებში ჩაწერის განმავლობაში აიკრძალოს მოდარაჯე ტაიმერი ან მოხდეს მისი განულება

თავი 6

ანალოგური კომპარატორი

Atmega 128 მოდულის შემადგენლობაში შედის ანალოგური კომპარატორი. ჩართულ მდგომარეობაში კომპარატორი გვაძლევს საშუალებას შევადაროთ ძაბვების მნიშვნელობები, რომლებიც მიკროკონტროლერის ორ შესასვლელზე გვაქვს. შედარების შედეგია ლოგიკური მნიშვნელობა (ლოგიკური 0 ან 1), რომელიც შეიძლება წაკითხული იყოს პროგრამიდან. შედარების შედეგის მიხედვით შესაძლებელია წყვეტის გენერირება, ასევე შესაძლოა განხორციელდეს T1 ტაიმერ/მთვლელის მდგომარეობის დაპყრობა. უკანასკნელი ფუნქცია გვაძლევს საშუალებას კონკრეტულად გავზომოთ ანალოგური სიგნალების ხანგრძლივობა.

კომპარატორის მიერ გამოყენებული გამომყვანები წარმოადგენს საერთო დანიშნულების შეტანა/გამოტანის პორტების კონტაქტებს. კომპარატორის მიერ გამოყენებული გამომყვანები ნაჩვენებია 6.1. ცხრილში.

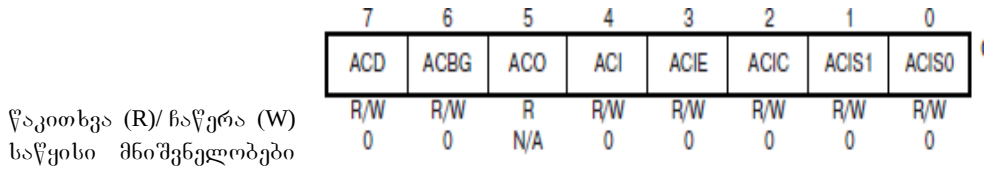
ცხრილი 6.1. ანალოგური კომპარატორის მიერ გამოყენებული გამომყვანები

კომპარატორის შესასვლელების დასახელება	პორტების გამომყვანები	დანიშნულება
AIN0	PE2	არა ინვერსული შესასვლელი
AIN1	PE3	ინვერსული შესასვლელი

იმისთვის, რომ მითითებული გამომყვანები გამოვიყენოთ ანალოგური კომპარატორისთვის, საჭიროა ისინი დაკონფიგურირდეს, როგორც შესასვლელები (DDRx რეგისტრის შესაბამისი თანრიგები დავაყენოთ ნულიან მდგომარეობაში).

6.1. კომპარატორის მართვა

კომპარატორის მართვა და მისი მდგომარეობის კონტროლი ხორციელდება ACSR რეგისტრის დახმარებით. აღნიშნული რეგისტრის ფორმატი მოყვანილია 6.1.სურ-ზე, ხოლო თანრიგების დანიშნულების მოკლე აღწერა 6.2. ცხრილში



სურ.6.1. ACSR რეგისტრის ფორმატი

ცხრილი 6.2. ACSR რეგისტრის თანრიგების აღწერა

თანრიგი	დასახელება	აღწერა
7	ACD	კომპარატორის გამორთვა ნოლის დროს ჩართულია, ერთიანის დროს გამორთული.
6	ACBG	კომპარატორის არაინვერსულ შესასვლელთან შიდა საყრდენ ძაბვის წყაროს მიერთება (ერთიანი-მიერთებულია, ნულიანი-მიერთებული არ არის).
5	ACO	შედარების შედეგი (კომპარატორის გამოსას-ვლელი)
4	ACI	კომპარატორიდან წყვეტის ალამი
3	ACIE	კომპარატორიდან წყვეტის ნებადართვა
2	ACIC	კომპარატორის მიერთება ტაიმერ/მთვლელ T1 დაპყრობის სქემასთან (ერთიანი-მიერთებულია, ნულიანი - გამორთულია)
1,0	ACIS1 : ACIS0	კომპარატორიდან წყვეტის გენერირების პირობა

იმ შემთხვევაში როდესაც ძაბვა კომპარატორის AIN0 გამომყვანზე (არაინვერსული შესასვლელი) მეტია AIN1 გამომყვანზე (ინვერსული შესასვლელი) მიწოდებული ძაბვის მნიშვნელობის, შედარების შედეგი ასახული იქნება ACO გამოსასვლელის მდგომარეობის ცვლილებით ლოგიკური ერთიანიდან, ლოგიკური ნულით ან პირიქით. კომპარატორის გამოსასვლელის მნიშვნელობა შეინახება ACSR რეგისტრის ACO თანრიგში.

ACD თანრიგი პასუხისმგებელია კომპარატორის ჩართვასა და გამორთვაზე (1-ის ჩაწერით კომპარატორი გამოირთვება, 0-ის ჩაწერით - ჩაირთვება). რამდენადაც მიკროკონტროლერზე ძაბვის მიწოდებისას ხორციელდება ACSR რეგისტრის ყველა თანრიგის განულება, კომპარატორი ავტომატურად ჩაირთვება მიკროკონტროლერის ჩართვის შემთხვევაში. ამ თანრიგის მნიშვნელობის შეცვლის დროს საჭიროა აიკრძალოს წყვეტა კომპარატორიდან.

როგორც უკვე იყო აღნიშნული შედარების შედეგის საფუძველზე კომპარატორის სქემას შეუძლია მოახდინოს წყვეტის მოთხოვნის გენერაცია. თუ კომპარატორის გამოსასვლელის მდგომარეობა (ACO თანრიგი) შეიცვალა, მოხდება ACSR რეგისტრის ACI თანრიგის წყვეტის ალმის დაყენება, რომელიც ანხორციელებს წყვეტის მოთხოვნის გენერაციას. ისევე როგორც სხვა წყვეტების დროს, ეს ალამი განულდება აპარატურულად, როდესაც გაეშვება წყვეტის დამუშავების ქვეპროგრამა ან პროგრამულად, მასში ლოგიკური ერთიანის ჩაწერით. წყვეტის ნებადართვისათვის აუცილებელია მოხდეს ACSR რეგისტრის ACIE თანრიგის და SREG რეგისტრის I ალამის ერთიანში დაყენება.

ACO გამოსასვლელის მდგომარეობის როგორი ცვლილება გამოიწვევს წყვეტას, დამოკიდებულია ACSR რეგისტრის ACIS1:ACIS0 თანრიგებში ჩაწერილ კოდზე 6.3.ცხრილის შესაბამისად. ამ თანრიგებში კოდის ჩაწერის დროს წყვეტა კომპარატორიდან უნდა აიკრძალოს.

ცხრილი 6.3. კომპარატორიდან წყვეტის მოთხოვნის გენერირების პირობე

AACIS1	ACIS0	პირობა
0	0	კომპარატორის გამოსასვლელის მდგომარეობის ნებისმიერი ცვლილება
0	1	დარეზერვირებულია
1	0	კომპარატორის გამოსასვლელის მდგომარეობის ცვლა “0” დან “1”-ით
1	1	კომპარატორის გამოსასვლელის მდგომარეობის ცვლა “1” დან “0”-ით

გარდა წყვეტის გენერაციისა, კომპარატორს ასევე შეუძლია T1 ტაიმერ/მთვლელის დაპყრობის სქემის მართვა. ამისათვის ACSR რეგისტრის ACIC თანრიგი უნდა დაყენდეს “1”-ში. შედეგად კომპარატორის გამოსასვლელი მიუერთდება დაპყრობის სქემის შესასვლელს მიკროკონტროლერის ICP1 გამომყვანის ნაცვლად. თუ ACIC თანრიგი განულებულია, კომპარატორი გამორთულია ტაიმერ/მთვლელის დაპყრობის სქემიდან.

კომპარატორს შეუძლია შეადაროს სიგნალები არა მხოლოდ AIN0 და AIN1 გამოსასვლელებზე. მიკროკონტროლერის AIN0 გამოსასვლელის მაგივრად შესაძლებელია მიუერთდეს ძაბვის 1.22ვ. მნიშვნელობის შიგა წყარო. ამისათვის საჭიროა ACSR რეგისტრის ACBG თანრიგში ჩაიწეროს “1”.

თავი 7

ანალოგურ-ციფრული გარდამქმნელი

ანალოგურ-ციფრული გარდამქმნელი (აცგ,ინგ. Analog-to-digital converter, ADC) არის მოწყობილობა, რომელიც შესასვლელ ანალოგურ სიგნალს გარდაქმნის დისკრეტულ სიგნალად (ორობით კოდად.). უკუ გარდაქმნას ანხორციელებს ცაგ (ციფრო - ანალოგური გარდამქნელი, DAC). მარტივი აცგ არის კომპარატორი.

გარჩევითობა

აცგ-ს გარჩევითობა -- ანალოგური სიგნალის სიდიდის მინიმალური ცვლილებაა, რომლითაც განსხვავდება გარდაქმნის შედეგად მიღებული მეზობელი ორობითი კოდები ერთმანეთისაგან. იგი დამოკიდებულია ამ კოდების თანრიგიანობაზე.

აცგ-ს თანრიგიანობა განსაზღვრავს ორობითი კოდების მნიშვნელობების რაოდენობას, რომელიც გარდამქმნელს შეუძლია გასცეს გამოსასვლელზე. აცგ-ს გარდაქმნის შედეგები წარმოდგება ბიტებში. მაგალითად, ორობითი რვა თანრიგიან აცგ-ს აქვს უნარი გასცეს 256 დისკრეტული მნიშვნელობა (0.....255), გამომდინარე იქიდან, რომ $2^8 = 256$. გარჩევითობა ძაბვის მიხედვით ტოლია აცგ-ს შესასვლელზე მიწოდებული მაქსიმალური და მინიმალური ძაბვების სხვაობის, გაყოფილს გამოსასვლელი დისკრეტული მნიშვნელობების რაოდენობაზე.

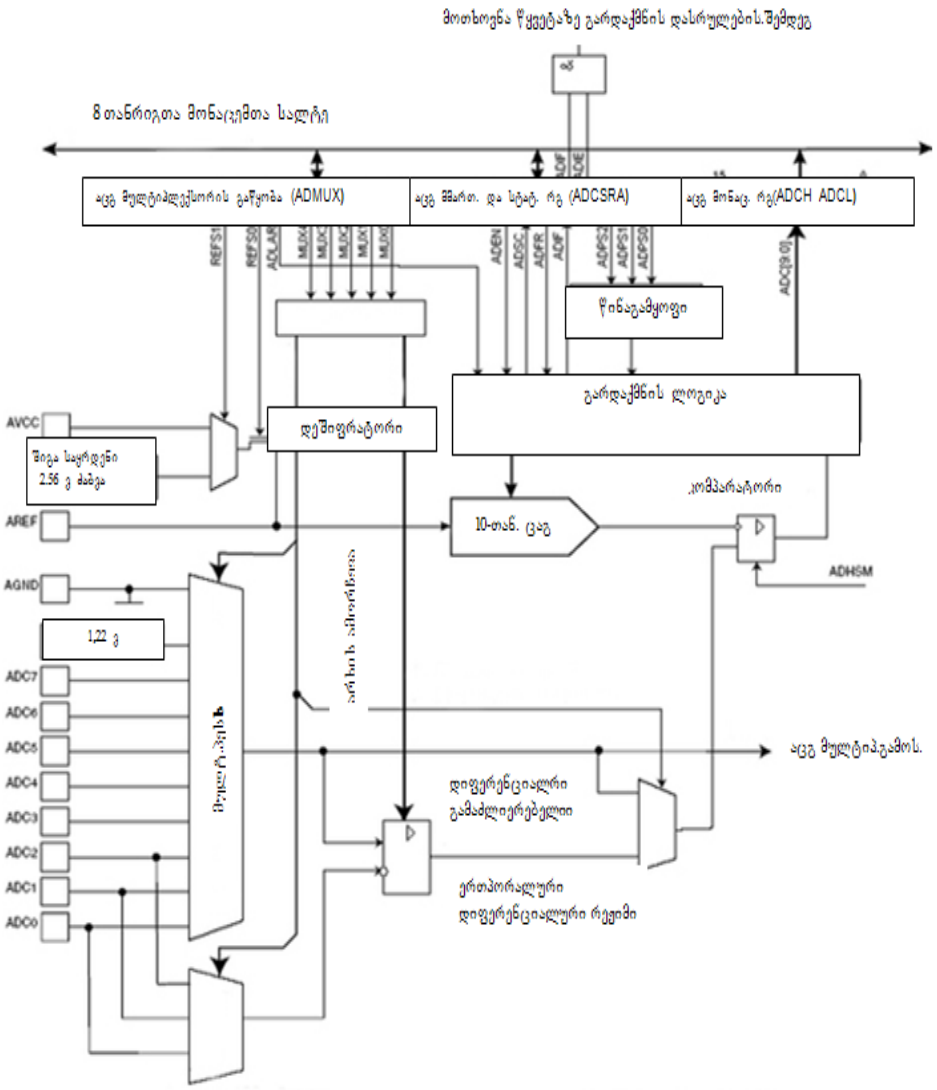
მაგალითად:

- ვთქვათ, შესავალი მნიშვნელობის დიაპაზონი ტოლია 0 დან 10 ვოლტამდე;
- აცგ-ს გამოსასვლელ ორობითი თანრიგთა რაოდენობა 12 ბიტი, რომლის საშუალებით ფორმირდება 2^{12} =4096 კოდი;
- აცგ გარჩევითობა ძაბვის მიხედვით იქნება: (10-00)/4096=0,00244ვ= =2,44მვოლტი.

Atmega 128 მიკროკონტროლერის შემადგენლობაში შედის მიმდევრობითი მიახლოების 10 თანრიგიანი აცგ მოდული. 7.1.სურ-ზე ნაჩვენებია მიმდევრობითი მიახლოების აცგ-ს ბლოკ-სქემა.

ანალოგური-ციფრული გარდამქმნელის მახასიათებლები

1. 10 თანრიგა გარჩევითობა;
2. გარდასახვის აბსოლუტური ცდომილება ±2 მლვ;
3. გარდაქმნის დრო 65-260 მკწმ;
4 გარდაქმნის სიხშირე 15 ათასამდე წმ-ში;
5. მულტიპლექსორის 8 ერთპოლარული შესასვლელი;
6. აცგ შესასვლელი ძაბვის დიაპაზონი 0....Vcc;
7. შიგა საყრდენი 2.56 ვ ამორჩევის ძაბვა;
8. ერთეულოვანი გარდაქმნის და ავტომატური გაშვების რეჟიმი;
9. აცგ მიერ გარდაქმნის დამთავრების შემდეგ წყვეტა;
10. ძილის რეჟიმში ხმაურის დათრგუნვის რეჟიმი.



სურ.7.1. აცგ-ს სტრუქტურული სქემა

7.1.ანალოგური-ციფრული გარდამქმნელის მოქმედების პრინციპი

მოცემულ ტიპის აცგ-ს საფუძვლად უდევს მიმდევრობითი მიახლოების გარდაქმნის სპეციალური რეგისტრი. ციკლის დასაწყისში ამ რეგისტრის ყველა თანრიგი დაყენებულია ნულის მდგომარეობაში, გარდა პირველი (უფროსი) თანრიგისა.

აღნიშნული შემთხვევისათვის 10 თანრიგიანი აცგ-ს მიმდევრობითი მიახლოების რეგისტრში დაყენებულია კოდი “1000000000” (რაც შეესაბამება აცგ შემავალი დიაპაზონის ნახევარს). ეს კოდი რეგისტრის გამოსასვლელიდან მიეწოდება ცაგ შესასვლელს, რომელიც თავის მხრივ გარდაქმნის მას ძაბვად და ეს უკანასკნელი გადაეცემა კომპარატორის ერთ-ერთ შესასვლელს, რომლის მეორე შესასვლელზე მიწოდებულია გარდასასახი ძაბვა. იმ შემთხვევაში, თუ შემავალი ძაბვა მცირეა აცგ შემავალი დიაპაზონის ნახევარზე (ეს იმის მაუწყებელია, რომ საწყისი კოდის მნიშვნელობა არის დიდი შემავალი ძაბვისათვის და საჭიროა მისი შემცირება)

კომპარატორის გამოსასვლელი მიიღებს ლოგიკური ნულიანის მნიშვნელობას, შედეგად მიმდევრობითი მიახლოების რეგისტრში ჩაიწერება შემცირებული კოდი 0100000000, რაც გამოიწვევს ცაგ გამოსასვლელზე ძაბვის ცვლილებას, რომელიც ისევ მიეწოდება კომპარატორს. იმ შემთხვევაში თუ კომპარატორის გამოსასვლელი ინარჩუნებს "ნულიანის" მდგომარეობას (ე.ი შემავალი ძაბვა კვლავ მცირეა მიღებულ კოდზე), მაშინ რეგისტრი გადაირთვება "00100000" მდგომარეობაში. თუ გარდაქმნის მომდევნო ბიჯზე ცაგ-ის გამომავალი ძაბვა აღმოჩნდა მცირე, ვიდრე შემავალი გასაზომი ძაბვა, კომპარატორის გამოსასვლელი გადაირთვება ლოგიკური ერთიანის მდგომარეობაში. ეს მითითებაა იმისა, რომ მიმდევრობითი მიახლოების რეგისტრის მეორე თანრიგში დარჩეს ლოგიკური ერთიანი და ჩაიწეროს ლოგიკური ერთიანი მესამე თანრიგში. მუშაობის აღწერილი ალგორითმი შემდგომში კვლავ მეორდება ბოლო თანრიგამდე. ამრიგად, აცგ მიმდევრობითი მიახლოების მეთოდი მოითხოვს ერთ შიგა ტაქტს გარდაქმნის თითოული თანრიგისათვის ან N ტაქტს N თანრიგა გარდაქმნისათვის.

აცგ მოდულს აქვს 8 შესასვლელი - ADC0...ADC7, რომელთაგან ერთ-ერთის მიერთება აცგ-თან გარდაქმნისათვის სრულდება ანალოგური მულტიპლექსორის საშუალებით.

ანალოგური შეტანის არხი და დიფერენციული მაძლიერებლის კასკადი ამოირჩევა ADMUX რეგისტრის MUX0-MUX2 ბიტებში შესაბამისი კოდის ჩაწერით. შესასვლელად შესაძლებელია არჩეული იყოს აგრეთვე GND და 1,22ვ ფიქსირებული წყაროს გამოსასვლელი.

მუშაობის პროცესში აცგ-ს შეუძლია იფუნქციონიროს ორ რეჟიმში:

- ერთეულოვანი გარდაქმნის რეჟიმი, როდესაც ცალკეულ შემავალ არხიდან გარდაქმნის ყოველი გაშვების ინიცირება ხორციელდება მომხმარებლის მიერ.
- უწყვეტი გარდაქმნის რეჟიმი, როდესაც გარდაქმნის გაშვების ინიცირება ხორციელდება რამდენიმე არხისთვის მიმდევრობით უწყვეტად გარკვეული დროითი ინტერვალით.

ერთეულოვანი გარდაქმნის რეჟიმში ყოველი ახალი გარდაქმნა ხორციელდება აცგ-ს ADCSRA რეგისტრის ADSC (გარდაქმნის დაწყების) თანრიგში ლოგიკური ერთიანის ჩაწერით. მოცემული ბიტი გარდაქმნის პროცესში რჩება ერთიანის მდგომარეობაში და განულდება გარდაქმნის დამთავრების შემდეგ.

უწყვეტი გარდაქმნის რეჟიმში აცგ უწყვეტად, ერთმანეთის მიყოლებით, ასრულებს ანალოგური სიგნალების ციფრულ ფორმაში გადაყვანას და ყოველი გარდასახვის ბოლოს აახლებს აცგ მონაცემების რეგისტრს. პირველი გარდაქმნა ინიცირდება ADSCRA რეგისტრის ADSC თანრიგში ლოგიკური ერთიანის ჩაწერით, რომლის მნიშვნელობა შენარჩუნებული იქნება გარდაქმნის მთელი პროცესის განმავლობაში.

რეჟიმების დაყენება სრულდება ADCSRA რეგისტრის ADFR ბიტის საშუალებით (ADFR=0 – ერთეული გარდაქმნის რეჟიმი, ADFR=1- უწყვეტი გარდაქმნის რეჟიმი).

შესასვლელი ძაბვის მინიმალური მნიშვნელობა ტოლია ნულის, ხოლო მაქსიმალური მნიშვნელობა - არა უმეტეს კვების ძაბვის მნიშვნელობისა (აღნიშნული მიკროკონტროლერისათვის V_{CC}=+5ვ).

მეტად მნიშვნელოვანია ე.წ. საყრდენი ძაბვის (სძწ) არჩევა, რომელიც მიეწოდება ციფრულ – ანალოგურ გარდამქმნელს და განსაზღვრავს მის მიერ გარდაქმნილი ძაბვის მნიშვნელობას. იგი შესაძლებელია მიუერთდეს აცგ-ს გარედან

AREF შესასვლელით (რეკომენდებულია 2.0 ვ - V_{CC} დიაპაზონში) ან შიგა ₹ყაროდან (2,56 ვ).

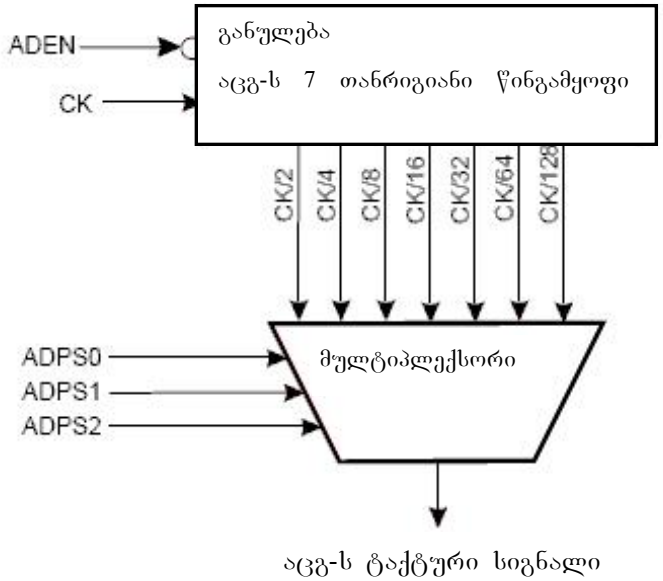
აცგ მუშაობის ნების დართვა ხორციელდება ADCSRA რეგისტრის ADEN თანრიგის დაყენებით. როდესაც ADEN = 0, მაშინ აცგ გამოირთვება და დენს არ მოიხმარს, ამიტომ "ძილის" ეკონომიურ რეჟიმში გადაყვანის შემთხვევაში რეკომენდებულია წინასწარ გაითიშოს აცგ.

გარდაქმნის შედეგად აცგ აფორმირებს ათ თანრიგა კოდს, რომელიც თავსდება აცგ მონაცემთა 16 თანრიგა რეგისტრში (იგი შედგება შეყვანა-გამოყვანის ორი რეგისტრისაგან - ADCH და ADCL). გარდაქმნის შედეგი შეიძლება ჩაიწეროს უმცროს 10 თანრიგში (მარჯვნივ გათანაბრებით), ან უფროს 10 თანრიგში (მარცხნივ გათანაბრებით) ADMUX რეგისტრის ADLAR ბიტის გამოყენებით.

გარდაქმნის დამთავრების შემდეგ აცგ გენერირებს მოთხოვნას წყვეტაზე, რომელიც ფიქსირდება ADCSRA რეგისტრის ADIF თანრიგში ერთიანის დაყენებით. წყვეტის ნების დართვა/აკრძალვისათვის ამავე რეგისტრში გათვალისწინებულია ADIE თანრიგი (ალამი).

აცგ შეიცავს წინაგამყოფის ბლოკს, რომლის საშუალებით შეგვიძლია აცგ-ს მივაწოდოთ ტაქტირებისათვის საჭირო, სხვადასხვა სიხშირის იმპულსები მიკრო-კონტროლერის ძირითადი სატაქტო იმპულსების სიხშირის შემცირებით (დაყოფით). 7.2. სურ.-ზე ნაჩვენებია წინაგამყოფის სტრუქტურული სქემა.

7.2.სურ.-ის მიხედვით აცგ წინაგამყოფს აქვს რამდენიმე გამოსასვლელი, რომლებზეც CK ძირითადი სატაქტო სიგნალების საფუძველზე ფორმირდება სხვადასხვა სიხშირის სიგნალები, რომელთა მნიშვნელობები განისაზღვრება შესაბამისი დაყოფის კოეფიციენტებით. მულტიპლექსორის საშუალებით ხდება ერთ-ერთი მათგანის მიერთება გამოსასვლელთან და მიეწოდება აცგ-ს. წინაგამყოფის აღნიშნული გამოსასვლელების არჩევა ხდება კოდით, რომელიც ჩაიწერება ADCSRA რეგისტრის ADPS0- ADPS2 თანრიგებში



სურ.7.2. აცგ წინაგამყოფის სტრუქტურა

აცგ წინაგამყოფის გაყოფის კოეფიციენტის მნიშვნელობები განისაზღვრება 7.1. ცხრილში.

ცხრილი 7.1. აცგ წინაგამყოფის გაყოფის კოეფიციენტის მნიშვნელობები

ADPS2	ADPS1	ADPS0	გაყოფის კოეფიციენტი
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

წინა გამყოფის მუშაობა იწყება ADCSRA რეგისტრის ADEN ბიტის ერთიანში დაყენებით, გამორთვა –ნულში დაყენებით.

7.2. აცგ-ს მართვის რეგისტრები

აცგ-ს მართვა ხორციელდება მის მართვის რეგისტრებში სხვადასხვა დანიშნულების თანრიგების (ალმების) დაყენებით.

რეგისტრები, რომლებიც გამოიყენება აცგ მოდულის მართვისათვის ნაჩვენებია 7.2. ცხრილში.

ცხრილი 7.2. აცგ მოდულის მმართველი რეგისტრები

რეგისტრი	აღწერა
ADCSRA	მართვის და მდგომარეობის A რეგისტრი
ADMUX	მულტიპლექსორის მართვის რეგისტრი
SFIOR	სპეციალური ფუნქციების რეგისტრი

ADCSRA რეგისტრების ფორმატი მოცემულია 7.3.სურ-ზე, ხოლო ADCSRA რეგისტრის თანრიგების აღწერა - 7.3. ცხრილში.

	7	6	5	4	3	2	1	0
	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0
წაკითხვა(R)/ჩაწერა(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მდგომარეობა	0	0	0	0	0	0	0	0

სურ.7.3. ADCSRA რეგისტრის ფორმატი

ცხრილი 7.3. ADCSRA რეგისტრის თანრიგების დანიშნულება

თანრიგები	დასახელება	ჩაწერა
7	ADEN	აცგ-ს ნებისდართვა (1-ჩართულია, 0-გამორთულია)
6	ADSC	გარდაქმნის გაშვება (1- გარდაქმნის დწყება)
5	ADFR	აცგ მუშაობის რეჟიმის ამორჩევა
4	ADIF	აცგ-დან წყვეტის მოთხოვნის ალამი
3	ADIE	აცგ-დან წყვეტის ნებადართვა
2..0	ADPS2:ADPS0	გარდაქმნის სიხშირის ამორჩევა (იხ.ცხრილი 7.1)

ADMUX რეგისტრების ფორმატი მოცემულია 7.4.სურ-ზე. ხოლო მათი თანრიგების დანიშნულება - 7.4. ცხრილში.

	7	6	5	4	3	2	1	0
	REFS1	REFS0	ADLAR			-	MUX2	MUX1	MUX0
წაკითხვის(R),ჩაწერის(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	0	0	0	0	0	0	0	0

სურ.7.4. ADMUX რეგისტრის ფორმატი

როგორც ზევით იყო ნათქვამი, ამ რეგისტრის MUX2-MUX0 თანრიგებში ჩაწერილი კოდები განსაზღვრავს აქტიური არხის ნომერს (ანალოგური შესასვლელი, რომელიც მიერთებულია აცგ-სთან) 7.5. ცხრილის თანახმად.

ცხრილი 7.5. შესასვლელი მულტიპლექსორის მართვა

MUX2…0	ლოგური შესასვლელები
000	ADC0 (PB5)
001	ADC1 (PB2)
010	ADC2 (PB3)
011	ADC3 (PB4)
100	არ გამოიყენებიან

შენიშვნა: ფრჩხილებში მითითებულია პორტების გამომყვანები, რომლებიც მიერთებულია აცგ-ს შესასვლელებთან.

MUX2.... MUX0 თანრიგების მდგომარეობა შესაძლებელია შეიცვალოს ნებისმიერ დროს, მაგრამ თუ ეს მოხდება გარდასახვის ციკლის პროცესში, არხის ცვლილება შესრულდება გარდასახვის დამთავრების შემდეგ. ამის გამო უწყვეტი გარდასახვის რეჟიმის შემთხვევაში შესაძლებელია არხების ადვილად სკანირება, რაც მოცემულ შემთხვევაში გულისხმობს რამდენიმე არხის სიგნალების მიმდევრობით (ციკლურად ან მოცემული პროგრამით) გარდასახვას.

ასევე იყო აღნიშნული, რომ აცგ მოდულში შეიძლება გამოვიყენოთ საყრდენი ძაბვის სხვადასხვა წყარო – ან გარედან AREF შესასვლელით, ან შიგა 2,56ვ, ან V_{CC} (კვების ძაბვა)..

კონკრეტული სძწ წყაროს ამორჩევა ხორციელდება ADMUX რეგისტრის REFS1: REFS0 თანრიგების დახმარებით (7.6.ცხრილი).

ცხრილი 7.3. საყრდენი ძაბვის წყაროს ამორჩევა.

REFSI	REFS0	საყრდენი ძაბვის წყარო
0	0	კვების ძაბვა V_{cc}, შიდა სძწ გამორთულია.
0	1	გარე სძწ, მიერთებულია AREF გამომყვანთან, შიდა სძწ გამორთულია.
1	0	შიგა სძწ 2,56ვ გამორთულია.
1	1	შიგა სძწ 2.56ვ. მიერთებულია.

SFIOR რეგისტრის ADHSM თანრიგის დაყენებით ხდება ცაგ-ის ჩართვა/გამორთვა. SFI0R რეგისტრის ფორმატი მოცემულია 7.5.სურ-ზე (რეგისტრის თანრიგები, რომლებიც ამ შემთხვევაში არ გამოიყენება ნახაზზე აღნიშნულია "X" სიმბოლოთი)

7	6	5	4	3	2	1	0
X	X	X	ADHSM	X	X	X	X
აკითხვის(R),ჩაწერის(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	0	0	0	0	0	0	0	0

სურ. 7.5. SFI0R რეგისტრის ფორმატი

7.3. აცგ მოდულის ფუნქციონირება

ყოველი გარდასახვის დაწყება ერთეულოვან გარდასახვის რეჟიმში, ასევე პირველი გარდასახვის დაწყება უწყვეტ გარდასახვის რეჯიმში სრულდება ADCSRA რეგისტრის ADSC ბიტის ერთიანში დაყენებით.

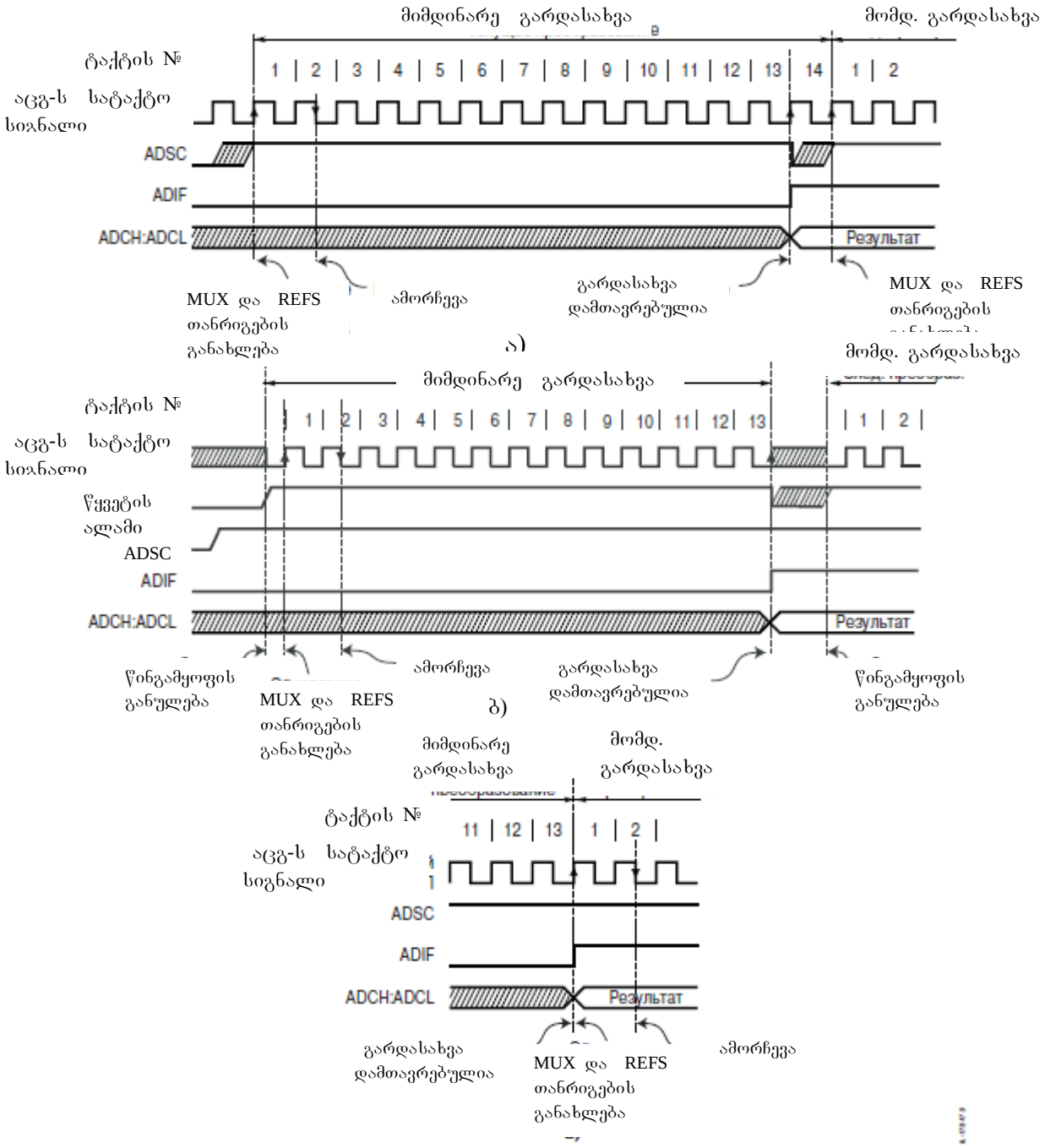
გარდაქმნის ციკლის ხანგრძლივობა შეადგენს 13 ტაქტს; შესასვლელი სიგნა- ლის ამორჩევა და დამახსოვრება ხორციელდება პირველი ტაქტის განმავლობაში. 13 ტაქტის შემდეგ გარდასახვა მთავრდება, ADSC ბიტი აპარატურულად დგება "0" მდგომარეობაში (ერთეულოვანი გარდასახვის რეჟიმში) და შედეგი ინახება აცგ-ს მონაცემთა რეგისტრში.

ერთდროულად ADCSRA რეგისტრის ADIF წყვეტის ალამში იწერება "1" და გენერირდება წყვეტის მოთხოვნა. როგორც სხვა წყვეტის ალმები, ADIF ალამი აპარატურულად განულდება აცგ-ს წყვეტის დამმუშავებელი პროგრამის გაშვების შემდეგ ან პროგრამულად მასში 1-ის ჩაწერით. წყვეტის ნების დართვა ხორციელდება ADCSRA რეგისტრის ADIE თანრიგში ერთიანის ჩაწერით.

თუ აცგ მუშაობს უწყვეტი გარდაქმნის რეჟიმში, ახალი ციკლი იწყება უმალვე წინა გარდასახვის შენახვის შემდეგ. ნათქვამის საილუსტრაციოდ 7.6.სურ.- ზე ნაჩვენებია გარდასახვის დროითი დიაგრამა.

გარდაქმნის შედეგი

გარდაქმნის დამთავრების შემდეგ (ADCSR რეგისტრის ADIF ალამის "1"-ში დაყენების დროს) მისი შედეგი შეინახება აცგ მონაცემთა რეგისტრში. რამდენადაც აცგ 10 თანრიგიანია, ეს რეგისტრი ფიზიკურად განთავსებულია შეტანა/გამოტანის ორ ADCH:ADCL 8 თანრიგა რეგისტრში, რომლებიდანაც შესაძლებელია მხოლოდ წაკითვა. მიკროკონტროლერის ჩართვის შემდეგ მას აქვს $0000 მნიშვნელობა.



სურ.7.6. ა) დროითი დიაგრამა აცგ მუშაობის ერთეულოვანი გარდაქმნის დროს, ბ)გაშვების რეჟიმი წყვეტის დროს, ვ) უწყვეტი გარდაქმნის რეჟიმი

უთქმელობით ხდება გარდაქმნის შედეგის ჩაწერა მარჯვნივ გათანაბრებით, ანუ შედეგი იკავებს ADCL რეგისტრს და ADCH რეგისტრის ორ თანრიგს (ADCH რეგისტრის უფროსი 6 თანრიგი არანიშნადია), მაგრამ შესაძლებელია შედეგის ჩაწერა მარცხნივ გათანაბრებით (ამ შემთხვევაში ADCL რეგისტრის უმცროსი 6 თანრიგი არანიშნადია). გარდაქმნის შედეგის გათანაბრებას ემსახურება ADMUX რეგისტრის ADLAR თანრიგი. როცა ეს თანრიგი დაყენებულია “1”-ში, მაშინ გარდაქმნის შედეგი გათანაბრება ხდება 16 თანრიგა სიტყვის უმცროსი თანრიგით (მარჯვნივ), ხოლო თუ “0”-შია - უფროსი თანრიგით (მარცხნივ).

გარდაქმნის შედეგების მიღებისთვის ADCH და ADCL რეგისტრებიდან მიმართვის დროს უნდა შესრულდეს მოქმედებების გარკვეული თანამიმდევრობა: დასაწყისში აუცილებელია მოხდეს ADCL რეგისტრის შემცველობის წაკითხვა, ხოლო შემდეგ - ADCH რეგისტრის წაკითხვა. ეს მოთხოვნები დაკავშირებულია იმასთან, რომ ADCL რეგისტრთან მიმართვის შემდეგ პროცესორი ბლოკირებას ახდენს მონაცემთა რეგისტრების აცგ მხრიდან მიმართვას მანამ, სანამ არ მოხდება ADCH რეგისტრის შემცველობის წაკითხვა. ამის გამო დარწმუნებული უნდა ვიყოთ იმაში, რომ რეგისტრების შემცველობის წაკითხვისას მათში იქნება ჩაწერილი ერთი და იგივე შედეგის შემადგენელი მნიშვნელობები. შესაბამისად, იმ შემთხვევაში თუ კი მორიგი გარდაქმნა დასრულდება ADCH რეგისტრთან მიმართვამდე, გარდაქმნის შედეგი დაიკარგება.

მეორე მხრივ, თუ გარდაქმნის შედეგი გათანაბრდება მარცხნივ და შედეგის მიღების მნიშვნელობა სიზუსტისათვის 8 თანრიგია საკმარისი, მაშინ შეიძლება წაკითხულ იქნეს მხოლოდ ADCH რეგისტრის შემცველობა.

აცგ-ს მახასიათებლების გამოთვლის საილუსტრაციოდ განვიხილოთ მაგალითი. როგორც ზევით იყო ნათქვამი, აცეგეს უნდა მიუერთოთ ერთ-ერთი საყრდენი ძაბვა. დაუშვათ - შიგა საყრდენი ძაბვა 2,56ვ-ის მნიშვნელობით. როდესაც აცგს შესასვლელზე გარდასასახი ძაბვის მნიშვნელობა ნულის ტოლია გარდასახვის რეგისტრში იწერება კოდი 1000000000 (ანუ 1024). აქედან გამომდინარე გარჩევითობა (ანუ მეზობელ კოდებს შორის განსხვავება) გვექნება U_{ref} /1024. შემავალი ძაბვის მაქსიმალურ მნიშვნელობას კი შეესაბამება ერთები უმცროს თანრიგებში (ანუ 1023).

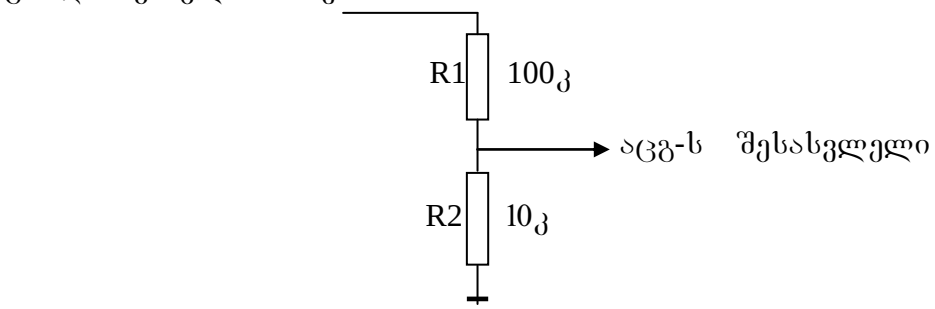
მაქსიმალური ძაბვა აცგს შესასვლელზე შეიძლება გამოვითვალოთ ფორმულით:

$$U_{emax} = 1023 * U_{ref}/1024.$$

განხილული შემთხვევისათვის გვექნება:

$$U_{emax} = 1023 * 2,56/1024 = 2.5575ვ.$$

დაუშვათ, გარდასაქმნელი ძაბვის მნიშვნელობა არის 0 - 25ვ დიაპაზონში. აცგ-ს შესასვლელზე კი დასაშვებია ძაბვის მნიშვნელობა 0 - 5ვ დიაპაზონში. გარდასასახი ძაბვის შემცირების მიზნით უნდა გამოვიყენოთ ძაბვის გამყო- ფი R1=100კ, R2=10კ წინააღმდეგობებით (7.7.სურ.).



სურ. 7.7.

განვსაზღვროთ მაქსიმალური შესასვლელი ძაბვა გამყოფზე R1=100კ, R2=10კ შემთხვევისათვის.

$$U_{emax} = U_{in} * R2/(R1+R2) .$$

სადაც U_{in} გარდასაქმნელი ძაბვის მაქსიმალური მნიშვნელობა, რომელიც შესაძლებელია მიწოდებული იყოს აცგ-ს შესასვლელზე.

ჩვენ მიერ განხილული შემთხვევისათვის გვექნება:

$$U_{in} = 2,5575 * 110_კ/10_კ=28,1325ვ.$$

ასევე საჭიროა ვიცოდეთ რა შედეგი ჩაიწერება აცგ-ს რეგისტრში მის შესასვლელზე ძაბვის სხვადასხვა მნიშვნელობის დროს.

გარდასახვის შედეგი გამოითვლება შემდეგი ფორმულით:

$$ADC= 1024*U_{emax} /U_{ref}.$$

მაგალითად, შესასვლელზე მაქსიმალური ძაბვის შემთხვევაში 2,557ვ, გარდასახვის შედეგი იქნება

$$ADC= 1024*2,557 /2,56=1023,$$

შესასვლელზე 2ვ მიწოდების შემთხვევაში შედეგი იქნება

$$ADC= 1024*2/2,56=800.$$

რეალური მნიშვნელობის მისაღებად ვოლტებში გარდასახვის შედეგი უნდა გამრავლდეს კოეფიციენტზე. განხილული მაგალითისათვის

$$k=2813/1023=2,75.$$

თავი 8

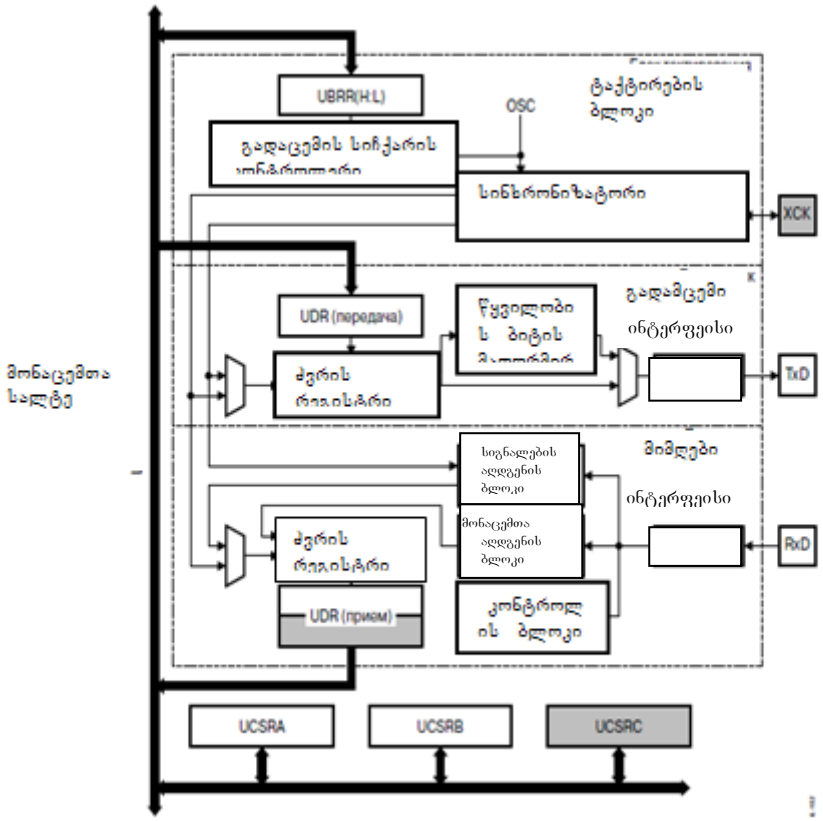
უნივერსალური ასინქრონული (სინქრონულ/ასინქრონული) მიმღებგადამცემი

მიკროკონტროლერ Atmega 128-ს თავის შემადგენლობაში აქვს ორი უნივერსალური მიმღებ/გადამცემი USART0 და USART1.

მიმღებ/გადამცემის ყველა მოდულები უზრუნველყოფენ სრულდუპლექსურ გაცვლას მიმდევრობითი პორტის მეშვეობით, სადაც მონაცემების გაცვლის სიჩქარე შესაძლებელია ვარირებდეს საკმაოდ ფართო საზღვრებში. USART მოდულის გზავნილის ფორმატი შეიძლება შეადგენდეს 5-დან 9 თანრიგამდე. USART მოდულის თავისებურებას წარმოადგენს ის, რომ მას გააჩნია ფორმირების და ლუწობის კონტროლის სქემები.

8.1.USART მოდულის სტრუქტურა

USART მოდულის გამარტივებული სტრუქტურული სქემა ნაჩვენებია 8.1.სურ.-ზე.



სურ.8.1. USART მოდულის სტრუქტურული სქემა

როგორც სურათიდან ჩანს, მოდული შედგება სამი ძირითადი ნაწილისაგან: ტაქტირების ბლოკისაგან, გადაცემის ბლოკისაგან და მიმღები ბლოკისაგან. USART

მოდულის ტაქტირების ბლოკს აქვს სინქრონიზაციის სქემა, რომელიც გამოიყენება სინქრონულ რეჟიმში მუშაობის დროს და გადაცემის სიჩქარის კონტროლერი.

გადაცემის ბლოკი შეიცავს მონაცემთა ბუფერს, ძვრის რეგისტრს, ლუწობის ბიტის ფორმირების და მართვის სქემას.

მიმღებ ბლოკს, თავის მხრივ, გააჩნია სატაქტო სიგნალის და მონაცემების აღდგენის სქემა, ლუწობის კონტროლის სქემა, ძვრის რეგისტრი და მონაცემთა ბუფერი.

მიმღების და გადამცემის ბუფერული რეგისტრები განლაგებულია ერთი და იგივე სამისამართო შეტანა/გამოტანის სივრცეში და აღინიშნებიან, როგორც UDR(Universal data registr ,UDR_n) - მონაცემთა რეგისტრი. ამ რეგისტრში ინახება გადასაცემი და მისაღები მონაცემების უმცროსი 8 თანრიგი. ამოკითხვის დროს ხდება მიმართვა მიმღების UDR ბუფერულ რეგისტრთან, ჩაწერის დროს – გადამცემის ბუფერულ რეგისტრთან.

თითოეულ USART ბლოკს აქვს სამი გამომყვანი RxD, TxD და XCK, რომლებიც დაკავშირებული არიან საერთო დანიშნულების პორტების გამოსასვლელებთან და ანხორციელებენ მათ ალტერნატიულ ფუნქციებს (USART ბლოკის მუშაობის დროს იგი იყენებს ამ გამოსასვლელებს, წინააღმდეგ შემთხვევაში - გამოსასვლელებს იყენებს პორტი). RxD გამომყვანით სრულდება მონაცემთა მიმდევრობით მიღება, TxC გამომყვანით - გაცემა. XCK გამოსასვლელზე გაიცემიან მასინქრონებელი სიგნალები. 8.1.ცხრილში მოცემულია USART მოდულის მიერ გამოყენებული პორტის გამოსასვლელები და მათი დანიშნულება.

ცხრილი 8.1. გამომყვანები, რომელსაც იყენებენ USART მოდულები

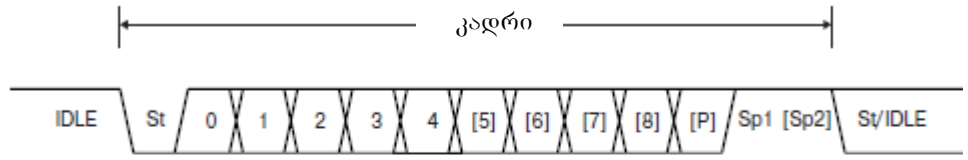
დანიშნულება	გამომყვანები	აღწერა
RXD0	PE0	USART0 შესასვლელი
TXD0	PE1	USART0 გამოსასვლელი
XCK0	PE2	USART0 გარე სატაქტო სიგნალის შესასვლელი/გამოსასვლელი
RXD1	PD2	USART1 შესასვლელი
TXD1	PD3	USART1 გამოსასვლელი
XCK1	PD5	USART1 გარე სატაქტო სიგნალის შესასვლელი/გამოსასვლელი

8.2. კადრის ფორმატი

ინფორმაციის გადაცემა ორივე მიმართულებით ხორციელდება კადრის საშუალებით. მოცემულ შემთხვევაში კადრის ქვეშ იგულისხმება მონაცემის ერთი სიტყვის და თანმხლები ინფორმაციის ერთობლიობა (8.2.სურ.).

კადრი იწყება დაწყების (Start) ბიტით, რომელსაც მიჰყვება მონაცემთა სიტყვის უმცროსი თანრიგი. მონაცემთა სიტყვის უფროსი თანრიგის შემდეგ - ერთი ან ორი დამთავრების (Stop) ბიტი. იმ შემთხვევაში, თუ გათვალისწინებულია გადაცემის ლუწობაზე კონტროლი, მონაცემთა ბიტებს დაემატება საკონტროლო ბიტი , ის მოთავსდება მონაცემის სიტყვის უფროს თანრიგსა და პირველ Stop ბიტს შორის.

კოდის ფორმატი განისაზღვრება $UCSR_nB$ და $UCSR_nC$ რეგისტრებში, რაც განხილული იქნება ქვევით.



St (0...8) P - "start" ბიტი, ყოველთვის "0"
Sp1 [Sp2] - მონაცემთა თანრიგები
IDLE - კონტროლის ბიტი
- "stop" ბიტი, ყოველთვის "1"

სურ.8.2. USART ინტერფეისის კადრის ფორმატი

8.3.USART-ის მართვის რეგისტრები

USART მოდულის სამართავად განკუთვნილია სამი რეგისტრი: $UCSR_nA$, $UCSR_nB$ და $UCSR_nC$ (n აღნიშნავს მოდულის ნომერს).

$UCSR_nA$, $UCSR_nB$ და $UCSR_nC$ რეგისტრების ფორმატები მოცემულია 8.3...8.5.სურ.-ზე, ხოლო ამ რეგისტრების თანრიგების მნიშვნელობების აღწერა - 8.2...8.4.ცხრილებში შესაბამისად.

7	6	5	4	3	2	1	0
RXC_n	TXC_n	$UDRE_n$	FE_n	DOR_n	UPE_n	$U2X_n$	$MPCM_n$
წაკითხვა(R)/ჩაწერა(w)	R	R/W	R	R	R	R	R/W	R/W
საწყისი მნიშვნელობა	0	0	1	0	0	0	0	0

სურ.8.3. $UCSR_nA$ რეგისტრის ფორმატი

7	6	5	4	3	2	1	0
$RXCIE_n$	$TXCIE_n$	$UDRIE_n$	$RXEN_n$	$TXEN_n$	UCSZn2	$RXB8_n$	$TXB8_n$
წაკითხვა(R)/ჩაწერა(w)	R	R/W	R	R	R	R	R/W	R/W
საწყისი მნიშვნელობა	0	0	1	0	0	0	0	0

სურ.8.4. $UCSR_nB$ რეგისტრის ფორმატი

7	6	5	4	3	2	1	0
-	UMSELn	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn
წაკითხვა(R)/ჩაწერა(w)	R	R/W	R	R	R	R	R/W	R/W
საწყისი მნიშვნელობა	0	0	1	0	0	0	0	0

სურ.8.5. $UCSR_nC$ რეგისტრიეს ფორმატი

ცხრილი 8.2. UCSR0A და UCSR1A რეგისტრების თანრიგები

თანრიგი	დანიშნულება	აღწერა
7	RXC_n	**მიღების დასრულების ალამი.** ალამი დაყენდება ერთიანში იმ შემთხვევაში, თუ არ არის ამოკითხული მონაცემები მიმღების ბუფერულ რეგისტრიდან (UDR მონაცემების რეგისტრიდან). ალმის აპარატურული განულება ხდება ბუფერის დაცლის შემდეგ. იმ შემთხვევაში თუ დაყენებულია $UCSR_nB$ რეგისტრის $RXCIE_n$ თანრიგი, მაშინ გენერირდება მოთხოვნა წყვეტაზე <<მიღება დასრულებულია>>.
6	TXC_n	**გადაცემის დასრულების ალამი.** ალამი დაყენდება ერთიანში თუ გადაცემის ძვრის რეგისტრიდან ყველა თანრიგების გადაგზავნა დასრულდება, იმ პირობით, რომ UDR მონაცემთა რეგისტრში არ იქნება ჩატვირთული ახალი მნიშვნელობა.თუ $UCSR_nB$ რეგისტრის $TXCIE_n$ თანრიგი დაყენებულია, მაშინ, გენერირდება წყვეტა <<გადაცემა დასრულებულია>>. ალამი განულდება აპარატურულად წყვეტის ქვეპროგრამის შესრულებისას ან პროგრამულად.
5	$UDRE_n$	**მონაცემების რეგისტრის დაცლის ალამი.**მოცემული ალამი დაყენდება ერთიანში იმ შემთხვევაში თუ გადაცემის ბუფერი ცარიელია (UDR მონაცემთა რეგისტრიდან ბაიტის გადაცემის შემდეგ გადადამცემის ძვრის რეგისტრში).დაყენებული ალამი ნიშნავს, რომ მონაცემთა რეგისტრში შეიძლება ჩაიტვირთოს ახალი მნიშვნელობა. თუ UCSRB რეგისტრის $UDRE_n$ თანრიგი დაყენებულია, მაშინ გენერირდება წყვეტა <<მონაცემთა რეგისტრი ცარიელია>>. ალამი განულდება აპარატურუ- ლად იმ შემთხვევაში როდესაც ჩაწერა ხდება მონაცემთა რეგისტრში.
4	FE_n	**კადრირების შეცდომის ალამი.** FE_n ალამი დაყენდება ერთიანში იმ შემთხვევაში, როდესაც ადგილი აქვს კადრირების შეცდომას. ე.ი. თუ მიღებული გზავნილის პირველი “stop”- ბიტი ტოლია ნულის.
3	DOR_n	**გადავსების ალამი**. DOR_n დაყენება მოხდება ერთიანში თუ ახალი “start”-ბიტის აღმოჩენის მომენტში მიმღების ძვრის რეგისტრში არის უკანასკნელი მიღებული სიტყვა, ხოლო მიმღების მონაცემთა რეგისრი სავსეა. ალამი განულდება, თუ ადგილი ექნება მიღებული მონაცემის გადაგზავნას ძვრის რეგისტრიდან მიმღების რეგისტრში.
2	UPE_n	**ლუწობის კონტროლის შეცდომის ალამი.** UPE_n დაყენდება ერთიანში, როდესაც მიმღების ბუფერულ რეგისტრში ლუწობაზე კონტროლის შედეგად მონაცემში აღმოჩდება შეცდომა.იმ შემთხვევაში თუ კონტროლი არ გამოიყენება, მაშინ ეს თანრიგი მუდმივად განულე- ბულია.
1	$U2X_n$	**გადაცემის სიჩქარის გაორმაგება.** თუ $U2X_n$ ალამი დაყენებულია ერთიანში, მაშინ წინაგამყოფის კონტროლერის გადაცემის გაყოფის სიჩქარის კოეფიციენტი მცირდება 16-დან 8-მდე. რის შედეგადაც ორმაგდება სიჩქარე ასინქრონული გადაცემის მიმდევრობით არხში. USART რეგისტრის $U2X_n$ თანრიგი გამოიყენება მხოლოდ ასინქრონული მუშაობის რეჟიმში. სინქრონულ რეჟიმში ის განულებული უნდა იყოს.
0	$MPCM_n$	**მულტიპროცესორული გაცვლის რეჟიმი.** $MPCM_n$ გამოიყენება მულტიპროცესორულ გაცვლის რეჟიმში. თუ ეს თანრიგი დაყენებულია ერთიანში, მაშინ მიმყოლი პროცესორი ელოდება კადრის მიღებას,რომელშიც ჩაწერილია მისამართი. კადრი, რომელშიც არ არის ჩაწერილი მისამართი იგნორირდება.

შენიშვნა: n=0 ან n=1

ცხრილი 8.3. UCSR0B და UCSR1B რეგისტრების თანრიგები

თანრიგი	დანიშნულება	აღწერა
7	$RXCIE_n$	**წყვეტის ნებადართვა მიღების დასრულების გამო.** თუ აღნიშნული თანრიგი დაყენებულია ერთიანში, მაშინ $UCSR_nA$ რეგისტრის RXCn ალმის დაყენებისას გენერირდება წყვეტა <<მიღება დასრულებულია>> (თუ ამავე დროს SREG რეგისტრის I თანრიგი დაყენებულია ერთიანში)
6	$TXCIE_n$	**წყვეტის ნებადართვა გადაცემის დასრულების გამო.** თუ მოცემული თანრიგი დაყენებულია ერთიანში, მაშინ $UCSR_nA$ რეგისტრის TXC_n ალმის დაყენებისას გენერირდება წყვეტა << გადაცემა დასრულებულია>> (თუ SREG რეგისტრის I თანრიგი დაყენებულია ერთიანში)
5	$UDRIE_n$	**წყვეტის ნებადართვა მონაცემების რეგისტრის გასუფთავების შემთხვევაში** . თუ მოცემული თანრიგი დაყენებულია ერთიანში, მაშინ $UCSR_nA$ რეგისტრის UDRE ალმის დაყენებისას გენერირდება წყვეტა <<მონაცემთა თანრიგი ცარიელია>> (თუ SREG რეგისტრის I თანრიგი დაყენებულია ერთიანში)
4	$RXEN_n$	**მიღების ნებადართვა** .თუ მოცემული თანრიგი დაყენებულია ერთიანში, მაშინ ნებადართულია USART მიმღების მუშაობა და წინასწარ განისაზღვრება RXD_n გამოსასვლელის ფუნქციონირება. $RXEN_n$ ჩამოგდების დროს მიმღების მუშაობა იკრძალება, ხდება მისი ბუფერის განულება. TXC_n, DOR_N და FE_n ალმების მნიშვნელო- ბები ხდება არაარსებითი.
3	$TXEN_n$	**გადაცემის ნებადართვა.** თუ მოცემული თანრიგი დაყენებულია ერთიანში, მაშინ ნებადართულია USART,როგორც გადამცემის მუშაობა და წინასწარ განისაზღვრება TXD_n გამოსასვლელის ფუნქციონირება.თუ გადაცემის პროცესში თანრიგი განულდება, მაშინ გადამცემის გამორთვა განხორციელდება მხოლოდ ძვრის რეგისტრში და გადამცემის ბუფერში მოთავსებული. მონაცემის გადაცემის დასრულების შემდეგ
2	$UCSZ_n2$	**გზავნილის ფორმატი.** ეს თანრიგი გამოიყენება იმისათვის, რომ განისაზღვროს მონაცემთა სიტყვის თანრიგების რაოდენობა, რომელიც გადაიცემა მიმდევრობითი არხით. USART მოდულებში ის გამოიყენება $UCSR_n C$ რეგისტრის $UCSZ_n1:0$ თანრიგებთან ერთობლიობაში.
1	$RXB8_n$	**მისაღები მონაცემის 8-e თანრიგი** . 9 თანრიგიანი მონაცემთა სიტყვის gamoyenebis დროს ამ თანრიგis SigTavsi warmoadgens miRებული სიტყვის უფროს თანრიგs. USART შემთხვევაში ამ თანრიგიდან ამოკითხვა უნდა განხორციელდეს ufri ადრე ვიდრე მოხდება მონაცემის ამოკითხვა UDR monacemTa registridan.
0	$TXB8_n$	**გადასაცემი მონაცემის 8-ე თანრიგი** . 9 თანრიგიანი მონაცემთა სიტყვის გამოყენების დროს ამ თანრიგის შიგთავსი წარმოად- გენს გადასაცემი სიტყვის უფროს თანრიგს. საჭირო მნიშვნელობა ამ თანრიგში უნდა შეტანილი იქნეს წინასწარ მონაცემითა ბაიტის ჩაიტვირთვამდე UDR რეგისტრში.

შენიშვნა: n=0 ან n=1

ცხრილი 8.4. UCSR0C და UCSR1C რეგისტრების თანრიგები

<table>
<tr><th>თანრიგი</th><th>დასახელება</th><th colspan="3">აღწერა</th></tr>
<tr><td>7</td><td>–</td><td colspan="3">დარეზერვირებულია, ამოიკითხება როგორც ნული</td></tr>
<tr><td>6</td><td>$UMSEL_n$</td><td colspan="3">USART მუშაობის რეჟიმი. თუ თანრიგში ჩაწერილია ნული, მაშინ USART მუშაობს ასინქროლურ რეჟიმში. თუ თანრიგი დაყენებულია ერთიანში, მაშინ USART მუშაობს სინქროლურ რეჟიმში.</td></tr>
<tr><td>5</td><td>UPM_n1</td><td colspan="3" rowspan="2">კონტროლისა და ლუწობის ფორმირების სქემის მუშაობის რეჟიმი. ეს თანრიგები განსაზღვრავენ კონტროლის სქემის ფუნქციონირებას და ლუწობის ფორმირებას.</td></tr>
<tr><td>4</td><td>UPM_n0</td></tr>
<tr><td>3</td><td>$USBS_n$</td><td colspan="3">სტოპ ბიტების რაოდენობა. ეს თანრიგი განსაზღვრავს სტოპ ბიტების რაოდენობას, რომელსაც გზავნის გადამცემი. თუ კი ეს თანრიგი დაყენებულია ნულში, გადამცემი გზავნის 1 სტოპ ბიტს, თუ დაყენებულია ერთიანში, მაშინ გადაიგზავნება 2 სტოპ ბიტი. მიმღებისათვის ამ თანრიგის შემცველობა არ არის არსებითი.</td></tr>
<tr><td>2</td><td>$UCSZ_n1$</td><td colspan="3" rowspan="2">გზავნილის ფორმატი. $UCSZ_n2$ თანრიგთან ერთად ეს თანრიგები განსაზღვრავს გზავნილში მონაცემის თანრიგების რაოდენობას (სიტყვის ზომას)</td></tr>
<tr><td>1</td><td>$UCSZ_n0$</td></tr>
<tr><td rowspan="4">0</td><td rowspan="4">$UCPOL_n$</td><td colspan="3">ტაქტური სიგნალის პოლარობა. ამ თანრიგის მნიშვნელობა განსაზღვრავს მომენტს მონაცემების გაცემის და ამოკითვის მოდულის გამომყვანებზე. თანრიგები გამოიყენება მხოლოდ სინქრონულ რეჟიმში. ასინქრონულ რეჟიმში ის უნდა იყოს განულებული.</td></tr>
<tr><td>$UCPOL_n$</td><td>TXD_n გამომყვანზე მონაცემების გამოტანა</td><td>RXD_n გამომყვანიდან მონაცემების ამოკითხვა</td></tr>
<tr><td>0</td><td>XCK_n კლებადი ფრონ ტი</td><td>XCK_n მზარდი ფრონ ტი</td></tr>
<tr><td>1</td><td>XCK_n მზარდი ფრონტი</td><td>XCK_n კლებადი ფრონ ტი</td></tr>
</table>

შენიშვნა: n=0 ან n=1

მონაცემთა სიტყვის ზომა განისაზღვრება $UCSR_nB$ და $UCSR_nC$ რეგისტრებში: - $UCSZ_n2$..... $UCSZ_n0$ თანრიგებით (8.5.ცხრილი).

ცხრილი 8.5. USART მოდულებში მონაცემთა სიტყვის ზომის განსაზღვრა

$UCSZ_n2$	$UCSZ_n1$	$UCSZ_n0$	მონაცემთა სიტყვის ზომა
0	0	0	5 თანრიგი
0	0	1	6 თანრიგი
0	1	0	7 თანრიგი
0	1	1	8 თანრიგი
1	0	0	დარეზერვირებულია
1	0	0	დარეზერვირებულია
1	1	0	დარეზერვირებულია
1	1	1	9 თანრიგი

Stop ბიტების რაოდენობა USART მოდულებში განისაზღვრება $UCSR_nC$ რეგისტრის $USBS_n$ თანრიგის საშუალებით. თუ ამ თანრიგში ნულია, მაშინ გადაცემის ბლოკი

აფორმირებს გზავნილის ბოლოს ერთ გაჩერების ბიტს. წინააღმდეგ შემთხვევაში, თუ თანრიგი დაყენებულია ერთიანში, მაშინ გადამცემის ბლოკი აფორმირებს 2 Stop ბიტს.

$UCSR_nC$ რეგისტრის UPM_n1 და UPM_n0 თანრიგები განსაზღვრავენ USART მოდულების ლუწობის კონტროლის სქემის ფუნქციონირებას 8.6.ცხრილს შესაბამისად.

ცხრილი 8.6. ლუწობის კონტროლის მართვა

UPM_n1	UPM_n1	მუშაობის რეჟიმი
0	0	კონტროლი გამორთულია
0	1	დარეზერვირებულია
1	0	კონტროლი ჩართულია, შემოწმება ლუწობა-ზე (even parity)
1	1	კონტროლი ჩართულია, შემოწმება კენტობა-ზე (odd parity)

მიღება/გაცემის სიჩქარის მართვა

ასინქრონულ და ასევე სინქრონულ რეჟიმში წამყვანის რანგში მუშაობის დროს მონაცემების მიღებისა და გადაცემისას სიჩქარეს განსაზღვრავს გადაცემის სიჩქარის კონტროლერი, რომელიც ფუნქციონირებს როგორც სისტემური სატაქტო სიგნალების გამყოფი პროგრამირებადი გაყოფის კოეფიციენტით. კოეფიციენტი განისაზღვრება კონტროლერის UBRR რეგისტრის შემცველობის შესაბამისად. მიმღების ბლოკს გადაეცემა ფორმირებული სიგნალი გაყოფის გარეშე, ხოლო გადამცემის ბლოკს – დამატებითი გამყოფის გავლით, რომლის გაყოფის კოეფიციენტი (2, 8 ან 16) დამოკიდებულია USART მოდულის მუშაობის რეჟიმზე.

UBRR რეგისტრი განთავსებულია შეტანა/გამოტანის ორ UBRRnH:UBRRnL რეგისტრში. ასინქრონულ რეჟიმში მუშაობის დროს გაცვლის სიჩქარე განისაზღვრება UBRR რეგისტრის შემცველობით და $UCSR_nA$ რეგისტრის $U2X_n$ თანრიგის მდგომარეობით. თუ ამ თანრიგში ჩაწერილია ერთი, სიხშირის გამყოფის გაყოფის კოეფიციენტი მცირდება ორჯერ და შესაბამისად მცირდება გაცვლის სიჩქარე. სინქრონულ რეჟიმში მუშაობის დროს ეს თანრიგი განულებულია.

8.4.მონაცემთა გადაცემა

USART მოდულის გადამცემი ბლოკის მუშაობა ნებადართულია, თუ $UCSR_nB$ რეგისტრის $TXEN_n$ თანრიგი დაყენებულია ერთიანის მდგომარეობაში. თანრიგის დაყენების შემდეგ TXD_n გამომყვანი უერთდება USART გადამცემს და იწყებს ფუნქციონირებას როგორც დამოუკიდებელი გამომყვანი, იმის მიუხედავად, თუ რა მდგომარეობაშია დაყენებული პორტების მართვის რეგისტრები. RXEN ბიტის ერთიანში დაყენებით მოდული გადადის მიმღების რეჟიმში და RXD გამომყვანები უერთდება USART-ის მიმღებს.

მუშაობის სინქრონული რეჟიმში, USART-თვის ასევე განისაზღვრება XCK_n გამომყვანის ფუნქციონირება.

გადაცემის დაწყება ხდება გადასაცემი მონაცემის ჩაწერით გადამცემის მონაცემთა ბუფერულ UDR რეგისტრში. რის შემდეგაც მონაცემები გადაიგზავნება *UDR* რეგისტრიდან გადამცემის ძვრის რეგისტრში. თუ გამოიყენება 9-თანრიგიანი

მონაცემი, მაშინ $UCSR_nB$ რეგისტრის $TXB8_nB$ თანრიგის კოპირება ხდება ძვრის რეგისტრის 9 –ე თანრიგში.

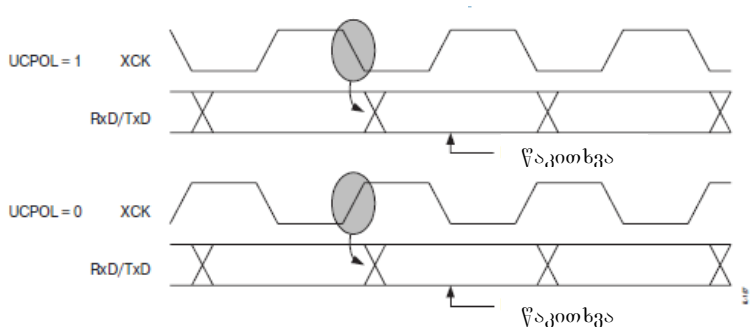
ასეთ შემთხვევაში შესაძლებელია ადგილი ჰქონდეს ორ ვარიანტს:

- UDR რეგისტრში ჩაწერა ხორციელდება იმ მომენტში, როდესაც გადამცემი იმყოფება მოლოდინის მდგომარეობაში (წინა მონაცემები გადაცემულია). ამ შემთხვევაში მონაცემები გადაიცემა ძვრის რეგისტრში მაშინვე UDR რეგისტრში ჩაწერის შემდეგ.
- UDR რეგისტრში ჩაწერა ხორციელდება გადაცემის დროს. ამ შემთხვევაში მონაცემები გადაიგზავნება ძვრის რეგისტრში მას შემდეგ, როდესაც გაიგზავნა მიმდინარე კადრის უკანასკნელი Stop ბიტი .

9-თანრიგა მონაცემთა გადაცემის შემთხვევაში მე-9 თანრიგი უნდა ჩაიტვირთოს TXB8n თანრიგში მანამ, მოხდება სიტყვის უმცროსი ბაიტის ჩაწერა მონაცემთა რეგისტრში.

მას შემდეგ, როდესაც მონაცემთა სიტყვა გადაიგზავნება ძვრის რეგისტრში, $UCSR_n$ რეგისტრის $UDRE_n$ ალამი დაყენდება ერთიანის მდგომარეობაში. რაც იმის მაუწყებელია, რომ გადამცემი მზადაა ახალი მონაცემის სიტყვის მისაღებად. ამ მდგომარეობაში ალამი რჩება მანამ, სანამ არ მოხდება ბუფერში მომდევნო ჩაწერა. ერთდროულად რეგისტრში გადაგზავნის დროს ფორმირდება Start - ბიტი, შესაძლებელია ლუწობის ბიტი და ერთი ან ორი Stop- ბიტი.

ძვრის რეგისტრის ჩატვირთვის შემდეგ მისი შემცველობა იწყებს ძვრას მარჯვნივ და მიეწოდება TXD_n გამომყვანს ბიტების იმ თანამიმდევრობით, რომელიც ზემოთ იყო განხილული. ძვრის სწრაფქმედება განისაზღვრება სატაქტო სიგნალების კონტროლერის გაწყობით. სინქროლურ რეჟიმში მუშაობის დროს TXD_n გამომყვანის მდგომარეობა იცვლება XCK_n სიგნალის ერთ-ერთი ფრონტის მიხედვით. თუ $UCSR_nC$ რეგისტრის $UCPOL_n$ თანრიგი დაყენებულია ნულში, მაშინ TXD_n გამომყვანის მდგომარეობის შეცვლა ხდება XCK_n სიგნალის მზარდ (წინა) ფრონტზე, თუ დაყენებულია ერთიანში, მაშინ სიგნალის კლებად (უკანა) ფრონტზე, როგორც ნაჩვენებია 8.6.სურ.-ზე.



სურ.8.6. USART-ის სინქრონულ რეჟიმში მუშაობის დროითი დიაგრამა

თუ გადაცემის დროს UDR რეგისტრში იყო ჩაწერილი ახალი მონაცემები, მაშინ უკანასკნელი Stop ბიტის გაცემის შემდეგ ის გადაიგზავნება ძვრის რეგისტრში. ხოლო იმ შემთხვევაში, თუ კოდის გადაცემის დამთავრების მომენტში ასეთი ჩაწერა შესრულებული არ იქნა, მაშინ დაყენდება $UCSR_nA$ რეგისტრის ალამი TXC_n ."გადაცემა

დასრულებულია". $UCSR_nA$ წყვეტის რეგისტრის ალმის განულება ხორციელდება აპარატურულად შესაბამის წყვეტის ქვეპროგრამის შესრულების დაწყების დროს ან პროგრამულად.

გადამცემის გამორთვა სრულდება $UCSR_nB$ რეგისტრის $TXEN_n$ თანრიგის განულებით. თუ ამ ბრძანების შესრულების მომენტში ხორციელდებოდა გადაგზავნა, თანრიგის განულება ხდება მხოლოდ მიმდინარე გადაგზავნის დამთავრების შემდეგ, ე.ი მას შემდეგ როდესაც მოხდება გადამცემის ძვრის და ბუფერული რეგისტრების გასუფთავება. იმ შემთხვევაში, როდესაც გადამცემი გამორთულია, TXD_n გამომყვანი შეიძლება გამოვიყენოთ, როგორც საერთო დანიშნულების შეტანა/გამოტანის კონტაქტი.

8.5.მონაცემის მიღება

მიმღების მუშაობა ნებადართულია $UCSR_nB$ რეგისტრის $RXEN_n$ თანრიგში ერთიანის ჩაწერით. თუ თანრიგი დაყენებულია 1-ში ხდება შესაბამისი პორტის RXDn გამომყვანის მიერთება USART მიმღებთან და იწყებს ფუნქციონირებას როგორც შესავალი, მიუხედავად იმის, თუ რა მდგომარეობაშია პორტის მართვის რეგისტრები. თუ გამოიყენება მუშაობის სინქრონული რეჟიმი, მაშინ ხორციელდება ასევე XCK_n გამომყვანის ფუნქციონირების წინასწარი გადანაწილება.

მიმღების მიერ Start ბიტის აღმოჩენის შემდეგ იწყება კადრის მონაცემთა ყოველი თანრიგის მიმდევრობით მიღება იმ სიხშირით, რომელსაც აწესებს გადაცემის სიჩქარის კონტროლერი ან ხორციელდება XCK სატაქტო სიგნალით. წაკითხული მონაცემთა თანრიგები თანამიმდევრულად განთავსდება მიმღების ძვრის რეგისრში მანამ არ მოხდება პირველ Stop ბიტის აღმოჩენა კადრში. რის შემდეგაც ძვრის რეგისტრის შემცველობა გადაიგზავნება მიმღების ბუფერში, საიდანაც მიღებული მნიშვნელობა შესაძლებელია წავიკითხოთ მოდულის რეგისტრიდან. 9 თანრიგიანი მონაცემთა სიტყვის გამოყენების შემთხვევაში უფროსი თანრიგის მნიშვნელობა განისაზღვრება $UCSR_nB$ რეგისტრის $RXB8_n$ ალამის მდგომარეობის მიხედვით. ამასთან, USART მოდულებში უფროსი თანრიგის მნიშვნელობა ამოკითხული უნდა იყოს მონაცემთა რეგისტრთან მიმართვამდე.

თუ კადრის მიღების დროს ჩართული არის ლუწობის კონტროლის სქემა, ის გამოითვლის ლუწობის ბიტს მონაცემთა სიტყვის ყველა თანრიგისთვის და ადარებს მას მიღებულ ლუწობის ბიტს. შედეგის დამახსოვრება ხდება მიმღების ბუფერში მიღებულ მონაცემთა სიტყვასთან და Stop ბიტთან ერთად. შეცდომის არსებობა ან არარსებობას ლუწობის კონტროლის დროს განსაზღვრავს UPE_n ალამი. ამ ალამის ერთიანში დაყენება მოხდება იმ შემთხვევაში, თუ მომდევნო სიტყვას, რომელიც შეიძლება წაკითხული იყოს ბუფერიდან აქვს ლუწობის კონტროლის შეცდომა. კონტროლის გამორთვის შედეგად, UPE_n შემცველობა წაიკითხება როგორც ნული.

USART მოდულის მიმღებს აქვს კიდევ ორი ალამი, რომელიც გაცვლის მდგომარეობას უჩვენებს: FE_n კადრირების შეცდომის ალამი და DOR_n გადავსების ალამი. FE_n კადრირების შეცდომის ალამი დაყენდება ერთიანში, როდესაც კადრის პირველი Stop ბიტის მდგომარეობა ტოლია ნულის. DOR_n ალამი USART-ში გვიჩვენებს,

რომ დაკარგულია მონაცემები მიმღების ბუფერის გადავსების გამო. DOR_n ალმის დაყენება ხდება ერთიანში, როდესაც სრულდება ახალი კადრის Start ბიტის მიღება იმ შემთხვევაში, როცა შევსებულია ბუფერი და მიმღების ძვრის რეგისტრი. დაყენებული DORn ალამი ნიშნავს, რომ UDR რეგისტრიდან ამოკითხულ ძველ ბაიტებსა და მოცემულ მომენტში ამოკითხულ ბაიტებს შორის მოხდა ერთი ან რამდენიმე კადრის დაკარგვა.

USART მოდულის მიმღების მდგომარეობის ინდიკაციისთვის გამოიყენება UCSRnA რეგისტრის ჩყვეტის ალამი-"გადაცემა დასრულებულია" RXCn. ეს ალამი დგება ერთიანში იმ შემთხვევაში, როცა მიმღების ბუფერში გვაქვს წაუკითხავი მონაცემები. USART მოდულებში ბუფერების დაცარიელების შემდეგ (როდესაც მოხდება მასში მყოფი ყველა მონაცემის ამოკითხვა) იგი განულდება.

მიმღების გამორთვა ხორციელდება UCSRnB რეგისტრის RXENn თანრიგის განულებით. გადამცემისგან განსხვავებით, მიმღები გამოირთვება თანრიგის განულებისთანავე, ე.ი, ამ მომენტრში მიღებული კადრი იკარგება. გარდა ამისა, USART მოდულებში, მიმღების გამორთვის შემთხვევაში ხდება მისი ბუფერის განულება, ე.ი იკარგება ყველა წაუკითხავი მონაცემი.

გამორთულ მიმღებში RXDn გამომყვანი შეიძლება გამოვიყენოთ, როგორც შეტანა/გამოტანის საერთო დანიშნულების კონტაქტი.

პროგრამებთან ურთიერთქმედებისთვის მოდულში გათვალისწინებულია 3 ჩყვეტა, რომელთა გენერაცია ხორციელდება შემდეგი მოვლენების შექმნის დროს: "გადაცემა დასრულებულია", "გადაცემის მონაცემის რეგისტრი ცარიელია" და "მიღება დასრულებულია". აღნიშნული ჩვეტების გენერირების დროს მიკროკონტროლერი გადაირთვება შესაბამის სამომსახურო პროგრამაზე, რომლის საშუალებით ხდება USART ბლოკთან მიმართვა მონაცემთა გაცვლის მიზნით.

8.6.მულტიპროცესორული მუშაობის რეჟიმი

თუ წამყვან მიკროკონტროლერს აქვს სურვილი რაიმე გადასცეს ის გზავნის მისამართის ბაიტს, რომელიც განსაზღვრავს რომელ მიკროკონტროლერთან სურს მიმართვა. იმ შემთხვევაში, თუ რომელიმე მიმყოლმა მიკროკონტროლერმა ამოიცნო თავისი მისამართი, მაშინ ის გადადის მონაცემთა მიღების რეჟიმში და ღებულობს მომდევნო ბაიტს, როგორც მონაცემს. ყველა სხვა მიმყოლი მიკროკონტროლერი ახდენს მიღებული ბაიტის იგნორირებას, მანამ არ მოხდება წამყვანი მიკროკონტროლერის მიერ ახალი მისამათის ბაიტის გადაგზავნა. მიღებული კადრების ამორჩევის ფილტრაციის რეჟიმის ჩართვა, რომელშიც არ არის მისამართი ხორციელდება UCSRnA რეგისტრის MPCMn თანრიგის ერთიანში დაყენებით.

წამყვანი მიკროკონტოლერი, გადაცემას ასრულებს 9 თანრიგიანი მონაცემებით. მისამართის ბაიტის გადაცემის დროს უფროს თანრიგში უნდა იყოს დაყენებული ერთიანი, ხოლო მონაცემის ბაიტის გადაცემის შემთხვევაში – ნული. მიმყოლ მიკროკონტროლერში მიღების მექანიზმი დამოკიდებულია მიმღების მუშაობის რეჟიმზე. თუ მიმღები გაწყობილია 5....8 თანრიგიან მონაცემების მიღებაზე, მაშინ კადრის შემცველობის იდენტიფიკაცია განისაზღვრება პირველი Stop ბიტის მეშვეობით. 9 თანრიგიანი მონაცემის მიღების შემთხვევაში, კადრის შემცველობის იდენტიფიკაცია ხდება მონაცემის სიტყვის უფროსი თანრიგით. თუ კადრის მე-9 თანრიგში ერთიანია, მაშინ კადრის შემცველობაა მისამართი, წინააღმდეგ შემთხვევაში – მონაცემი.

იმისათვის რომ, განხორციელდეს მონაცემთა გაცვლა მულტიპროცესორულ რეჟიმში აუცილებელია შესრულდეს შემდეგი მოქმედებები:

1. ყველა მიმყოლი მიკროკონტროლერი უნდა გადაერთოს მულტიპროცესორულ გაცვლის რეჟიმში UCSRnA რეგისტრის MPCMn თანრიგის ერთიანში დაყენებით;
2. წამყვანი მიკროკონტროლერი გზავნის მისამართის კადრს, ხოლო ყველა მიმყოლი მიკროკონტროლერი მას ღებულობს. შესაბამისად ყველა მიმყოლ მიკროკონტროლერში დაყენდება UCSRnA რეგისტრის RXCn ალამი;
3. ყოველი მიმყოლი მიკროკონტროლერი კითხულობს მონაცემთა რეგისტრის შემცველობას. მიკროკონტროლერი, რომლის მისამართის მნიშვნელობა დაემთხვევა წამყვანის მიერ გადმოცემულ მისამართს, ანულებს MPCMn თანრიგს.
4. დამისამართებული მიკროკონტროლერი იწყებს კადრის მიღებას რომლის შემცველობაც მონაცემია.
5. მას შემდეგ, როდესაც მოხდება მონაცემის ბოლო ბაიტის მიღება, დამისამართებული მიკროკონტროლერი აყენებს ერთიანში MPCMn თანრიგს და კვლავ ელოდება მისამართის კადრის მოსვლას. პროცესი მეორდება 2 პუნქტიდან.

თავი 9

მიმდევრობითი პერიფერიული ინტერფეისი (SPI)

მიმდევრობით პერიფერიულ ინტერფეისს SPI (Serial Peripheral interface) აქვს ორმხრივი დანიშნულება. პირველ რიგში მას შეუძლია განახორციელოს მონაცემების გაცვლა მიკროკონტროლერსა და სხვადასხვა პერიფერიულ მოწყობილობებს შორის, როგორებიცაა ციფრული პოტენციომეტრები, ცაგ/აცგ, FLASH - მეხსიერება და სხვა. ამ ინტერფეისის საშუალებით ასევე შესაძლებელია განხორციელდეს მონაცემების გაცვლა რამოდენიმე AVR მიკროკონტროლერს შორის. მოცემულ თავში განიხილება SPI ინტერფეისის გამოყენება მაღალ სიჩქარიანი კავშირის არხის სახით.

გარდა ამისა SPI ინტერფეისის საშუალებით შესაძლებელია განხორციელდეს მიკროკონტროლერის პროგმარირება (ე.წ. მიმდევრობითი პროგრამირების რეჟიმში).

SPI ინტერფეისით მონაცემების გაცვლის დროს მიკროკონტროლერს შეუძლია მუშაობა როგორც წამყვანს («Master») ან როგორც მიმყოლს («Slave»). მომხმარებელს ასევე შეუძლია აირჩიოს გადაცემის სიჩქარე (შვიდი პროგრამირებული მნიშვნელობა) და გადაცემის ფორმატი (უმცროსი თანრიგიდან უფროსისაკენ და პირიქით).

SPI ქვესისტემის დამატებითი შესაძლებლობა არის “ გაღვიძება” Idle რეჟიმიდან მონაცემების მიწოდების დროს.

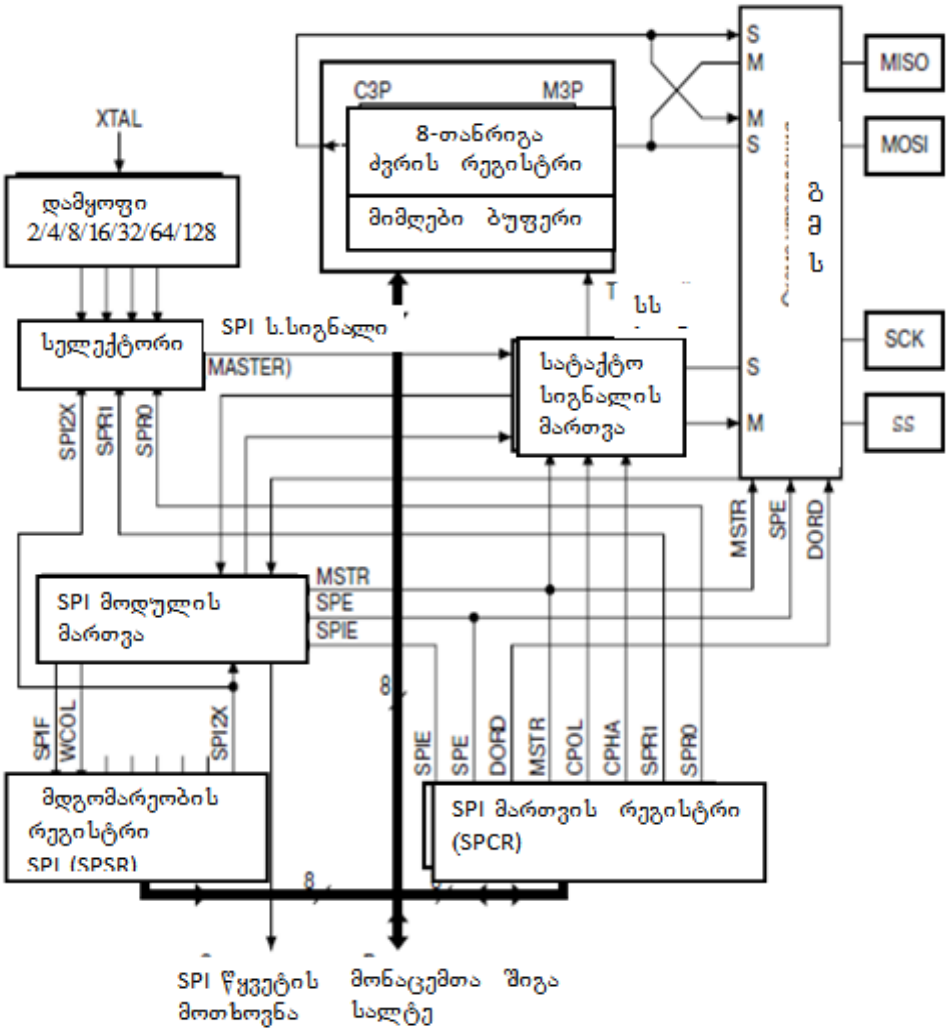
9.1. SPI მოდულის ფუნქციონირება

SPI მოდულის სტრუქტურული სქემა ნაჩვენებია 9.1.სურ-ზე. SPI მოდული გამოიყენებს ოთხ გამომყვანს. ისევე როგორც დიდი რაოდენობის სხვა პერიფერიულ მოწყობილობებისათვისაც, ეს გამომყვანები საერთო დანიშნულების შეტანა/გამოტანის პორტების სადენებია. (ცხრილი 9.1.)

SPI მოდული ჩართულ მდგომარეობაში ყოფნის დროს მითითებული გამომყვანების მუშაობა (მონაცემების გადაცემის მიმართულება) წინასწარ განისაზღვრება 9.2.ცხრილის მიხედვით, DDRB რეგისტრის შესაბამისი თანრგების მნიშვნელობით.

ცხრილი 9.1. SPI მოდულის მიერ გამოყენებული გამომყვანები

გამომყვანები	დასახელება	დანიშნულება
SCK	PB1	სატაქტო სიგნალის გამოსასვლელი (master)/ შესასვლელი(slave)
MISO	PB3	მონაცემების შესასვლელი (master)/გამოსასვ-ლელი(slave)
MOSI	PB2	მონაცემების გამოსასვლელი (master)/ შესასვლე-ლი(slave)
SS	PB0	მიმყოლი მოწყობილობის ამორჩევა



გმს- გამომყვანების მართვის სქემა
სს – სატაქტო სიგნალი

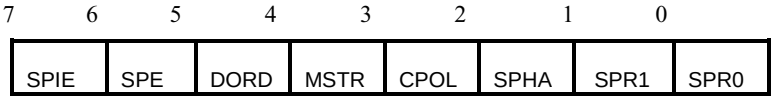
სურ.9.1. SPI მოდულის სტრუქტურული სქემა

ცხრილი 9.2. SPI მოდულის გამომყვანების მუშაობის რეჟიმის დაყენება

გამომყვანი	«Master» რეჟიმი	«Slave» რეჯიმი
MOSI	განისაზღვრება მომხმარებლის მიერ	შესასვლელი
MISO	შესასვლელი	განისაზღვრება მომხმარებლის მიერ
SCK	განისაზღვრება მომხმარებლის მიერ	შესასვლელი
SS	განისაზღვრება მომხმარებლის მიერ	შესასვლელი

როგორც 9.2. ცხრილიდან ჩანს, ზოგიერთ შემთხვევაში მომხმარებელმა თვითონ უნდა განსაზღროს გამომყვანის მუშაობის რეჟიმი, რომელსაც იყენებს SPI მოდული მისი დანიშნულების მიხედვით.

SPI მოდულის მართვისათვის განკუთვნილია SPCR მართვის რეგისტრი. აღნიშნული რეგისტრის ფორმატი მოცემულია 9.2.სურ.-ზე, ხოლო რეგისტრების თანრიგების დანიშნულება 9.3.ცხრილში.



წაკითხვა(R)/ჩაწერა(W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	0	0	0	0	0	0	0	0

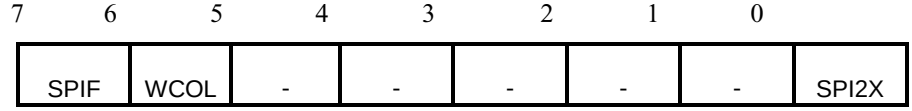
სურ.9.2. SPCR რეგისტრის ფორმატი

ცხრილი 9.3. SPCR რეგისტრის თანრიგები

თანრიგები	დასახელება	აღწერა
7	SPIE	SPI –დან წყვეტის ნებადართვა
6	SPE	SPI ჩართვა/გამორთვა
5	DORD	მონაცემების გადაცემის რიგითობა
4	MSTR	«Master»/«Slave» მუშაობის რეჟიმის ამორჩევა
3	CPOL	ტაქტური სიგნალის პოლარობა
2	SPHA	ტაქტური სიგნალის ფაზა
1,..0	SPR1-SPR0	გადაცემის სიჩქარის დაყენება

SPSR რეგისტრის დახმარებით ხორციელდება მოდულის მდგომარეობის კონტროლი და ასევე გაცვლის სიჩქარის დამატებითი მართვა. ამ რეგისტრის თანრიგები 7-დან 1 თანრიგის ჩათვლით მისაწვდომია მხოლოდ წაკითხვისათვის, ხოლო 0-თანრიგი როგორც ჩაწერის ასევე წაკითხვისათვის.

ამ რეგისტრის ფორმატი მოცემულია 9.3.სურ.-ზე, ხოლო მისი თანრიგების აღწერა 9.4.ცხრილში.



წაკითხვა(R)/ჩაწერა(W)	R	R	R	R	R	R	R	R/W
საწყისი მნიშვნელობა	0	0	0	0	0	0	0	0

სურ. 9.3. SPSR რეგისტრის ფორმატი

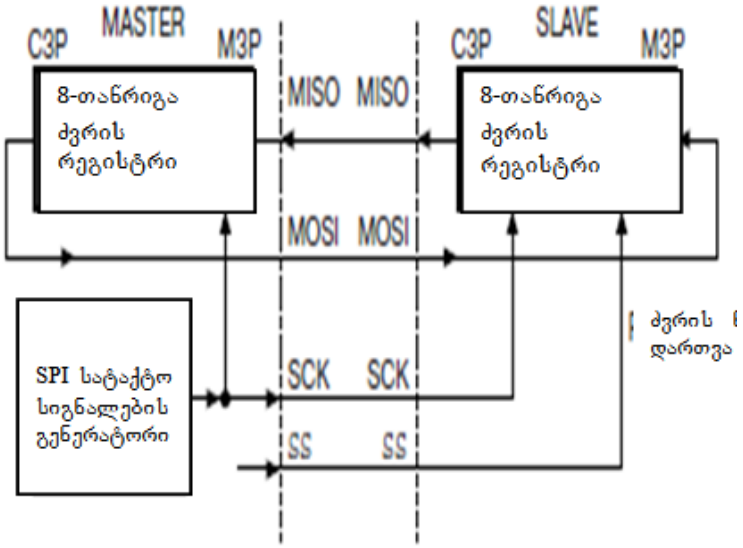
ცხრილი 9.4. SPSR რეგისტრის თანრიგები

თანრიგები	დასახელება	აღწერა
7	SPIF	**წყვეტის ალამი SPI-დან.** მოცემული ალმის ერთიანში დაყენება ხდება მიმდინარე ბაიტის გადაცემის დამთავრების შემდეგ. თუ SPCR რეგისტრის SPIE თანრიგი ერთიანის მდგომარეობაშია და წყვეტა ნებადართულია, ალმის დაყენებასთან ერთად ხორციელდება წყვეტის გენერაცია SPI –დან. ასევე SPIF თანრიგის ერთიანის მდგომარეობაში დაყენება ხდება იმ შემთხვევაში, როდესაც მიკროკონტროლერი გადადის «Master» რეჟიმიდან «Slave» რეჟიმში SS გამოსასვლელის მეშვეობით. ალმის ჩამოგდება ხდება აპარატურულად ნულიანში ან წყვეტის ქვეპროგრამის შესრულების გაშვების დროს ან SPI მდგომარეობის რეგისტრის ამოკითხვის შემდეგ, შემდგომი მიმართვით მონაცემთა SPI(SPDR) რეგისტრთან
6	WCOL	**ჩაწერის კომფლიკტის ალამი.** შესაბამისი ალმის დაყენება ხდება ერთიანში (SPDR) მონაცემთა რეგისტრში მონაცემის ჩაწერისას მიმდინარე ბაიტის გადაცემის დროს. ალამის ჩამოგდება ხდება აპარატურულად SPI მდგომარეობის რეგისტრის წაკითხვის შემდეგ, რის შემდეგაც ხორციელდება მონაცემთა რეგისტრთან მიმართვა.
5...1	-	დარეზერვირებულია, ამოიკითება როგორც ნულიანი
0	SPI2X	**გაცვლის სიჩქარის გაორმაგება.** ამ თანრიგის ერთიანში დაყენების შემდეგ და მიკროკონტროლერ-ის «Master» რეჟიმში მუშაობის შემთხვევაში SCK სიგნალის სიხშირე ორმაგდება.

გადასაცემი მონაცემები იწერება, ხოლო მიღებული – ამოიკითხება SPDR მონაცემთა რეგისტრიდან. ამ რეგისტრში ჩაწერა ახდენს გადაცემის დაწყების ინიცირებას, ხოლო წაკითხვის დროს ხდება ბუფერული ძვრის რეგისტრიდან მისი შემცველობის წაკითხვა. სხვა სიტყვებით რომ ავხსნათ, მონაცემთა რეგისტრი არის ბუფერი მიკროკონტროლერის რეგისტრულ ფაილსა და SPI მოდულის ძვრის რეგისტრს შორის. SPI ინტერფეისით ორი მიკროკონტროლერის (წამყვანი - მიმყოლი) შეერთება ნაჩვენებია. 9.4. სურ-ზე.

მონაცემთა გაცვლისათვის აუცილებელია, უპირველეს ყოვლისა, SPI მოდულის მუშაობის ნების დართვა. ამისათვის SPCR რეგისტრის SPE თანრიგში უნდა ჩაიწეროს “1”. მუშაობის რეჟიმი განისაზღვრება ამავე რეგისტრის MSTR თანრიგის მდგომარეობით: თუ თანრიგში ჩაწერილია “1”, მიკროკონტროლერი მუშაობს რეჟიმში “Master”, თუ “0” – რეჟიმში “Slave”.

მონაცემის გადაცემა ხორციელდება შემდეგი თანამიმდევრობით: წამყვანი მიკროკონტროლერის SPI მოდულის მონაცემთა რეგისტრში ჩაწერის შემდეგ გაეშვება SPI მოდულის სატაქტო იმპულსების გენერატორი, მონაცემები თანრიგ-თანრიგ გადაეცემა MOSI გამომყვანს და შესაბამისად მიეწოდება მიმყოლი მიკროკონტროლერის MOSI გამომყვანს. თანრიგების გადაცემის რიგი განისაზღვრება SPCR რეგისტრის DORD თანრიგით. თუ თანრიგი ერთიანის მდგომარეობაშია, მაშინ



სურ..9.4. მიკროკონტროლერები შეერთება SPI ინტერფეისის მეშვეობით.

პირველად გადაიცემა ბაიტის უმცროსი თანრიგი, თუ განულებულია - უფროსი თანრიგი. მიმდინარე ბაიტის უკანასკნელი თანრიგის გადაცემის შემდეგ სატაქტო იმპულსების გენერატორი წყვეტს მუშაობას. ერთდროულად ხდება (SPIF) "გადაცემა დასრულებულია" ალმის ერთიანში დაყენება. ამის შემდეგ წამყვან მიკროკონტროლერს შეუძლია შემდეგი ბაიტის გადაცემის დაწყება ან მიმყოლის SS შესასვლელს მიაწოდოს მაღალი დონის ძაბვა, რაც მიმყოლ კონტროლერს გადაიყვანს მოლოდინის რეჟიმში.

ერთდროულად წამყვანიდან მიმყოლზე მონაცემების გადაცემის დროს გადაცემა ხორციელდება უკუ მიმართულებითაც იმ შემთხვევაში, როდესაც მიმყოლის SS შესავალზე მიწოდებულია დაბალი დონის ძაბვა. ძვრის ყოველი ციკლის დროს ხორციელდება მოწყობილობებს შორის მონაცემების გაცვლა. ყოველი ციკლის ბოლოს ანალოგიურად ხდება SPIF ალამის ერთიანში დაყენება. როგორც წამყვან, ისე მიმყოლ მიკროკონტროლერებში. მიღებული ბაიტები შეინახება მიმღებ ბუფერებში შემდგომი გამოყენებისთვის. მოდულში რეალიზებულია ერთჯერადი ბუფერიზაცია გადაცემის დროს და მიღების შემთხვევაში ორმაგი ბუფერიზაცია. ეს გულისხმობს იმას რომ, გაგზავნისათვის მომზადებული მონაცემი არ შეიძლება ჩაიწეროს SPI მონაცემთა რეგისტრში მანამ არ მოხდება წინა გაცვლის ციკლის დამთავრება. გადაცემის დროს მონაცემის რეგისტრის შემცველობის შეცვლის ცდის შემთხვევაში ხდება SPSR რეგისტრის WCOL ალმის ერთიანში დაყენება. ამ ალმის განულება ხდება SPSR რეგისტრის შემცველობის წაკითხვის დამთავრებისა და შემდგომში SPI რეგისტრთან მიმართვის შემდეგ. შესაბამისად მონაცემთა მიღების დროს SPI რეგისტრიდან უნდა მოხდეს მონაცემის წაკითხვა მანამ, სანამ ძვრის რეგისტრში შემოვა მომდევნო ბაიტის ბოლო თანრიგი, წინააღმდეგ შემთხვევაში პირველი ბაიტი იქნება დაკარგული.

9.2. მონაცემთა გადაცემის რეჟიმები

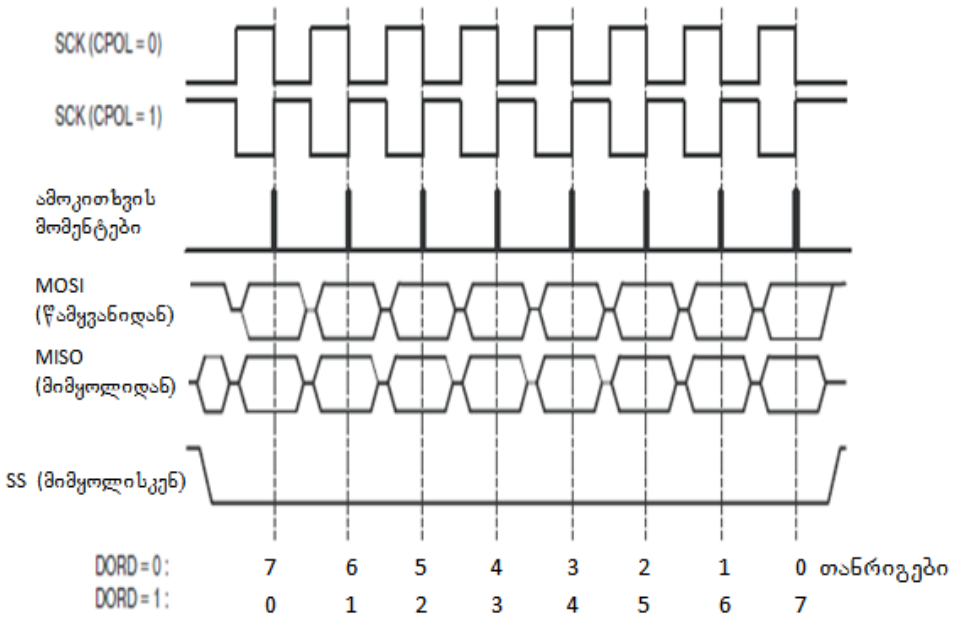
SPI ინტერფეისის სპეციფიკაცია ითვალისწინებს გადაცემის ოთხ რეჟიმს. ეს რეჟიმები განსხვავდება SCK ტაქტური სიგნალის პოლარობით და მონაცემის

ამოკითხვის მომენტით. სულ არსებობს ოთხი კომბინაცია, რომლებიც განისაზღვრება SPCR რეგისტრის CPHA და CPOL თანრიგების მდგომარეობით (ცხრილი 9.5.).

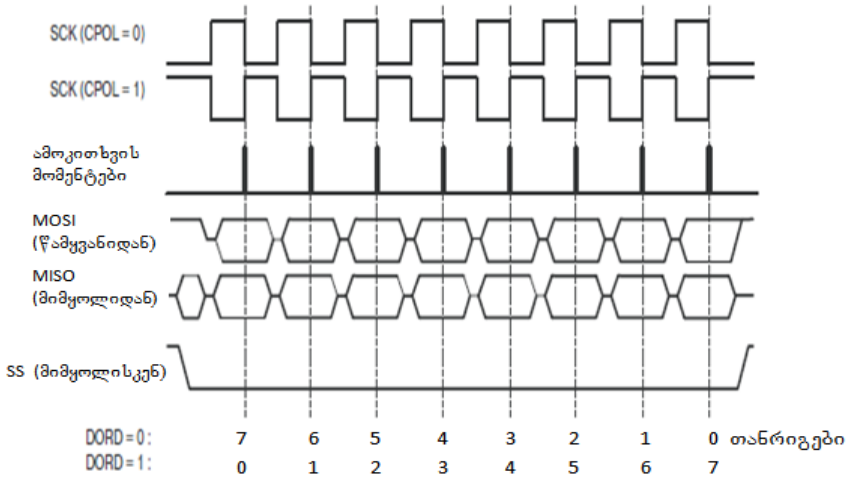
ცხრილი 9.5. მონაცემთა გადაცემის რეჟიმის დაყენება

თანრიგები	აღწერა
CPOL	**ტაქტური სიგნალის პოლარობა.** ულიანი მნიშვნელობის დროს - ხორციელდება დადებითი პოლარობის იმპულსის გენერაცია, იმპულსების არ არსებობის დროს გამოსასვლელზე ფიქსირდება დაბალი დონე; ერთიანის დროს - ხორციელდება უარყოფითი პოლარობის იმპულსის გენერაცია, იმპულსის არ არსებობის დროს გამოსასვლელზე ფიქსირდება მაღალი დონე.
CPHA	**ტაქტური სიგნალის ფაზა.** ნულიანის დროს - მონაცემების დამუშავება ხორციელდება SCK სიგნალის იმპულსის წინა ფრონტზე (თუ CPOL ტოლია ნულის, მაშინ მზარდი ფრონტით, ხოლო თუ CPOL ტოლია ერთის - მაშინ უკანა ფრონტით); ერთიანის დროს ხორციელდება მონაცემის დამუშავება SCK სიგნალის იმპულსის უკანა ფრონტზე (თუ CPOL ტოლია ნულის, მაშინ უკანა ფრონტით, ხოლო თუ CPOL ტოლია ერთის, მაშინ წინა ფრონტით)

SPI ინტერფეისით მონაცემების გაცვლის რეჟიმების შესაბამისი ფორმატები მოცემულია 9.5. და 9.6 სურათებზე. (გადაცემა ხორციელდება უფროსი თანრიგიდან უმცროსისაკენ).



სურ.9.5. მონაცემების გადაცემა, როდესაც CPHA=0.



სურ.9.6. მონაცემების გადაცემა, როდესაც CPHA = «1»

SCK ტაქტური სიგნალის სიხშირე და შესაბამისად, ინტერფეისით მონაცემების გადაცემის სიჩქარე განისაზღვრება SPCR რეგისტრის SPR1:SPR0 თანრიგების მდგომარეობით და SPSR რეგისტრის SPI2X თანრიგით (9.6.ცხრილი). ცხადია, იგულისხმება ის მიკროკონტროლერი, რომელიც მუშაობს «Master» რეჟიმში, ვინაიდან იგი არის წყარო ტაქტური სიგნალის. მოწყობილობებისთვის, რომლებიც «Slave» რეჟიმში მუშაობენ აღნიშნული თანრიგების მნიშვნელობას ყურადღება არ ექცევა.

მხედველობაში უნდა მივიღოთ ის ფაქტი, რომ «Slave» რეჟიმში მიკროკონტროლერის მუშაობა გარანტირებულია მხოლოდ $f_{CLK}/4$ ტოლ ან მასზე ნაკლებ სიხშირეზე.

ცხრილი 9.6. SCK სიგნალისათვის ტაქტური სიხშირის განსაზღვრა

SPI2X	SPR1	SPR0	SCK სიგნალის სიხშირე
0	0	0	$f_{CLK}/4$
0	0	1	$f_{CLK/16}$
0	1	0	$f_{CLK/64}$
0	1	1	$f_{CLK/128}$
1	0	0	$f_{CLK/2}$
1	0	1	$f_{CLK/8}$
1	1	0	$f_{CLK/32}$

შენიშვნა: f_{CLK} – მიკროკონტროლერის ტაქტური სიხშირეა.

SS გამომყვანის გამოყენება

ზოგადად შეიძლება აღინიშნოს, რომ ეს გამომყვანი გამოიყენება აქტიური მიმყოლი მიკროკონტროლერის ამორჩევისათვის და «Slave» რეჟიმში ყოველთვის არის შესასვლელი. მასზე დაბალი დონის ძაბვის მიწოდების შემდეგ ხდება SPI მოდულის გააქტიურება და MOSI გამომყვანი გადაირთვება მონაცემების გაცემის რეჟიმში (თუ კი ეს განსაზღრულია მომხმარებლის მიერ). ამ რეჟიმში SPI მოდულის დანარჩენი გამომყვანები შესასვლელებია. თუ SS გამომყვანს მივაწვდით მაღალი დონის ძაბვას,

მაშინ SPI მოდულის ყველა გამომყვანი გადაირთვება მონაცემის მიღების რეჟიმში. ამ შემთხვევაში მოდული გადადის პასიურ მდგომარეობაში და მონაცემის მიღება არ ხორციელდება. როგორც წესი, ამ მდგომარეობაში, პროგრამა ცვლის მონაცემთა რეგისტრის შემცველობას.

საჭიროა გვახსოვდეს, რომ ყოველთვის, როდესაც SS შესასვლელს მიეწოდება მაღალი დონის ძაბვა, ხორციელდება SPI მოდულის განულება. შესაბამისად, თუ კი ამ გამომყვანის მნიშვნელობის შეცვლა მოხდება მონაცემების გადაცემის ან მიღების დროს, გადაცემა მყისვე შეწყდება, გადაცემული და მიღებული ბაიტები დაიკარგებიან.

თუ მიკროკონტროლერი არის «Master» რეჟიმში (SPCR რეგისტრის MSTR თანრიგი ერთიანშია დაყენებული), მონაცემების გადაცემის მიმართულება SS გამომყვანით განისაზღვრება მომხმარებლის მიერ. თუ გამომყვანი დაკონფიგურირებულია როგორც გამოსასვლელი, მაშინ ის მუშაობს როგორც საერთო დანიშნულების გამომყვანი, რომელიც ზემოქმედებას არ ახდენს SPI მოდულის მუშაობაზე, როგორც წესი ასეთ შემთხვევაში ის გამოიყენება მიკროკონტროლერის SS გამომყვანით მართვისათვის, რომელიც მუშაობს «Slave» რეჟიმში.

თუ გამომყვანი დაკონფიგურირებულია როგორც შესასვლელი, მაშინ SPI მოდულის ნორმალური მუშაობის უზრუნველსაყოფად მას უნდა მივაწოდოთ მაღალი დონის ძაბვა, რომელიღაც გარე სქემიდან. აღნიშნულ შესასვლელზე დაბალი დონის ძაბვის მიწოდების შემთხვევაში SPI მოდულის მიერ აღქმული იქნება როგორც ამორჩეული მიკროკონტროლერი მიმყოლის როლში და მისთვის მონაცემის გადაცემის დასაწყისად. კონფლიქტის აღმოფხვრის მიზნით სალტეებზე SPI მოდული ასეთ შემთხვევაში ასრულებს შემდეგ მოქმედებებს:

1.ხორციელდება SPCR რეგისტრის MSTR ალამის ჩამოგდება, და მიკროკონტროლერი გადაირთვება «Slave» რეჟიმში. როგორც შედეგი , MOSI და SCK გამომყვანები იწყებენ ფუნქციონირებას როგორც შესასვლელები.

2.ხორციელდება SPSR რეგისტრის MSTR ალმის დაყენება, SPI-დან ხდება წყვეტის გენერაცია, თუ კი ნებადართულია წყვეტა SPI-დან და SREG რეგისტრის I ალამი ერთიანშია დაყენებული. ასეთ შემთხვევაში ხორციელდება წყვეტის დამუშავების ქვეპროგრამის გაშვება.

თუ წამყვანი მიკროკონტროლერი იყენებს მონაცემების გადაცემას წყვეტით მართვით და არსებობს იმის ალბათობა, რომ SS შესასვლელს მიეწოდოს დაბალი დონის ძაბვა, მაშინ SPI მოდულიდან აუცილებლად უნდა მოხდეს წყვეტის დამუშავების ქვეპროგრამაში MSTR ალმის მდგომარეობის შემოწმება. შემოწმების შედეგად თუ აღმოჩნდება, რომ ეს ალამი განულებულია, უნდა მოხდეს მისი დაყენება ერთიანის მდგომარეობაში, რათა მიკროკონტროლერი დაუბრუნდეს «Master» რეჟიმს.

თავი 10

მიმდევრობითი ორგამტარიანი ინტერფეისი (TWI)

10.1 საერთო ცნობები

ორგამტარიანი მიმდევრობითი ინტერფეისის მოდული (Two_wire Serial Interface, TWI) შედის მიკროკონტროლერ Atmega 128 შემადგენლობაში. მოცემული ინტერფეისი არის ფირმა Philips I^2C ინტერფეისის ბაზური ვერსიის სრული ანალოგი. ინტერფეისი TWI გვაძლევს საშუალებას გავაერთიანოთ 128 სხვადასხვა მოწყობილობა ორმიმართულებიანი სალტის საშუალებით, რომელიც იყენებს ორ გამტარს: ტაქტური სიგნალების (SCL) გამტარს და (SDA) მონაცემთა გამტარს. დამატებით ელემენტად სალტის რეალიზაციისათვის გამოყენებულია ორი მომჭიმავი რეზისტორი თითოეული გამტარისათვის ცალ-ცალკე (სურ.10.1.).

სურ. 10.1. მოწყობილობების გაერთიანება საერთო სალტით

სალტური მაფორმირებელი TWI ყველა თავსებადი მოწყობილობებისათვის შესრულებულია ე.წ. ღია კოლექტორის სქემის გამოყენებით, რაც იძლევა "სამონტაჟო და" ფუნქციის რეალიზაციის საშუალებას. შესაბამისად დაბალი დონე გამტარზე არის იმ შეთხვევაში თუ ერთი ან მეტი მოწყობილობა გამტარზე გასცემს ლოგიკური ნულის სიგნალს, ხოლო მაღალი დონე გამტარზე არის იმ შემთხვევაში, როდესაც მასზე მიერთებულ ყველა მოწყობილობას გამოსასვლელებზე მესამე მდგომარეობა უკავია.

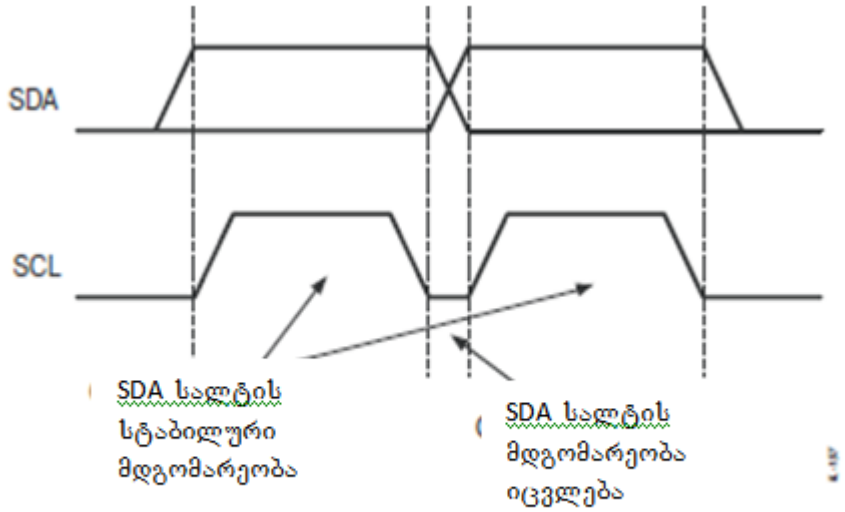
TWI შემდგომი განხილვის დროს ხშირად შეგვხვდება ზოგიერთი ტერმინები, რომელთა განმარტება მოცემულია 10.1.ცხრილში.

ცხრილი 10.1., TWI მოდულის აღწერის დროს გამოყენებული ტერმინები.

ტიპი	აღწერა
წამყვანი(Master)	მოწყობილობა, რომელიც ახდენს სალტით მონაცემების გადაცემის ინიცირებას და გადაცემის დასრულებას, აგრეთვე აფორმირებს SCL სატაქტო სიგნალებს.
მიმყოლი(Sleve)	მოწყობილობა, რომელსაც მიმართავს წამყვანი.
გადამცემი(Transmitter)	მოწყობილობა, რომელიც გასცემს მონაცემებს სალტეზე.
მიმღები (Resiver)	მოწყობილობა, რომელიც იღებს მონაცემებს სალტიდან.

10.2 . TWI სალტით მონაცემთა გაცვლის პრინციპი

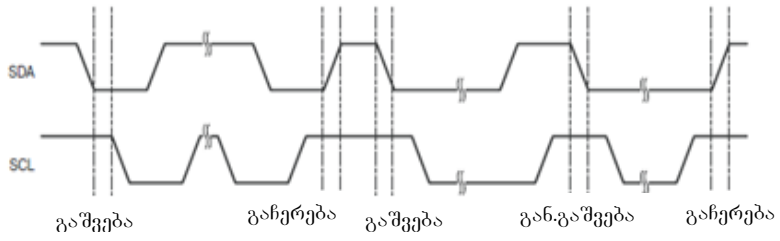
რამდენადაც TWI სალტე არის მიმდევრობითი, ყველა მონაცემი გადაიგზავნება თანრიგ მიმდევრობით SDA გამტარის საშუალებით. ყოველ გადასაცემ თანრიგს თან ახლავს SCL გამტარზე თანმხლები სიგნალი. ამასთანავე SDA გამტარზე სიგნალი უნდა იყოს სტაბილური მთელი იმ დროის განმავლობაში, სანამ SCL გამტარზე არის ლოგიკური ერთიანის სიგნალი (სურ.10.2.). აღნიშნული წესის გამონაკლისია TWI სალტის "გაშვების" და "გაჩერების" განსაკუთრებული მდგომარეობა.



სურ.10.2. SDA და SCA სალტეებზე სიგნალების დიაგრამა

"გაშვების" ("Start") და "გაჩერების" ("Stop") მდგომარეობის ფორმირება ხორციელდება წამყვანის მიერ მონაცემების გადაცემების შემთხვევაში, შესაბამისად დასაწყისში და დამთავრების დროს. ამ მდგომარეობებს შორის სალტე ითვლება დაკავებულად და სხვა წამყვანებს არ უნდა ჰქონდეთ მცდელობა, რომ მართონ იგი. იმ შემთხვევაში, როდესაც წამყვანს სურს ახალი მონაცემის ბლოკის გადაცემის დაწყება სალტეზე კონტროლის დაკარგვის გარეშე, მას შეუძლია დააფორმიროს "გაშვების" მდგომარეობა "გაჩერების" მდგომარეობის ფორმირებამდე. ასეთი სახით მდგომარეობის დაფორმირებას ეწოდება "განმეორებითი გაშვება" (განგაშვება), ხოლო სალტე ითვლება დაკავებულად "გაჩერების" მდგომარეობის ფორმირებამდე.

"გაშვებისა" და "გაჩერების" მდგომარეობები ფორმირდებიან სიგნალების დონეების ცვლილების გზით SDA სალტეზე იმ შემთხვევაში, როდესაც SCL სალტეზე მაღალი დონეა. "გაშვების" მდგომარეობას შეესაბამება დონის ცვლილება ერთიდან ნულში, ხოლო "გაჩერებას" - პირიქით ნულიდან ერთში (სურ.10.3.).



სურ.10.3. “გაშვების” და “გაჩერების” მდგომარეობის დიაგრამა.

უნდა აღინიშნოს რომ TWI ინტერფეისის პროტოკოლი გვაძლევს საშუალებას სალტეს მიუერთოთ რამდენიმე წამყვანი მოწყობილობა (Multi-Master რეჟიმი). ამ დროს შესაძლებელია წარმოიქმნას პრობლემები. ერთ-ერთი მათგანია სხვადასხვა წამყვანების მიერ გენერირებული ტაქტური სიგნალების სიხშირეების არათანხვედრა. მათი სინქრონიზაცია ხორციელდება მარტივად, ვინაიდან ყველა მოწყობილობა მიერთებულ SCL გამტართან “სამონტაჟო და " სქემით. შედეგად ტაქტური იმპულსის ხანგრძლივობა SCL გამტარზე განისაზღვრება უმცირესი ხანგრძლივობის მქონე სატაქტო სიგნალების იმპულსებით, ხოლო პაუზა ტაქტურ იმპულსებს შორის - უდიდესი პაუზით.

სხვა პრობლემა, რომლის გადაწყვეტაც საჭირო ხდება სალტესთან რამდენიმე წამყვანი მოწყობილობის მიერთების დროს, არის პრიორიტეტების განაწილება მიკროკონტროლერებს შორის, იმ შემთხვევაში როდესაც ორი ან მეტი წამყვანი მოწყობილობა ერთდროულად ცდილობს დაიწყოს გადაცემა. ასეთი სიტუაციის წარმოქმნის დროს გადაცემა შეიძლება განახორციელოს მხოლოდ ერთმა წამყვანმა, დანარჩენები უნდა გადაერთონ მიმყოლის რეჟიმში. ამასთან, პრიორიტეტების გადანაწილების დროს გადაცემული მონაცემები არ უნდა დამახინჯდეს.

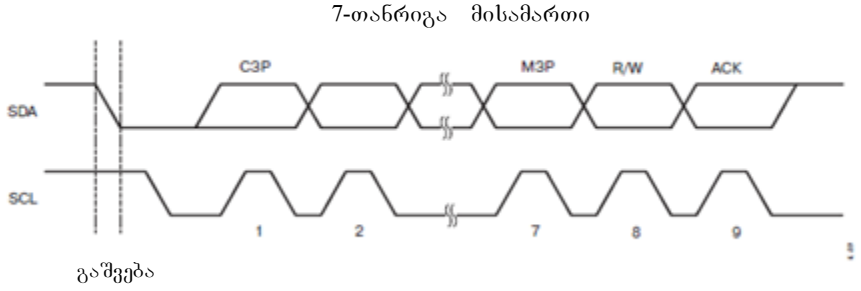
აღნიშნული ამოცანის გადაწყვეტის დროს ყველა წამყვანი მოწყობილობა მონაცემების გაცემის შემდეგ ამოწმებს SDA გამტარის მდგომარეობას. თუ გამტარის მდგომარეობა განსხვავდება იმ მდგომარეობისაგან, რომელშიც წამყვანი ცდილობს მის გადაყვანას, მაშინ ის კარგავს პრიორიტეტს. ეს შესაძლებელია მხოლოდ იმ შემთხვევაში, როდესაც მის მიერ გაიცემა მაღალი დონის სიგნალი, და სხვა წამყვანებს (რომლებმაც მიიღეს პრიორიტეტი) აქვთ დაბალი დონის სიგნალი. წამყვანი, რომელიც დაკარგავს პრიორიტეტს უნდა გადაირთოს მიმყოლის რეჟიმში და შეამოწმოს იყო თუ არა ის დამისამართებული რომელიმე წამყვანის მიერ, რომელმაც მიიღო პრიორიტეტი.

პრიორიტეტების განაწილების პროცესი გაგრძელდება მანამ, სანამ სალტეზე არ დარჩება მხოლოდ ერთი წამყვანი. იმ შემთხვევაში როდესაც რამოდენიმე წამყვანი ცდილობს დაამისამართოს ერთი და იგივე მიმყოლი, პრიორიტეტების განაწილების პროცესი გრძელდება მონაცემების პაკეტების გადაცემის დროსაც. აქედან გამომდინარეობს რომ, კერძოდ, ყველა გაცვლის ციკლს უნდა ჰქონდეს ერთნაირი მონაცემების პაკეტების რაოდენობა, წინააღმდეგ შემთხვევაში პრიორიტეტების განაწი-ლების პროცესი გაურკვეველი იქნება.

TWI სალტით მონაცემის გადაცემის დროს, მასთან ერთად გადაიცემა სამომსახურო ინფორმაცია. მონაცემების და შესაბამისი სამომსახურო ინფორმაციის ერთობლიობას პაკეტი ქვია. განასხვავებენ სამისამართო და მონაცემთა პაკეტებს.

სამისამართო პაკეტის ფორმატი

ყველა სამისამართო პაკეტს, რომელიც გადაიცემა TWI სალტით აქვს 9 ბიტიანი სიგრძე, აქედან 7 თანრიგი მისამართისათვისაა (პირველად გადაიცემა უფროსი თანრიგი), R/W მმართველი ბიტი და ACK კვიტირების ბიტი (სურ.10.4.). მმართველი R/W ბიტი განსაზღვრავს მონაცემის გადაცემის მიმართულებას სალტეზე. R/W განულებული თანრიგი იმის მაჩვენებელია, რომ წამყვანი მოწყობილობის მიერ შემდგომში სალტეზე მოხდება მონაცემის გადაცემა, ხოლო ერთიანში დაყენებული – მიმყოლიდან მონაცემის წაკითხვა. სამისამართო პაკეტები, რომელთა მმართველი ბიტი განულებული ან ერთიანის მდგომარეობაშია დაყენებული შესაბამისად აღინიშნება SLA+W და SLA+R:.



სურ.10.4. სამისამართო პაკეტის ფორმატი

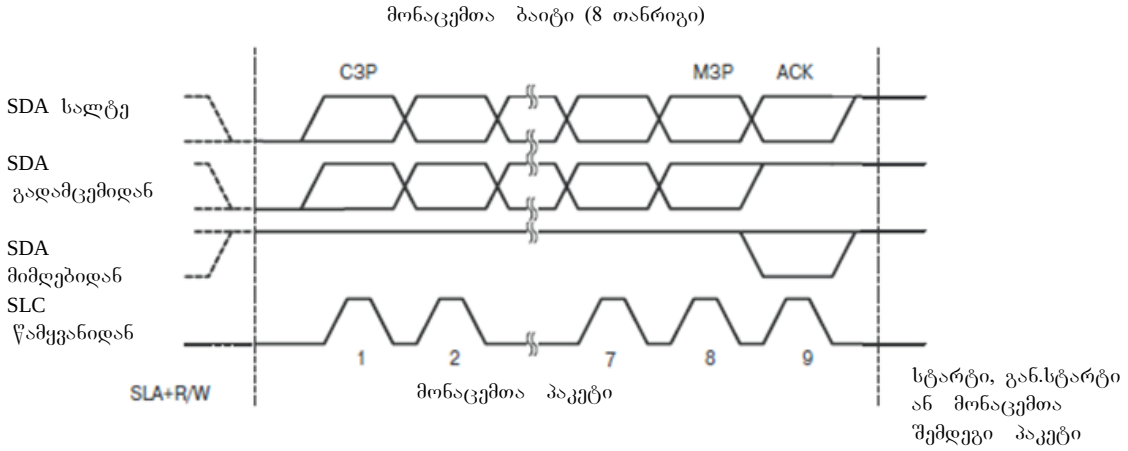
იმ შემთხვევაში, როდესაც მიმყოლი ამოიცნობს თავის მისამართს, მან უნდა საპასუხოდ დააფორმიროს დადასტურება SDA გამტარზე დაბალი დონის სიგნალის დაფორმირებით მე-9 (ACK) სატაქტო იმპულსის განმავლობაში. თუ მიმყოლს რაიმე მიზეზის გამო არ შეუძლია მოემსახუროს წამყვანის მოთხოვნას, მან მე-9 სატაქტო იმპულსის დროს უნდა შეინარჩუნოს გამტარზე მაღალი დონის სიგნალი (NACK). მიმყოლ მოწყობილობებს შესაძლებელია მივანიჭოთ ნებისმიერი მისამართი გარდა ნული მისამართისა და მისამართებისა დიაპაზონიდან “1111000.....1111111”. ნული მისამართი დარეზერვირებულია ეგრეთწოდებული საერთო გამოძახების რეალიზაციისთვის, ხოლო “1111xxxx” მისამართები დარეზერვირებულია შემდგომი გამოყენებისთვის.

საერთო გამოძახება გამოიყენება, იმ შემთხვევაში როდესაც წამყვანს სურს შეტყობინება გადასცეს ყველა იმ მიმყოლებს, რომლებიც მიერთებულია სალტზე. ისეთი პაკეტის მიღებისას, რომელსაც აქვს ნულიანი მისამართი და განულებული მმართველი ბიტი (მოთხოვნა გადაცემაზე) ყველა მიმყოლი მოწყობილობა ვალდებულია დააფორმიროს დასტური. გამონაკლისია ის მოწყობილობები, რომლებსაც საერთო გამოძახების გამოცნობა რაღაც მიზეზების გამო აკრძალული აქვთ. შესაბამისად, მომდევნო მონაცემების პაკეტების გადაცემის დროს, რომელსაც

აგზავნის წამყვანი, მიღებული იქნება ყველა იმ მიმყოლი მოწყობილობის მიერ, რომლებმაც აღიქვეს საერთო გამოძახების მისამართი. ცხადია, რომ საერთო გამოძახებას ერთიანში დაყენებული მმართველი თანრიგით (მოთხოვნა წაკითხვაზე) აზრი არა აქვს, იქიდან გამომდინარე, რომ სხვადასხვა მოწყობილობას ერთდროულად არ შეუძლიათ განახორციელოს მონაცემების გადაცემა სალტით.

მონაცემების პაკეტის ფორმატი

ყველა მონაცემთა პაკეტი, რომელიც გადაიცემა TWI სალტით აგრეთვე 9 ბიტიანია. პაკეტი შედგება მონაცემთა ბაიტისაგან (გადაცემა იწყება უფროსი თანრიგიდან) და ACK კვიტირების ბიტისაგან (სურ..10.5.).



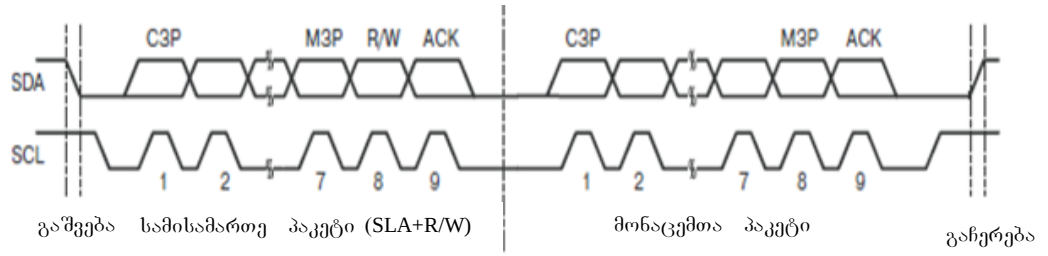
სურ.10.5. მონაცემთა პაკეტის ფორმატი

ტაქტური სიგნალების გენერაცია, ასევე გაშვების და გაჩერების მდგომარეობის ფორმირება ხორციელდება წამყვანის მიერ. მიმღებმა თავის მხრივ, ყოველი ბაიტის მიღების შემდეგ მე-9 სატაქტო იმპულსის დროს უნდა დააფორმიროს დასტური SDA გამტარზე დაბალი დონის სიგნალის გაცემით (ACK). თუ მიმღებმა მიიღო უკანასკნელი ბაიტი ან რაღაც სხვა მიზეზების გამო არა აქვს საშუალება გააგრძელოს მონაცემების მიღება, მან მე-9 ტაქტური იმპულსის განმავლობაში უნდა შეინარჩუნოს გამტარზე მაღალი დონის სიგნალი (NACK-არდადასტურება). როდესაც წამყვანი არ მიიღებს დასტურს მიმღებიდან, მას შეუძლია შეწყვიტოს მონაცემების გადაცემა და დააფორმიროს გაჩერების მდგომარეობა.

პრაქტიკაში TWI სალტით გაცვლის ყოველი ციკლი შედგება შემდეგი ეტაპებისგან (სურ.10.6.):

1) გაშვების მდგომარეობის ფორმირება;
2) SLA+R/W მისამართის პაკეტის გადაცემა;
3) ერთი ან რამდენიმე მონაცემთა პაკეტის გადაცემა;
4) გაჩერების მდგომარეობის ფორმირება.

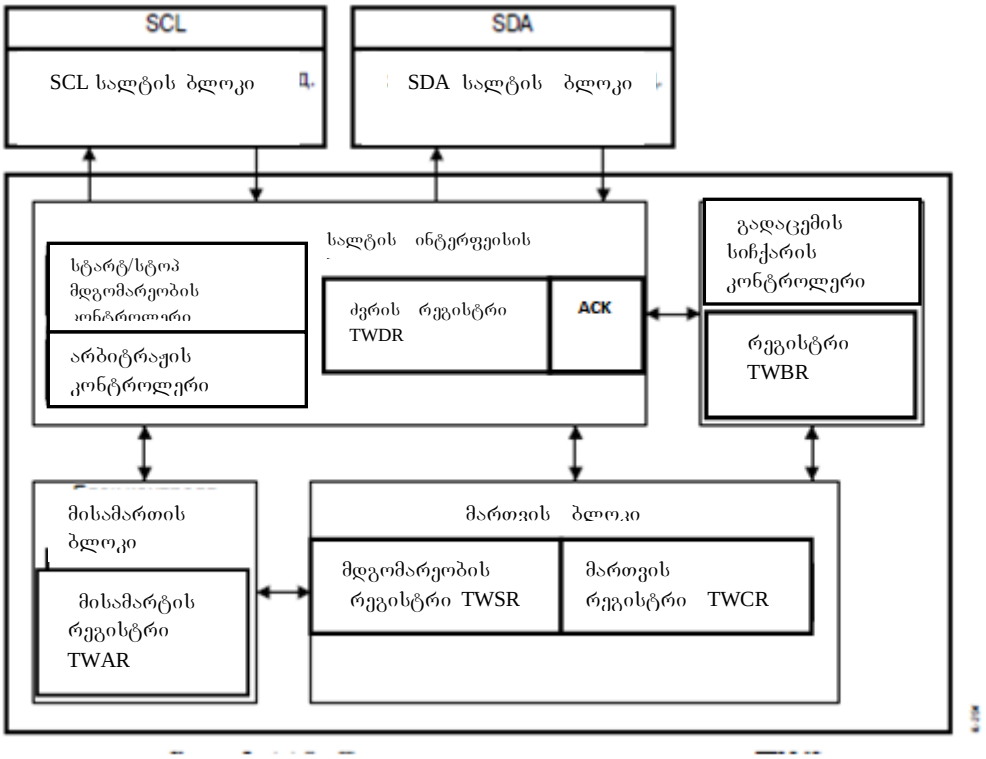
TWI სალტით გაცვლის სიჩქარეს განსაზღვრავს წამყვანი, რამდენადაც ის ახდენს ტაქტური იმპულსების გენერაციას. იმ შემთხვევაში თუ მიმღებს არ შეუძლია მიიღოს მოცემული სიჩქარით ან მიმღებს სჭირდება დრო მონაცემების დამუშავებისათვის მონაცემების მიღებებს შორის, მას შეუძლია გაზარდოს პაუზის დრო ტაქტურ იმპულსებს შორის, SCL გამტარზე დაბალი დონის სიგნალის შენარჩუნებით. რაც ტაქტური სიგნალების იმპულსების ხანგრძლივობაზე არ მოქმედებს.



სურ.10.6. TWI ინტერფეისით გადაცემის ციკლის დიაგრამა

10.3.TWI მოდულის სტრუქტურა

TWI მოდულის სტრუქტურული სქემა მოცემულია 10.7.სურ.-ზე. როგორც ნახაზიდან ჩანს, მოდული შედგება რამდენიმე ბლოკისაგან, სადაც თითოეული ასრულებს განსაზღრულ ფუნქციას.



სურ.10.7. TWI მოდულის სტრუქტურული სქემა

ურთიერთქმედება პროგრამასა და TWI მოდულს შორის ხორციელდება ხუთი შეტანა/გამოტანის რეგისტრის საშუალებით. 10.2.ცხრილში მოცემულია ამ რეგისტრების დასახელება და დანიშნულება.

ცხრილ 10.2. TWI მოდულის შეტანა/გამოტანის რეგისტრები

რეგისტრი	დანიშნულება
TWBR	გადაცემის სიჩქარის რეგისტრი
TWSR	მდგომარეობის რეგისტრი
TWAR	მისამართების რეგისტრი
TWDR	მონაცემების რეგისტრი
TWCR	მართვის რეგისტრი

SCL და SDA გამომყვანები

ეს გამომყვანები გამოიყენება TWI მოდულის მისაერთებლად სალტის ერთსახელა გამტარებთან. TWI ინტერფეისის სპეციფიკაციის შესაბამისად გამოსასვლელი კასკადები, რომლებიც მიერთებულია გამომყვანებთან, შეიცავს სიგნალის სიჩქარის ზრდის შემზღუდველს, ხოლო შესასვლელ კასკადებს აქვთ ფილტრი, რომლებიც თრგუნავენ პარაზიტულ იმპულსებს. ორივე გამომყვანი შეთავსებულია მიკროკონტროლერის შეტანა/გამოტანის პორტების გამტარებთან: SDA მონაცემების სალტე - PD1 გამომყვანთან, SCL სატაქტო სიგნალის სალტე - PD0 გამომყვანთან.

გადაცემის სიჩქარის კონტროლერი (Bit Rate Generator)

ეს ბლოკი განსაზღვრავს იმპულსების მიმდევრობის პერიოდს SCL სალტეზე იმ შემთხვევაში, როდესაც მიკროკონტროლერი მუშაობს წამყვანის რეჟიმში. TWBR რეგისტრთან ერთად ამ მიზნით ასევე გამოიყენება TWSR რეგისტრის (TWPS1:TWPS0) ორი უმცირესი თანრიგი. მოდულებისთვის SCL სიგნალის სიხშირე განისაზღვრება გამოსახულებით:

$$f_{SCL} = f_{CLK}/(16 + 2TWBR \cdot 4^{TWPS}),$$

სადაც f_{CLK} არის პროცესორის სატაქტო იმპულსი; *TWBR* — *TWBR* რეგისტრში ჩაწერილი მნიშვნელობა; TWPS - TWSR რეგისტრის TWPS1:TWPS0 თანრიგების მნიშვნელობა. მიმყოლის რეჟიმში მუშაობის დროს აღნიშნულ თანრიგების შემცველობა მხედველობაში არ მიიღება, მაგრამ მიკროკონტროლერის ტაქტური სიხშირე ამ შემთხვევაში, როგორც მინიმუმი 16 –ჯერ მეტი უნდა იყოს SCL სიგნალის სიხშირეზე.

ნებისმიერი მოწყობილობას, რომელიც მიერთებულია TWI სალტესთან, აქვს საშუალება გაზარდოს ტაქტურ იმპულსებს შორის პაუზის ხანგრძლივობა, რაც თავისთავად ამცირებს სალტეზე მონაცემთა გადაცემის სიჩქარეს.

სალტის ინტერფეისის ბლოკი (Bus Interface Unit)

ამ ბლოკის შემადგენლობაში შედის ორი კვანძი:

- "გაშვების"/"გაჩერების" მდგომარეობის კონტროლერი, რომელიც აფორმირებს და აღმოაჩენს "გაშვების", "განმეორებითი გაშვების" და "გაჩერების" მდგომარეობებს. სალტეების მდგომარეობის შემოწმება ხორციელდება იმ შემთხვევაშიაც, როდესაც მიკროკონტროლერი იმყოფება "ძილის" რეჟიმში. ამის გამო საჭიროების შემთხვევაში შესაძლებელი ხდება მიკროკონტროლერის გამოყვანა "ძილის" რეჟიმიდან, როდესაც მის დამისამართებას ანხორციელებს რომელიღაც წამყვანი;
- არბიტრაჟის კონტროლერი განსაზღვრავს სალტეზე კონფლიკტის არსებობას, როდესაც კონტროლერი მუშაობს წამყვანის რეჟიმში. იმ შემთხვევაში როდესაც მოწყობილობა კარგავს პრიორიტეტს კონტროლერი ამის შესახებ ინფორმაციას აწვდის მართვის ბლოკს, რომელიც ასრულებს აუცილებელ მოქმედებებს და აფორმირებს მდგომარეობის შესაბამის კოდებს.

გარდა ამისა, ბლოკის შემადგენლობაში შედის მისამართის/მონაცემების ძვრის TWDR რეგისტრი, რომლიც შეიცავს გადასაცემი ან მისაღები მონაცემების პაკეტს. რეგისტრის შემცველობის მონაცემთა სალტეზე გაცემასთან ერთად, ამავე სალტიდან მონაცემები ჩაიწერებიან რეგისტრში. კვების ჩართვის დროს ხორციელდება ამ რეგისტრის ყველა თანრიგების "ერთიანში" დაყენება.

გარდა TWDR რეგისტრისა, ბლოკის შემადგენლობაში შედის სპეციალური რეგისტრი, რომლის ერთ-ერთი თანრიგი შეიცავს გადაცემული ან მიღებული დასტურის (ACK) ბიტის მნიშვნელობას. მიღების დროს ამ ბიტის მდგომარეობა განისაზღვრება TWCR მმართველი რეგისტრის ერთ-ერთ თანრიგით, ხოლო გადაცემისას მიღებული ბიტის მდგომარეობა შეიძლება განისაზღვროს კოდის მეშვეობით, რომელიც განთავსებულია TWSR მდგომარეობის რეგისტრში.

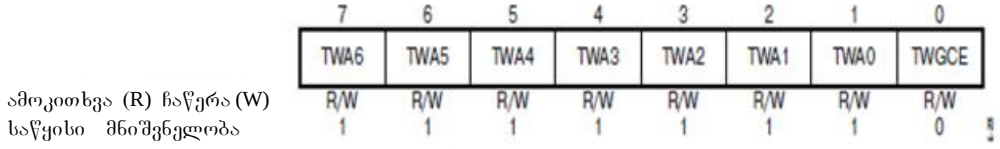
მისამართის კონტროლის ბლოკი (Address Match Unit)

მისამართის კონტროლის ბლოკი ამოწმებს მიღებული მისამართის შესაბამისობას TWAR მისამართების რეგისტრის უფროს შვიდ თანრიგში განთავსებულ მნიშვნელობასთან. ასევე ამოწმებს საერთო გამოძახების არსებობას იმ შემთხვევაში, როდესაც ნებადართულია მათი გამოცნობა.

კორექტული მისამართის აღმოჩენის შემთხვევაში იგი ინფორმაციას აწვდის მართვის ბლოკს ამის შესახებ, რომელიც აფორმირებს ან არ აფორმირებს დადასტურებას მისამართის პაკეტის მიღებაზე.

ბლოკი ანხორციელებს მისამართის პაკეტის კონტროლს იმ მომენტშიაც კი როდესაც მიკროკონტროლერი იმყოფება "ძილის" მდგომარეობის რეჟიმში, რაც იძლევა საშუალებას, რომ წამყვანი მოწყობილობის მიერ დამისამართების შემთხვევაში მიკროკონტროლერი გადავიყვანოთ მუშა რეჟიმში.

TWAR რეგისტრის ფორმატი ნაჩვენებია 10.8.სურ.-ზე, ხოლო მისი თანრიგების აღწერა - 10.3.ცხრილში.



სურ.10.8. TWAR რეგისტრის ფორმატი

ცხრილი 10.3. TWAR რეგისტრის თანრიგების დანიშნულება

თანრიგები	დანიშნულება	აღწერა
7...1	TWA6-TWA0	**მოწყობილობის მისამართი.** ამ თანრიგების შემცველობა არის მისამართი, რომელზედაც რეაგირებას ახდენს მიმყოლის რეჟიმში მომუშავე მოწყობილობა. ხოლო იმ მოწყობილობებისთვის, რომლებიც მუშაობს წამყვანის რეჟიმში რეგისტრის შემცველობას არავითარი მნიშვნელობა არ გააჩნია.
0	TWGCE	**საერთო გამოძახებების ნებადართვის ამოცნობა.** თუ ეს თანრიგი დაყენებულია ერთიანის მდგომარეობაში, მოწყობილობა გამოეხმაურება საერთო გამოძახებებს (პაკეტები $00 მისამართით), ასევე გამოძახებებს მისამართით, რომლებიჩ განთავსებულია TW6..0 მისამართით.

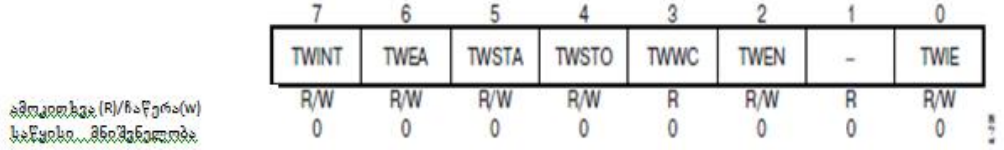
მართვის ბლოკი (Control Unit)

ეს ბლოკი ანხორციელებს TWI ყველა მოდულის მართვას TWCR მმართველი რეგისტრის თანრიგების დაყენების შესაბამისად და იმ ინფორმაციის საფუძველზე, რომელსაც ის ღებულობს მოდულის ცალკეული ბლოკებისაგან. გარკვეული მოვლენის წარმოქმნის შემთხვევაში, რომელიც მითითებულია ქვემოთ, მართვის ბლოკი TWSR მდგომარეობის რეგისტრში აფორმირებს მოვლენის შესაბამის მდგომარეობის კოდს და აყენებს TWCR რეგისტრში TWINT წყვეტის მოთხოვნის ალამს. ამ ალმის განულების მომენტამდე SCL გამტარზე ხდება დაბალი დონის სიგნალის შენარჩუნება, რის საფუძველზეც ხორციელდება სალტეზე მონაცემთა გადაცემის შეჩერება

წყვეტაზე მოთხოვნის ფორმირება ხდება შემდეგი მოვლენების წარმოქმნის დროს:

- "გაშვება"/"განგაშვების" მდგომარეობის ფორმირების დასრულების შემდეგ;
- სამისამართო (SLA+R/W) პაკეტის გადაცემის დასრულების შემდეგ;
- მისამართის პაკეტის ბაიტის გადაცემის დასრულებისას;
- მოწყობილობის მიერ პრიორიტეტის დაკარგვისას;
- მოწყობილობის დამისამართების ან საერთო გამოძახების არსებობისას;
- მონაცემის ბაიტის მიღების დასრულებისას;
- სალტეზე შეცდომის წარმოქმნის დროს, გამოწვეული «გაშვება» /«გაჩერების» მდგომარეობის ფორმირების დაუშვებელი პირობებით.

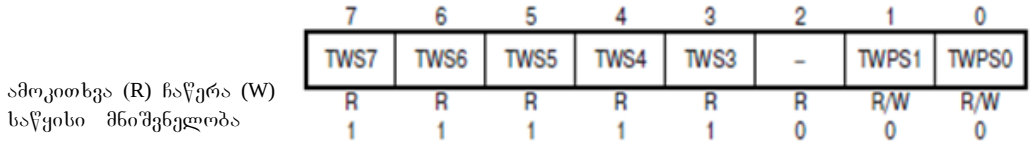
TWCR რეგისტრის ფორმატი ნაჩვენებია 10.9.სურ..-ზე, ხოლო მისი თანრიგების აღწერა მოცემულია 10.4.ცხრილში. TWSR რეგისტრის ფორმატი ნაჩვენებია 10.10.სურ. -ზე, მისი თანრიგების აღწერა - 10.5.ცხრილში.



სურ.10.9. TWCR რეგისტრის ფორმატი

ცხრილი 10.4. TWCR მმართველი რეგისტრის თანრიგების დანიშნულება

თანრიგები	დასახელება	აღწერა
7	TWINT	**წყვეტის ალამი TWI მოდულისაგან.** ამ ალმის დაყენება ხორციელდება აპარატურულად მიმდინარე ოპერაციის შესრულების შემდეგ, როდესაც მოდული ელოდება გამოხმაურებას პროგრამის მხრიდან. იმ შემთხვევაში, თუ დაყენებულია SREG რეგისტრის I ალამი და TWCR რეგისტრის TWIE ალამი, მაშინ ხდება წყვეტის გენერაცია და ხორციელდება შესაბამისი დამმუშავებლის გამოძახება. სანამ TWINTალამი დაყენებულია, SCL სალტეზე შენარჩუნდება დაბალი დონის სიგნალი.. ალმის ჩამოგდება შეიძლება განხორციელდეს მხოლოდ მასში ლოგიკური ერთიანის ჩაწერით.
6	TWEA	**დასტურის ბიტის ნებადართვა.** TWEA თანრიგი ახდენს დასტურის ბიტის ფორმირების მართვას. თუ ეს თანრიგი დაყენებულია ერთიანში, მაშინ მოწყობილობა აფორმირებს დასტურის სიგნალს საჭიროების შემთხვევაში. თანრიგის ნულში დაყენების შემთხვევაში მოწყობილობა გამოირთვება TWI სალტიდან, ე.ი. დასტურის ბიტი არ ფორმირდება.
5	TWSTA	**«გაშვების »მდგომარეობის ალამი.** იმ შემთხვევაში, როდესაც TWSTA თანრიგში ჩაწერილია ლოგიკური ერთიანი, მოდული ამოწმებს TWI სალტის მდგომარეობას, თუ სალტე თავისუფალია, აფორმირებს «გაშვების » მდგომარეობას, თუ სალტე დაკავებულია ,მაშინ TWI მოდული ელოდება «გაჩერების » მდგომარეობის გამოჩენას, მხოლოდ ამის შემდეგ ის აფორმირებს «გაშვების» მდგომარეობას. ალმის ჩამოგდება სრულდება აპარატურულად მას შემდეგ, როდესაც დასრულდება «გაშვების» მდგომარეობის ფორმირება .
4	TWSTO	**«გაჩერების» მდგომარეობის ალამი.** წამყვანის რეჟიმში TWST0 ალმის ერთიანში დაყენება იწვევს სალტეზე «გაჩერების» მდგომარეობის ფორმირებას. ალამი განულდება აპარატურულად «გაჩერების« მდგომარეობის დასრულების ფორმირების შემდეგ. TWST0 ალმის დაყენება წამყვანის რეჟიმში შეიძლება გამოყენებული იყოს შეცდომის სიტუაციიდან გამოსვლისთვის. მას შემდეგ, როდესაც ამ თანრიგში ჩაიწერება ლოგიკური ერთიანი ,ასეთ სიტუაციაში მოდული უბრუნდება წამყვანის დაუმისამართებელ რეჟიმს, ხოლო SCL და SDA გამომყვანები გადადიან მესამე მდგომარეობაში. «გაჩერების» მდგომარეობა არ ფორმირდება.
3	TWWC	**ჩაწერის კონფლიკტის ალამი.** ალმის დაყენება ერთიანის მდგომარეობაში ხორციელდება იმ შემთხვევაში, როდესაც ჩაწერის მცდელობაა TWDR რეგისტრში და TWINT წყვეტის ალამი განულებულია. ალმის განულება ხორციელდება, როდესაც TWDR რეგისტრში სრულდება ჩაწერა და TWINT წყვეტის ალამი დაყენებულია ერთიანში.
2	TWEN	**TWI მოდულის მუშაობის ნებადართვა.** TWEN თანრიგი ანხორციელებს TWI მოდულის ფუნქციონირების მართვას. ამ თანრიში ლოგიკური ერთიანის ჩაწერის შემთხვევაში TWI მოდული ჩაირთვება და თავისთავზე იღებს SCL და SDA გამომყვანების შესაბამისი მიკროკონტროლერის შეტანა/გამოტანის კონტაქტების მართვას,. TWI მოდული გამოირთვება , როდესაც TWEN ბიტში ჩაიწერება ნული.
1	---	დარეზერვირებულია და ამოიკითხება, როგორც ნული.
0	TWIE	**TWI მოდულისაგან წყვეტის ნებადართვა.**თუ ამ თანრიგში ჩაწერილია ლოგიკური ერთიანი და SREG რეგისტრის I თანრიგი ასევე დაყენებულია ერთიანის მდგომარეობაში, მაშინ TWI მოდულისაგან წყვეტა ნებადართულია.



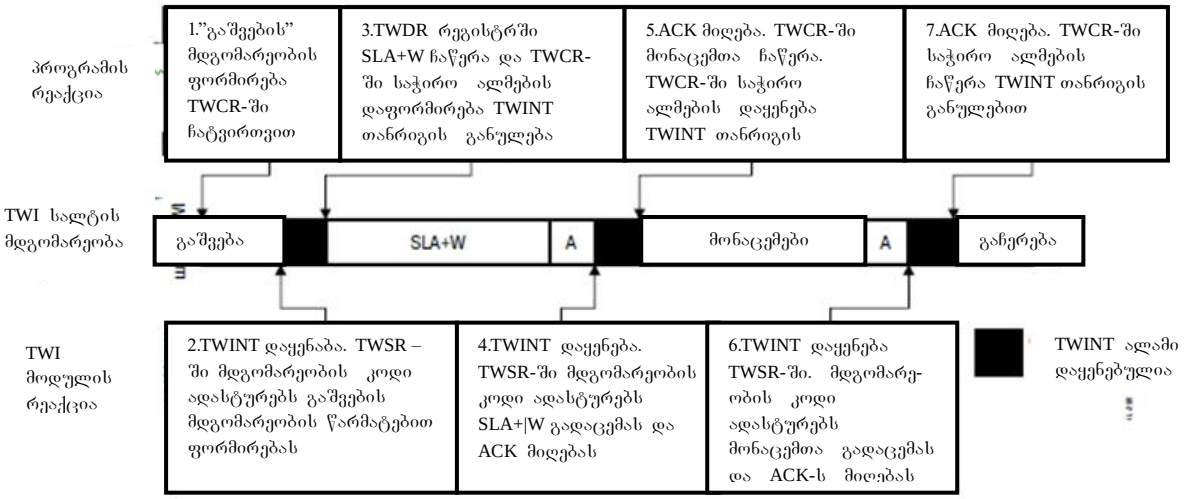
სურ.10.10. TWSR რეგისტრის ფორმატი

ცხრილი 10.5. TWSR მდგომარეობის რეგისტრის თანრიგები

თანრიგი	დასახელება	აღწერა
7 ..3	TWS7…TWS3	**TWI მოდულის მდგომარეობა**. ამ თანრიგებში არსებული მნიშვნელობები ასახავენ მოდულის კვანძების და TWI სალტის მდგომარეობებს რაიმე ოპერაციის შესრულების შემდეგ. მდგომარეობის კოდები განიხილებიან შემდგომში. TWI7…3 თანრიგები დან შესაძლებელია მხოლოდ წაკითხვა.
2	-	დარეზერვირებულია. იკითხება როგორც “0”
1.,0	TWPS1:TWPS0	გადაცემის სიჩქარის კონტროლერის წინაგამყოფის გაყოფის კოეფიციენტი.

10.4. TWI მოდულის და გამოყენებითი პროგრამის ურთიერთქმედება

TWI მოდულისა და სამომხმარებლო პროგრამის ურთიერთქმედება ეფუძნება მოდულიდან წყვეტის გამოყენებას, რომელიც ფორმირდება სალტეზე ყოველი მოვლენის წარმოშობის შემდეგ (ბაიტის მიღება, “გაშვება”/”გაჩერების” მდგომარეობა და ა.შ). რაც შესაძლებლობას იძლევა სალტით მონაცემების გადაცემის დროს პროგრამამ პარალელურად შეასრულოს სხვა ამოცანები. თუ TWI მოდულიდან წყვეტის გამოყენება რაიმე მიზეზის გამო არ არის შესაძლებელი, იგი შეიძლება ავკრძალოთ. ასეთ შემთხვევაში პროგრამამ, მოვლენაზე რეაგირებისთვის, მუდმივად უნდა უთვალთვალოს TWINT ალმის მდგომარეობას, რომელიც წარმოიქმნება სალტეზე. TWINT ალმის დაყენება TWCR რეგისტრში გულისხმობს, რომ TWI მოდულმა დაამთავრა მორიგი ოპერაციის შესრულება და ელოდება პროგრამის რეაქციას. TWSR რეგისტრის ხუთ უფროს თანრიგში (TWS7:0) ფორმირდება რაიმე მნიშვნელობა, რომელიც ახასიათებს TWI სალტის მიმდინარე მდგომარეობას. შესაბამისად პროგრამამ უნდა გაანალიზოს ეს მნიშვნელობა და გასცეს TWI მოდულის შემდგომი ფუნქციონირების გზა, TWCR და TWDR რეგისტერების შემდგომი მანიპულირებით. 10.11.სურ..-ზე მარტივ მაგალითზე ნაჩვენებია TWI მოდულისა და გამოყენებითი პროგრამის ურთიერთქმედება, რომელიც ითვალისწინებს მონაცემთა ერთი ბაიტის გადაცემას წამყვანიდან.



სურ. 10.11. TWI მოდულის და პროგრამის ურთიერთმოქმედება

მონაცემის გადაცემა სრულდება შემდეგი თანამიმდევრობით:

1.TWI სალტეზე მონაცემის გადაცემის დროს პირველი ოპერაცია არის “გაშვების” მდგომარეობის ფორმირება. ამისათვის აუცილებელია TWCR რეგისტრში ჩაიწეროს განსაზღრული მნიშვნელობა, რომლის შესაბამისადაც TWI მოდული დააფორმირებს სალტეზე “გაშვების” მდგომარეობას. ამასთან ერთად TWINT ალამი უნდა დაყენებული იყოს ერთიანის მდგომარეობაში. “გაშვების” მდგომარეობის ფორმირება იწყება TWINT ალმის ჩამოგდებასთან ერთად.

2. “გაშვების” მდგომარეობის ფორმირების დაწყებისთანავე ხდება TWINT ალმის დაყენება. ამ ოპერაციის დამთავრების შედეგად TWSR მდგომარეობის რეგისტრში იწერება კოდი, რომელიც უჩვენებს შესრულებული ოპერაციის შედეგის ხასიათს.

3. უნდა დავრწმუნდეთ, რომ წარმატებით განხორციელდა “გაშვების” მდგომარეობის ფორმირება TWSR რეგისტრის შემცველობის შემოწმებით. თუ მდგომარეობის კოდი შეესაბამება მოსალოდნელს, მაშინ აუცილებელია TWDR რეგისტრში ჩაიტვირთოს SLA+W პაკეტის შემცველობა და დაფორმირდეს TWCR რეგისტრში პაკეტის გადაგზავნის ბრძანება. სამისამართო პაკეტის გადაცემა დაიწყება TWINT ალმის ჩამოგდებისთანავე

4. სამისამართო პაკეტის გადაცემის შემდეგ ხორციელდება TWINT ალმის დაყენება TWCR რეგისტრში. სტატუსის კოდი გვიჩვენებს პაკეტის გადაცემის წარმატებით შესრულებას და წამყვანი მოწყობილობიდან დასტურის მიღებას.

5. უნდა დავრწმუნდეთ, რომ მოხდა პაკეტის წარმატებულად გადაცემა და მიღებულია დასტური, რაც ხორციელდება TWSR რეგისტრის შემცველობის შემოწმებით. თუ მდგომარეობის კოდი შეესაბამება მოსალოდნელს, მაშინ TWDR რეგისტრში უნდა ჩაიტვირთოს მონაცემები. ამის შემდეგ უნდა დაფორმირდეს TWCR რეგისტრში პაკეტის გადაცემის ბრძანება. (ამასთან, შესრულდეს TWINT ალმის ჩამოგდება) ალმის ჩამოგდებისთანავე დაიწყება მონაცემის პაკეტის გადაცემა.

6. მონაცემთა პაკეტის გადაცემის დამთავრების შემდეგ ხდება TWINT ალმის დაყენება. TWCR რეგისტრში დაფორმირებული მდგომარეობის კოდი მიუთითებს პაკეტის წარმატებით გადაცემაზე და მიმყოლი მოწყობილობიდან დასტურის მიღებაზე.

7. TWSR რეგისტრის შემცველობის შემოწმების საფუძველზე დავრწმუნდეთ პაკეტის წარმატებულ გადაცემაში და დასტურის მიღებაზე. თუ მდგომარეობის კოდი შეესაბამება მოსალოდნელს, მაშინ საჭიროა TWCR რეგისტრში ჩაიწეროს მნიშვნელობა (აგრეთვე TWINT ალმის ჩამოგდება), რომლის შესაბამისად TWI მოდული აფორმირებს სალტეზე "გაჩერების" მდგომარეობას. "გაჩერების" მდგომარეობის ფორმირება დაიწყება TWINT ალმის ჩამოგდებისთანავე.

ზემოთ თქმულიდან ჩანს, რომ გამოყენებითი პროგრამისა და TWI მოდულს შორის ურთიერთქმედების ნებისმიერი ეტაპი შედგება სამი ნაწილისაგან:

- ყოველი ოპერაციის დამთავრების შემდეგ მოდული ანხორციელებს TWCR რეგისტრის TWINT ალმის დაყენებას და ელოდება პროგრამის რეაქციას. სანამ ალამი დაყენებულია ერთიანის მდგომარეობაში SCL გამტარზე შენარჩუნებულია დაბალი დონე;
- TWINT ალმის დაყენების შემდეგ მომხმარებელმა მოდულის TWDR და TWCR რეგისტრებში უნდა შეიტანოს მნიშვნელობები, რომლებიც შეესაბამება გაცვლის მომდევნო ეტაპს;
- მოდულის რეგისტრების განახლების შემდეგ, მომხმარებელმა უნდა დააფორმიროს TWCR რეგისტრში ბრძანება მომდევნო გაცვლის ეტაპის შესრულებაზე. რეგისტრში ახალი მნიშვნელობის ჩატვირთვის დროს აუცილებელია მოხდეს TWINT ალმის ჩამოგდება მასში ლოგიკური ერთიანის ჩაწერით. ალმის ჩამოგდების შემდეგ მოდული იწყებს იმ ოპერაციის შესრულებას, რომელიც TWCR რეგისტრის შემცველობითაა განსაზღვრული.

მდგომარეობის კოდების შესაძლო მნიშვნელობები, რომელებიც მითითებულია მაგალითში, მოყვანილი იქნება მოგვიანებით TWI მოდულის კონკრეტული რეჟიმების აღწერის დროს. გარდა ამისა გვაქს ($F8 და $00) ორი მდგომარეობის კოდი, რომელიც კავშირში არ არიან რომელიმე მუშაობის რეჟიმთან. (ცხრილი 10.6.)

მდგომარეობა $F8 კოდი არის შუალედური. ეს კოდი მიუთითებს, რომ არ არსებობს არავითარი ინფორმაცია TWINT ალმის დაყენების შესახებ. $00 სტატუსის კოდი იძლევა შეტყობინებას სალტეზე შეცდომის შესახებ. ასეთი სახის შეცდომა ჩნდება იმ შემთხვევაში, როდესაც წამყვანი აფორმირებს "გაშვების" ან "გაჩერების" მდგომარეობას არა სასურველ ადგილას, მაგალითად, მისამართის ბაიტის, მონაცემთა ბაიტის ან დადასტურების ბიტის გადაცემის დროს. ამ სიტუაციიდან გამოსვლისათვის აუცილებელია მოხდეს ლოგიკური ერთიანის ჩაწერა TWSTO და TWINT თანრიგებში. ამის შედეგად მოდული გადავა "დაუმისამართებელ წამყვანის" რეჟიმში, გაანთავისუფლებს SDA და SCL გამტარებს და ჩამოაგდებს TWSTO ალამს. ამ სიტუაციაში "გაჩერების" მდგომარეობა დაფორმირებული არ იქნება.

10.5 TWI მოდულის მუშაობის რეჟიმები

TWI მოდულს შეუძლია იმუშაოს შემდეგ რეჟიმებში:

- წამყვანი გადამცემი (Master Transmitter);
- წამყვანი მიმღები (Master Receiver);
- მიმყოლი გადამცემი (Slave Transmitter);
- მიმყოლი მიმღები (Slave Receiver).

კონკრეტული რეჟიმის ამორჩევა განისაზღვრება პროგრამის მუშაობის ლოგიკით, და შესაბამისი მოქმედებებით.

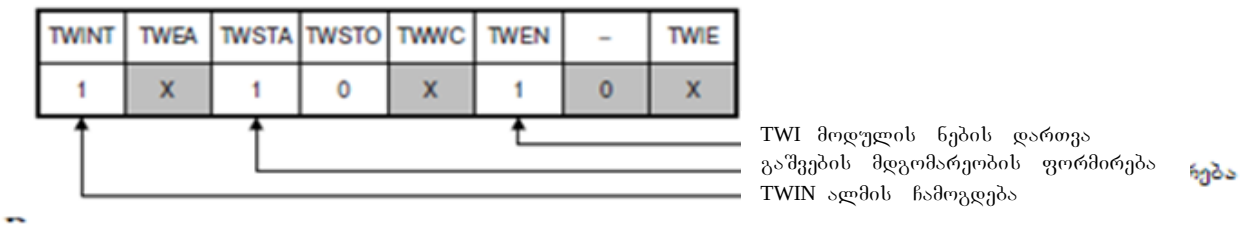
ცრილი 10.6. საერთო დანიშულების სტატუსის კოდები

სტატუსის კოდი	სალტის და TWI მოდულის მდგომარეობა	პროგრამის მოქმედება					TWEმოდულის მიერშემდეგი მოქმედების შესრულება
		TWDR დან/მასში	TWCR რეგისტრში				
			STA	ST0	TWINT	TWEA	
$F8	არარის ინფორმაცია; TWINT= 0	მოქმედე-ბა არ არის	მოქმედება არ არის				TWINTალამის დაყენების მოლოდინი
$00	შეცდომა სალტეზე «გაშვების» ან «გაჩერების» მდგომარეობის არა კორექტუ-ლი ფორმირე-ბის დროს.	მოქმედე ბა არ არის	0	1	1	X	ველა მოქმდება სრულდება აპარატურულად. სალტე თავისუფლდება, TWST0 ჩამოიგდება ნულში.

“წამყვანი გადამცემი” რეჟიმი

“წამყვანი გადამცემი” (Master Transmitter) რეჟიმში ხორციელდება მონაცემების გადაცემა წამყვანი მოწყობილობიდან მიმყოლი მოწყობილობისკენ. იმისათვის, რომ მოხდეს მოწყობილობის “წამყვანი რეჟიმში” გადართვა, TWI მოდულმა უნდა დააფორმიროს სალტეზე “გაშვების” მდგომარეობა. მისამართის პაკეტის ფორმატი, რომელიც გადაიცემა შემდეგ, განსაზღვრავს თუ რომელ რეჟიმში იმუშავებს წამყვანი. იმ შემთხვევაში, როდესაც გადაიცემა SLA+W პაკეტი, მაშინ მოდული გადადის “წამყვანი გადამცემი” რეჟიმში, ხოლო იმ შემთხვევაში, როდესაც გადაიცემა SLA+R პაკეტი, მაშინ მოდული გადადის “წამყვანი მიმღები” რეჟიმში.

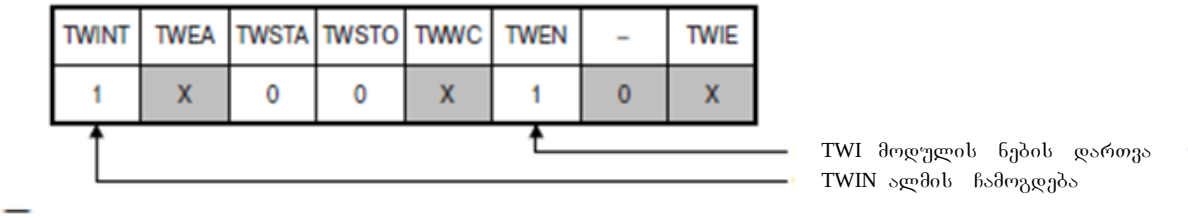
“გაშვების” მდგომარეობის ფორმირება იწყება მას შემდეგ, როდესაც TWCR რეგისტრში ჩაიწერება შემდეგი მნიშვნელობა (სურ.10.12):



სურ.10.12. TWCR რეგისტრში ჩაწერილი “გაშვების” ფორმირების კოდი

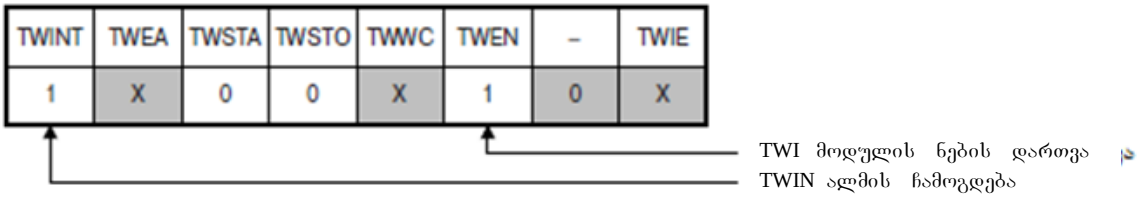
მითითებული მნიშვნელობების ჩაწერის შედეგად TWI მოდული იწყებს სალტის მდგომარეობის შემოწმებას და აფორმირებს “გაშვების” მდგომარეობას , როგორც კი ის

გახდება თავისუფალი. მას შემდეგ როდესაც დასრულდება “გაშვების” მდგომარეობის ფორმატირება, ხორციელდება TWINT ალმის დაყენება. ამ შემთხვევაში მდგომარეობის კოდს უნდა ჰქონდეს $08 მნიშვნელობა (იხილეთ ცხრილი 10.7.). მოდულის “წამყვანი გადამცემი” რეჟიმში გადართვისათვის, აუცილებელია სალტით გადაიცეს SLA+W პაკეტი, პაკეტის შემცველობა იწერება TWDR რეგისტრში, ხოლო TWCR რეგისტრში ჩაიტვირთება შემდეგი კოდი (სურ.10.13).:



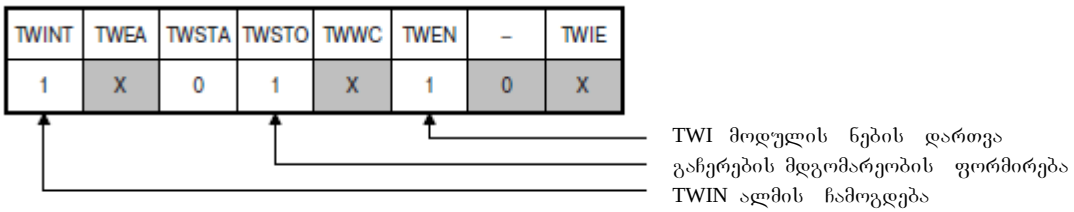
სურ. 10.13. TWCR რეგისტრში ჩაწერილი სამისამართო პაკეტის გაგზავნის კოდი

მისამართის პაკეტის გადაცემის და დასტურის ბიტის მიღების შემდეგ, TWINT ალამი კვლავ დაყენდება “1” მდგომარეობაში. მდგომარეობის კოდს ამ ეტაპზე შეიძლება ჰქონდეს ერთ-ერთი მნიშვნელობა $18, $20 ან $38. 10.7.ცხრილში დაწვრილებით მოცემულია, თუ როგორ უნდა ვიმოქმედოთ ამა თუ იმ კოდის მიღების დროს. მისამართის პაკეტის გადაცემის შემდეგ უნდა მოხდეს მონაცემების პაკეტის გადაცემა. მონაცემების ბაიტის მნიშვნელობა იტვირთება TWDR რეგისტრში. მონაცემების პაკეტის გადაცემა დაიწყება მას შემდეგ, როდესაც TWCR რეგისტრში ჩაიწერება მნიშვნელობა (სურ.10.14):



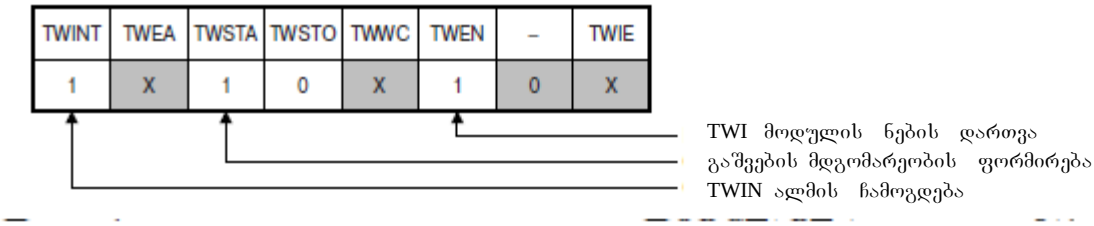
სურ.10.14. TWCR რეგისტრში ჩაწერილი მონაცემთა პაკეტის გადაცემის კოდი

აღწერილი პროცედურა გამოიყენება ყველა მონაცემების პაკეტის გადაცემისთვის. მონაცემების ბოლო ბაიტის გადაცემის შემდეგ წამყვანმა სალტეზე უნდა დააფორმიროს «გაშვების» ან «გაჩერების» მდგომარეობა. “გაჩერების” მდგომარეობის ფორმირება იწყება მას შემდეგ, როდესაც TWCR რეგისტრში ჩაიწერება შემდეგი მნიშვნელობა (სურ.10.15):



სურ.10.15. TWCR რეგისტრში ჩაწერილი
"გაჩერების" ფორმირების კოდი

"განმეორებითი გაშვების" მდგომარეობის ფორმირებისთვის TWCR რეგისტრში საჭიროა ჩაიწეროს შემდეგი მნიშვნელობა (სურ.10.16):



სურ.10.16. TWCR რეგისტრში ჩაწერილი "განმეორებით გაშვების"
ფორმირების კოდი

მას შემდეგ, როდესაც სალტეზე დაფორმირდება "განმეორებითი გაშვება ($10 მდგომარეობის კოდი) წამყვანს შეუძლია დაამისამართოს იგივე ან სხვა მიმყოლები ისე, რომ არ დააფორმიროს "გაჩერების" მდგომარეობა. სხვაგვარად, რომ ვთქვათ "განმეორებით გაშვების" მდგომარეობა საშუალებას იძლევა განახორციელოს მიმყოლი მოწყობილობების შეცვლა, ასევე მოახდინოს გადართვა რეჟიმებს შორის: "წამყვანი გადამცემი" ან " წამყვანი მიმღები", ისე რომ არ დაკარგოს კონტროლი სალტეზე.

როგორც იყო აღნიშნული, მონაცემთა გადაცემის ყოველი ბიჯის დასრულების შემდეგ, ოპერაციის შედეგის მიხედვით, TWSR რეგისტრში ფორმირდება მდგომარეობის კოდი, რომელიც განსაზღვრავს TWI ბლოკის შემდგომ მოქმედებას. 10.7.ცხრილში ნაჩვენებია მდგომარეობის კოდები "წამყვანი გადამცემის" რეჟიმისათვის და თითოეულ კოდთან დაკავშირებული შემდეგი მოქმედება.

ცხრილი 10.7. მდგომარეობის კოდები " წამყვანი გადამცემის" რეჟიმისათვის

მდგომარეობის კოდი	სალტის და TWI მოდულის მდგომარეობა	პროგრამის მდგომარეობა					შემდეგი მოქმედება, რომელსაც ასრულებს TWI მოდული
		TWDR – დან/მასში	TWCR რეგისტრი				
			STA	ST0	TWINT	TWEA	
08	დაფორმირებული იყო «გაშვების» მდგომარეობა	ჩაიტვირთოს SLA +W	X	0	1	X	გადაცემული იქნება SLA+W. მიღებული იქნება ACK ან NACK
10	დაფორმირებული იყო «გან.გაშვების» მდგომარეობა	ჩაიტვირთოს SLA +W	X	0	1	X	გადაცემული იქნება SLA +W. მიღებული იქნება ACK ან NACK
		ჩაიტვირთოს SLA +R	X	0	1	X	გადაცემული იქნება SLA +R. მოდული გადაირთვება «წამ- ყვანი მიმღების» რეჟიმში
18	მოხდა გადაცემა SLA+W პაკეტის და მიღებულია (ACK) დასტური	ჩაიტვირთოს მონაცემები	0	0	1	X	გადაიცემა მონაცემის ბაიტი. მიღებული იქნება ACK ან NACK
		მოქმედება არ არის	1	0	1	X	დაფორმირდება «განმ.გაშვების» მდგომატეობა.
		მოქმედება არ არის	0	1	1	X	დაფორმირდება «გაჩერების» მდგომარეობა (ჩამოიგდება TWST0 ალამი)
		მოქმედება არ არის	1	1	1	X	დაფორმირდება « გაჩერების» მდგომარეობა, ხოლო შემდეგ « გაშვების» მდგომარეობა (ჩამოიგდება TWST0 ალამი)
20	მოხდა SLA+W პაკეტის გადაცემა, მაგრამ არ იყო მიღებული (NACK) დასტური	ჩაიტვირთოს მონაცემები	0	0	1	X	მოხდება მონაცემთა ბაიტის გადაცემა. მიღებული იქნება ACK ან NACK
		მოქმედება არ არის	1	0	1	X	მოხდება გან.გაშვების მდგომარეობის დაფორმირება
		მოქმედება არ არის	0	1	1	X	მოხდება გაჩერების მდგომარეობის დაფორმირება (ჩამოიგდება TWSTO ალამი
		მოქმედება არ არის	1	1	1	X	მოხდება გაჩერების მდგომარეობის დაფორმირება, შემდეგ დაფორმირდება გაშვების მდგომარეობა (მოხდება TWSTO ალამის ჩამოგდება)
28	მოხდება მონაცემის პაკეტის გადაცემა და მიღებულია (ACK) დასტური	მოქმედება არ არის	0	0	1	X	მოხდება მონაცემის ბაიტის გადაცემა. მიღებული იქნება ACK ან NACK
		მოქმედება არ არის	1	0	1	X	დაფორმირდება გან.გაშვების მდგომარეობა
		მოქმედება არ არის	0	1	1	X	დაფორმირდება გაჩერების მდგომარეობა (შესრულდება TWSTO ალამის ჩამოგდება)
		მოქმედება არ არის	!	1	1	X	დაფორმირდება გაშვების მდგომარეობა (მოხდება (TWSTO ალამის ჩამოგდება)

მდგომარეობის კოდი	სალტის და TWI მოდულის მდგომარეობა	პროგრამის მდგომარეობა					შემდეგი მოქმედება, რომელსაც ასრულებს TWI მოდული
		TWDS – დან/მასში	TWCR რეგისტრი				
			STA	STO	TWINT	TWEA	
30	მოხდება მონაცემის პაკეტის გადაცემა და დასტური(NACK) არ იყო მიღებული	მოქმედება არ გვაქს	0	0	1	X	მოხდება მონაცემის ბაიტის გადაცემა. იქნება მიღებული ACK ან NACK
		მოქმედება არ გვაქს	1	0	1	X	დაფორმირდება გან.გაშვების მდგომარეობა
		მოქმედება არ გვაქს	0	1	1	X	დაფორმირდება გაჩერების მდგომარეობა(შესრულდება TWSTO ალამის ჩამოგდება)
		მოქმედება არ გვაქს	1	1	1	X	დაფორმირდება გაჩერების მდგომარეობა, ხოლო შემდეგ გაშვების მდგომარეობა (შესრულდება TWSTO ალამის ჩამოგდება)
38	პრიორიტეტის დაკარგვა მონაცემის ან მისამართის პაკეტის გადაცემის დროს	მოქმედება არ გვაქს	0	0	1	X	მოწყობილობა გაანთავისუფლებს სალტეს და გადავა არა დამისამართებული მიმყოლის რეჟიმში
		მოქმედება არ გვაქს	1	0	1		სალტის განთავისუფლების შემდეგ დაფორმირდება გაშვების მდგომარეობა

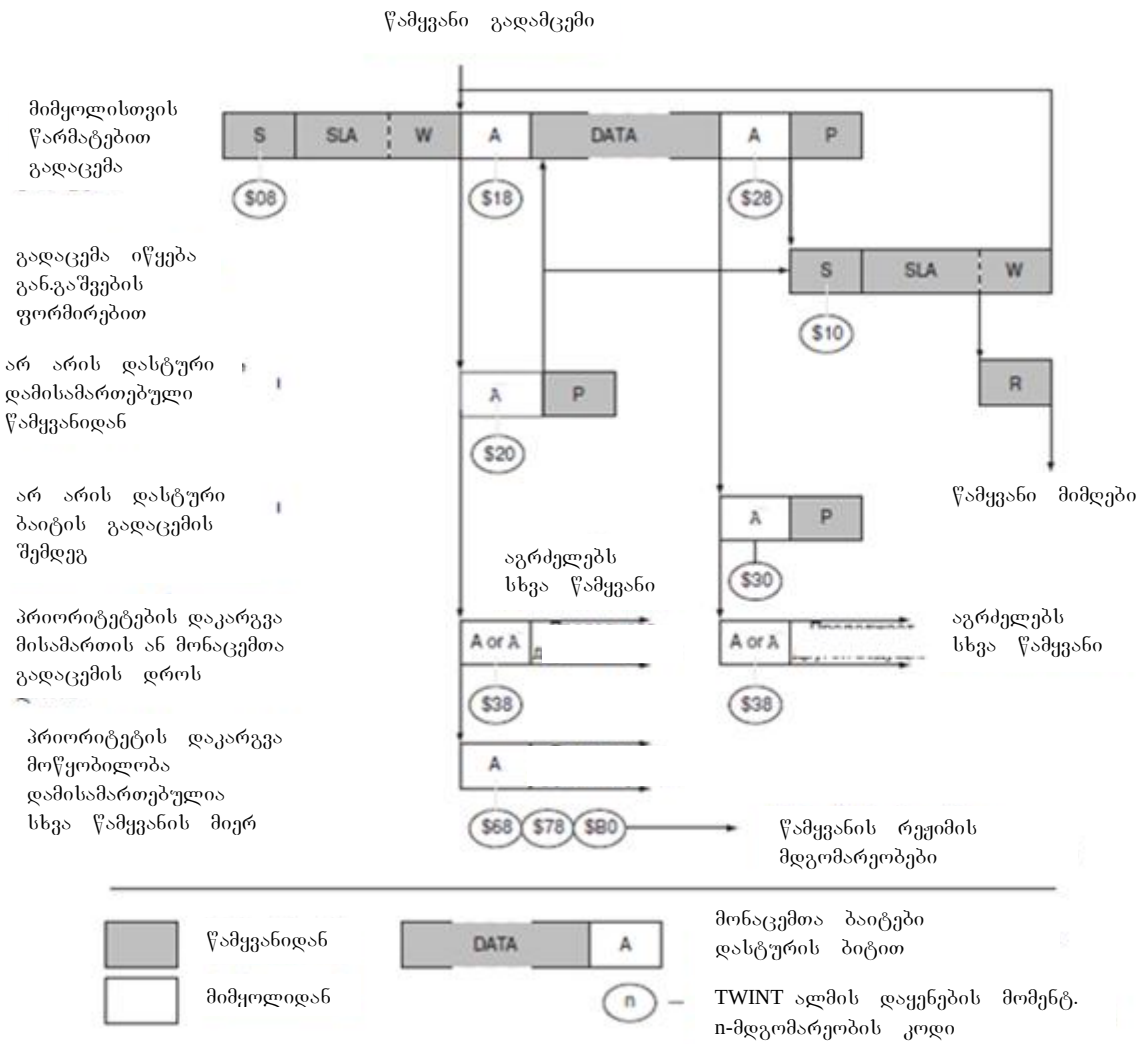
.

10.17.სურ.-ზე მოცემულია TWI ინტერფეისის “წამყვან გადამცემი” რეჟიმში მუშაობის დიაგრამა.

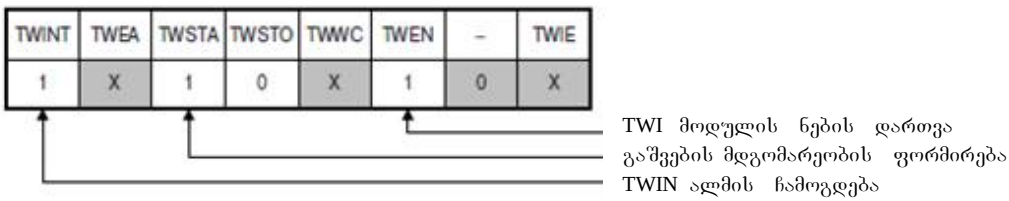
“წამყვანი მიმღები” რეჟიმი

“წამყვანი მიმღები” რეჟიმში წამყვანი ასრულებს მონაცემთა მიღებას მიმყოლი მოწყობილობიდან. როგორც იყო ნათქვამი მოწყობილობის გადართვისათვის წამყვანის რეჟიმში TWI მოდულმა სალტეზე უნდა დააფორმიროს “გაშვების” მდგომარეობა. შემდეგ ბიჯზე გააზავნის სამისამართო პაკეტს SAL+R, რითაც განისაზღვრება მისი მუშაობა “წამყვანი მიმღების” რეჟიმში.

“გაშვების” მდგომარეობის ფორმირება იწყება TWCR რეგისტრში შემდეგი მნიშვნელობის ჩაწერით (სურ.10.18):



სურ.10.17. TWI ინტერფეისის "წამყვან გადამცემი" რეჟიმში მუშაობის დიაგრამა

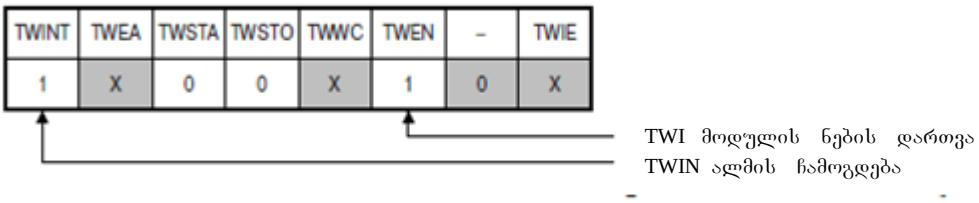


სურ. 10.18. TWCR რეგისტრში ჩაწერილი "გაშვების" ფორმირების კოდი

აღნიშნული მნიშვნელობის ჩაწერის შემდეგ TWI მოდული იწყებს სალტის მდგომარეობის შემოწმებას და როგორც კი იგი განთავისუფლდება აფორმირებს "გაშვების" მდგომარეობას.

"გაშვების" მდგომარეობის დამთავრების შემდეგ დგება TWINT ალამი; მდგომარეობის კოდს ამ დროს უნდა ჰქონდეს $08 მნიშვნელობა (ცხრილი 10.8). "წამყვანი მიმღები"-ს რეჟიმში მოდულის გადართვისათვის საჭიროა სალტით

გადაიცეს სამისამართო პაკეტი SLA+R. ამისათვის პაკეტი უნდა ჩაიწეროს TWDR რეგისტრში, ხოლო TWCR რეგისტრში ჩაიწერება შემდეგი მნიშვნელობა (სურ.10.19):

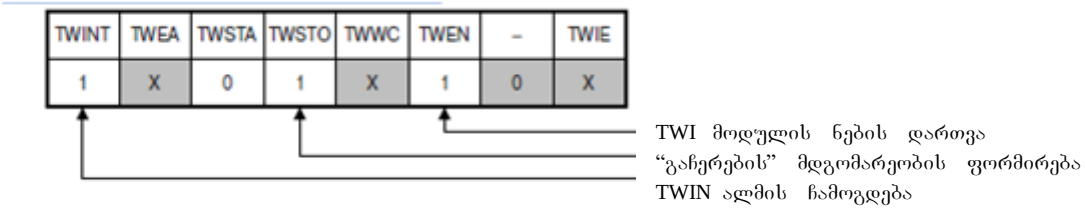


სურ.10.19. TWCR რეგისტრში ჩაწერილი სამისამართო პაკეტის გაგზავნის კოდი

სამისამართო პაკეტის გადაცემისა და დადასტურების ბიტის მიღების შემდეგ TWINT ალამი ისევ დაეება 1-ში. მიმყოლიდან მიღებული მონაცემთა ბაიტი იმყოფება TWDR რეგისტრში. მდგომარეობის კოდს ამ ეტაპზე შესაძლებელია ჰქონდეს ერთ-ერთი მნიშვნელობა: $38, $40 ან $48. რა მოქმედებები უნდა შესრულდეს ამა თუ იმ კოდის აღმოჩენის შემთხვევაში განხილულია 10.8.ცხრილში.

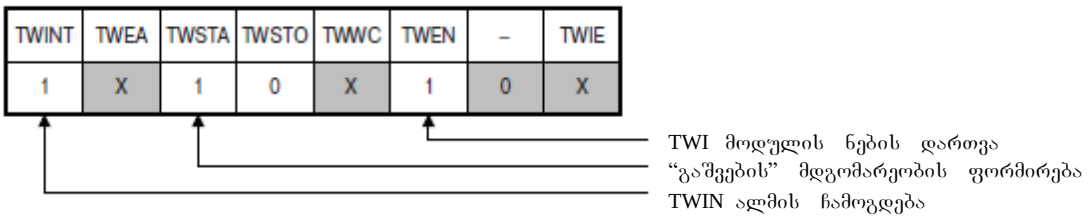
აღწერილი პროცედურა გამოიყენება მონაცემთა ყველა პაკეტების გადაცემისთვის. მონაცემთა ბოლო ბაიტის მიღების შემდეგ წამყვანმა უნდა შეატყობინოს ამის შესახებ მიმყოლ გადამცემს "არდადასტურების" სიგნალით (NACK). ამის შემდეგ წამყვანმა სალტეზე უნდა დააფორმიროს "გაჩერების" ან "გან.გაშვების" მდგომარეობა.

"გაჩერების" მდგომარეობის ფორმირება დაიწყება TWCR რეგისტრში შემდეგი მნიშვნელობის ჩაწერით (სურათი 10.20):



სურ.10.20. TWCR რეგისტრში ჩაწერილი "გაჩერების" ფორმირების კოდი

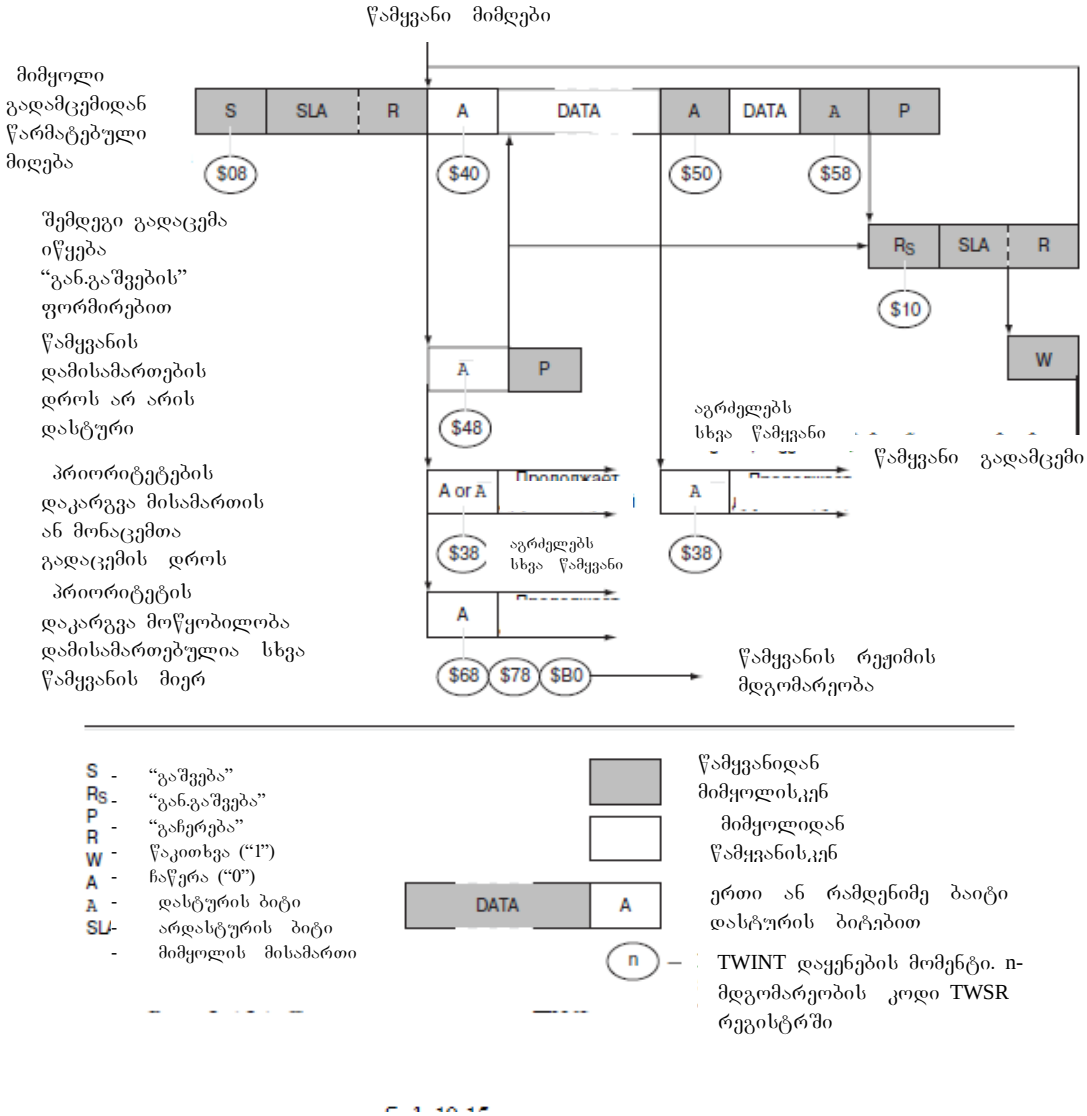
"გან.გაშვების" მდგომარეობის ფორმირებისათვის საჭიროა TWCR რეგისტრში ჩაიწეროს შემდეგი მნიშვნელობა (სურათი 10.21):



სურ.10.21. TWCR რეგისტრში ჩაწერილი "გან.გაშვების" ფორმირების კოდი

როგორც ითქვა, სალტეზე “გან.გაშვების” მდგომარეობის (კოდი $10) დაფორმირების შემდეგ წამყვანს შეუძლია დაამისამართოს იგივე ან სხვა მიმყოლი, “გაშვების” სიგნალის ფორმირების გარეშე. სხვა სიტყვებით რომ ვთქვათ, “გან.გაშვების” მდგომარე- ობის გამოყენება საშუალებას იძლევა განხორციელდეს მიმყოლი მოწყობილობის შეცვლა, აგრეთვე “წამყვანი გადამცემისა” და “წამყვან მიმღები” რეჟიმებს შორის გადართვა სალტეზე კონტროლის დაკარგვის გარეშე.

10.22.სურ.-ზე მოცემულია TWI ინტერფეისის მუშაობის დიაგრამა “წამყვანი მიმღების” რეჟიმში.



სურ.10.22. TWI ინტერფეისის “წამყვან მიმღების” რეჟიმში მუშაობის დიაგრამა

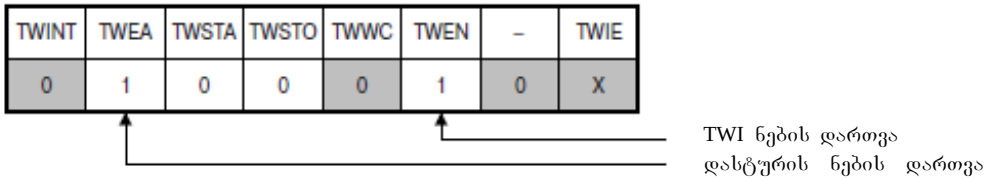
ცხრილი 10.8. მდგომარეობის კოდები “წამყვანი მიმღები” რეჟიმისათვის

მდგომარეობის კოდი	სალტისა და TWI მოდულის მდგომარეობა	პროგრამის მოქმედება					TWI მოდულის მიერ შესრულებული მომდევნო მოქმედება
		TWDR ში/დან	TWCR რეგისტრში				
			STA	STO	TWINT	TWEA	
08	დაფორმირებულია მდგომარეობა “გაშვება”	ჩაიტვირთოს SLA+R	X	0	1	X	გადაცემული იქნება SLA+R მიღებული იქნება ACK ან NACK
10	დაფორმირდა მდგომარეობა “გან.გაშვება”	ჩაიტვირთოს SLA+R	X	0	1	X	გადაცემული იქნება SLA+R მიღებული იქნება ACK ან NACK
		ჩაიტვირთოს SLA+W	X	0	1	X	გადაცემული იქნება SLA+W მოდული გადაირთვება “წამყვანი მიმღები” რეჟიმში
38	პრიორიტეტის დაკარგვა მისამართის ან მონაცემთა პაკეტის გადაცემის დროს	არ არის მოქმედება	0	0	1	X	მოწყობილობა ანთავისუფლებს სალტეს დ გადავა არადამისა მართებად წამყვანის რეჟიმში
		არ არის მოქმედება	1	0	0	X	სალტის განთავისუფლების შემდეგ დაფორმირდება “გაშვების” მდგომარეობა
40	გადაცემული იყო პაკეტი SLA+R და მიღებულია დასტური (ACK)	არ არის მოქმედება	0	0	1	0	მიღებული იქნება მონაცემების ბაიტი დ გადაიცემა დასტურის უარყოფა (NACK)
		არ არის მოქმედება	0	0	1	1	მიღებული იქნება მონაცემების ბაიტი დ გადაიცემა დასტური (ACK)
48	გადაცემული იყო პაკეტი SLA+R და მიღებულია დასტურის უარყოფა (NACK)	არ არის მოქმედება	1	0	1	X	დაფორმირდება მდგომარეობა “გან.გაშვება”
		არ არის მოქმედება	0	1	1	X	დაფორმირდება მდგომარეობა “გაჩერება” (TWSTO ალამი ჩამოვარდება)
		არ არის მოქმედება	1	1	1	X	დაფორმირდება მდგომარეობა “გაჩერება”, შ “გაშვება” (TWSTO ალამი ჩამოვარდება)
50	მიღებულია მონაცემების ბაიტი და გადაცემულია დასტური (ACK)	წაკითხული იყვნენ მონაცემები	0	0	1	0	მიღებული იქნება მონაცემთა ბაიტი და გადაიცემა დასტურის უარყოფა (NACK)
		წაკითხული იყვნენ მონაცემები	0	0	1	1	მიღებული იქნება მონაცემთა ბაიტი და გადაიცემა დასტური (ACK)
58	მიღებულია მონაცემთა ბაიტი და გადაცემულია დასტურის უარყოფა (NACK)	წაკითხული იყვნენ მონაცემები	1	0	1	X	დაფორმირდება მდგომარეობა “გან.გაშვება”
		წაკითხული იყვნენ მონაცემები	0	1	1	X	დაფორმირდება მდგომარეობა “გაჩერება” (TWSTO ალამი ჩამოვარდება)
		წაკითხული იყვნენ მონაცემები	1	1	1	X	დაფორმირდება მდგომარეობა “გაჩერება”, შემდეგ “გაშვება” (TWSTO ალამი ჩამოვარ

“მიმყოლი მიმღები” რეჟიმი

”მიმყოლი მიმღები” რეჟიმში მიმყოლი მოწყობილობა ასრულებს მონაცემთა მიღებას წამყვანისაგან. ”მიმყოლი მიმღები” რეჟიმში მოდულის გადართვის შემთხვევაში TWAR რეგისტრის უფროს თანრიგებში წინასწარ უნდა ჩაიწეროს მოწყობილობის მისამართი და პროგრამის მუშაობის ლოგიკის შესაბამისად

დავაყენოთ ან ჩამოვაგდოთ ამ რეგისტრის უმცროსი თანრიგი (TWGCE). ამის შემდეგ TWCR რეგისტრში საჭიროა ჩაიწეროს შემდეგი მნიშვნელობა (სურ.10.23):



სურ.10.23. მიმყოლის TWCR რეგისტრში ჩაწერილი ინიციალიზაციის კოდი

TWAR და TWCR რეგისტრების ინიციალიზაციის შემდეგ მოდული ელოდება სამისამართო პაკეტს TWAR რეგისტრში მითითებული მისამართით ან საერთო გამოძახებას (თუ მისი ამოცნობა ნებადართულია). სამისამართო პაკეტის უმცროს თანრიგში ჩაწერილი მიმართულების ბიტის მნიშვნელობა განსაზღვრავს რეჟიმს, რომელშიც გადავა მოდული. თუ ამ თანრიგში არის “0” (W-ჩაწერა), TWI მოდული გადაირთვება “მიმყოლი მიმღები” რეჟიმში. წინააღმდეგ შემთხვევაში მოდული გადაირთვება “მიმყოლი გადამცემი” რეჟიმში. გავიხსენოთ, რომ მოწყობილობას შეუძლია ავტომატურად გადავიდეს “მიმყოლი მიმღები” რეჟიმში წამყვანის რეჟიმიდან პრიორიტეტის დაკარგვის შემთხვევაში. SLA+W პაკეტის მიღების შემდეგ დგება TWINT ალამი და სალტით გადაცემის ხასიათი, როგორც წინა შემთხვევებში, განისაზღვრება მდგომარეობის კოდით. შესაძლო მოქმედებები პროგრამის მხრიდან, მდგომარეობის კოდის შესაბამისად, ნაჩვენებია 10.9.ცხრილ- ში.

მონაცემთა ნაკადის შეწყვეტისათვის საჭიროა TWCR რეგისტრის TWEA თანრიგი ჩამოვაგდოთ “0”-ში. შედეგად SDA სალტეზე მომდევნო ბაიტის გადაცემის შემთხვევაში გაიცემა არდადასტურების სიგნალი, რომელიც ატყობინებს წამყვანს იმის შესახებ, რომ მიმყოლს აღარ შეუძლია მონაცემთა მიღება. სამისამართე პაკეტების დამუშავება ჩამოგდებული TWEA თანრიგის დროს წყდება, მაგრამ შეიძლება განახლდეს ნებისმიერ დროს ამ თანრიგის ხელახალი დაყენებით.

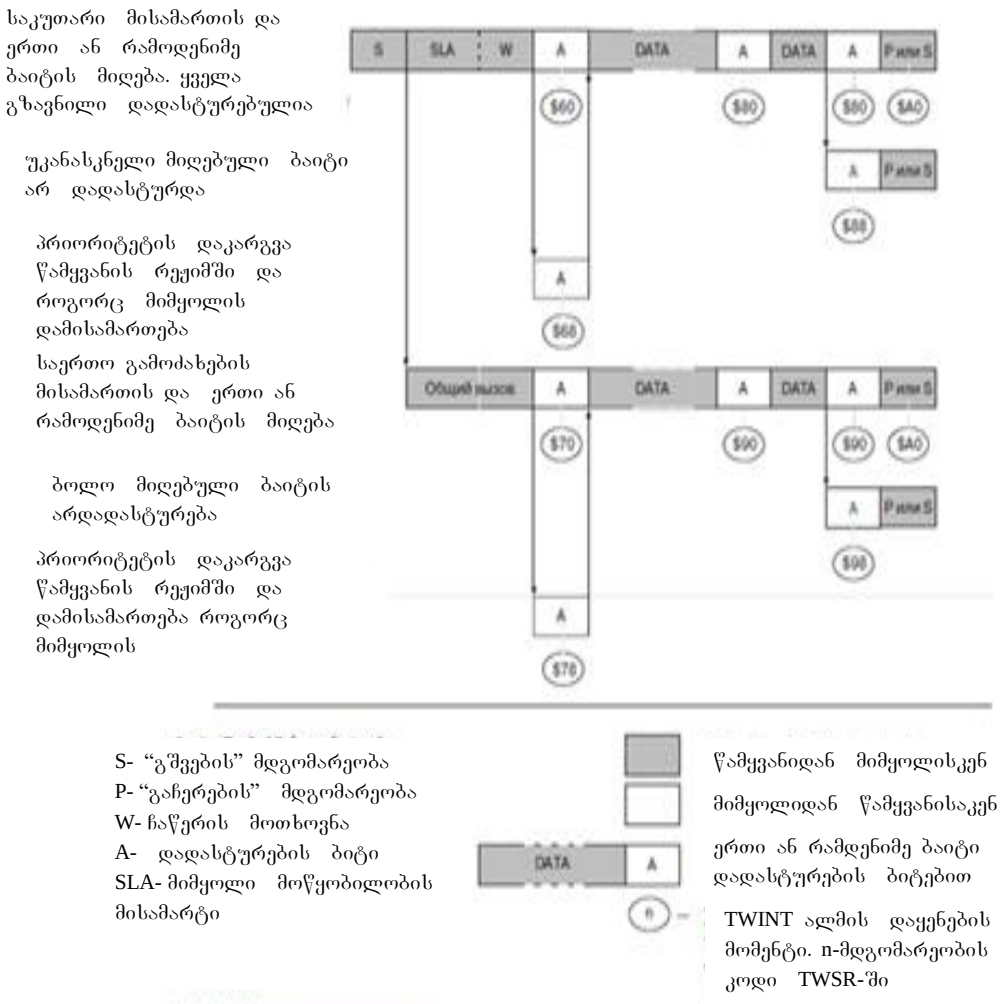
თუ მოწყობილობის დამისამართება და ამასთან TWEA თანრიგის დაყენება მოხდება მიკროკონტროლერის “ძილის” რეჟიმში ყოფნის დროს, მიკროკონტროლერი გადადის მუშა რეჟიმში.

ცხრილი 10.9. მდგომარეობის კოდები “ მიმყოლი მიმღები” რეჟიმისათვის

მდგომარეობის კოდი	TWI სალტის და მოდეულის მდგომარეობა	პროგრამის მოქმედება					TWI მოდულის მიერ შესასრულებელი შემდგომი მოქმედებები
		WDR ში/დან	TWCR რეგისტრში				
			STA	STO	TWINT	TWEA	
$60	მიღებული იყო SLA+W საკუთარი მისამართით და გაიგზავნა დადასტურება (ACK)	არ არის მოქმედება	X	0	1	0	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება არადასტური (NACK)
		არ არის მოქმედება	X	0	1	1	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება დასტური (ACK)
$68	პრიორიტეტის დაკარგვა წამყვა - ნის რეჟიმში SLA+R/W გაგზავნის დროს, მიღებული იქნა SLA+W საკუთარი მისამარ- თით და გადაიცა დადასტურება (ACK)	არ არის მოქმედება	X	0	1	0	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება არადასტური (NACK)
		არ არის მოქმედება	X	0	1	1	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება დასტური (ACK)
$70	მიღებული იქნა საერთო შეტყობინება და გაიგზავნა დადასტურება(ACK)	არ არის მოქმედება	X	0	1	0	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება არადასტური (NACK)
		არ არის მოქმედება	X	0	1	1	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება დასტური (ACK)
$78	პრიორიტეტის დაკარგვა წამყვანის რეჟიმში SLA+R/W გაგზავნის დროს, მივიღეთ საერთო შეტყობინება და გაიგზავნა დადასტურება (ACK)	არ არის მოქმედება	X	0	1	0	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება არადასტური (NACK)
$80	მოწყობილობასთან იყო მიმართვა; მიღებულია მონაცემთა ბაიტი და გადაიცა დადასტურება (ACK)	წაკითხუ- ლი იყოს მონაცემე ბი	X	0	1	0	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება არადასტური (NACK
		წაკითხუ- ლი იყოს მონაცემე ბი	X	0	1	1	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება დადასტურება (ACK)

მდგომარეობის კოდი	TWI სალტის და მოდეულის მდგომარეობა	პროგრამის მოქმედება					TWI მოდულის მიერ შესასრულებელი შემდგომი მოქმედებები
		TWDR ში/დან	TWCR რეგისტრში				
			STA	STO	TWINT	TWEA	
$88	მოწყობილობასთან იყო მიმართვა; მიღებულია მონაცემთა ბაიტი და გადაიცა არდადასტურება (NACK)	წაკითხული იყოს მონაცემები	0	0	1	0	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია
		წაკითხული იყოს მონაცემები	0	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართიანი SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1
		წაკითხული იყოს მონაცემები	1	0	1	0	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა.
		წაკითხული იყოს მონაცემები	1	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადარტულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა.
$90	მოწყობილობასთან იყო მიმართვა (საერთო გამოძახება); მიღებულია მონაცემთა ბაიტი და გადაიცა დადასტურება (ACK)	წაკითხული იყოს მონაცემები	X	0	1	0	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება არადასტური (NACK)
		წაკითხული იყოს მონაცემები	X	0	1	1	მიღებული იქნება მონაცემთა ბაიტი და გადაცემული იქნება დადასტურება (ACK)
$98	იყო მიმართვა (საერთო გამოძახება); მიღებულია მონაცემთა ბაიტი და გადაიცა არადადასტურება (NACK)	წაკითხული იყოს მონაცემები	0	0	1	0	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია
		წაკითხული იყოს მონაცემები	0	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1..
		წაკითხული იყოს მონაცემები	1	0	1	0	გადაირთოს მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა.
		წაკითხული იყოს მონაცემები	1	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1.. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა
$A0	აღმოჩენილი იყო “გაშვება “ ან “გან.გაშვება”, როდესაც მოწყობილობასთან იყო მიმართვა, როგორც მიმყოლთან	წაკითხული იყოს მონაცემები	0	0	1	0	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია
		წაკითხული იყოს მონაცემები	0	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1..
		წაკითხული იყოს მონაცემები	1	0	1	0	მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა

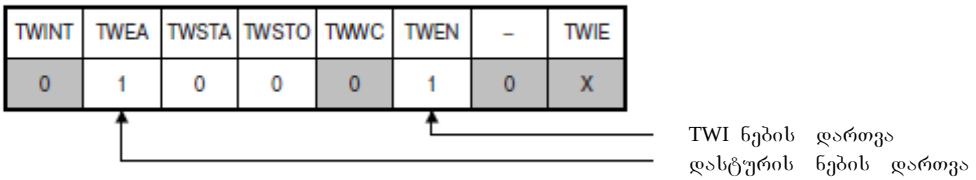
10.24.სურ.-ზე ნაჩვენებია TWI ინტერფეისის “მიმყოლი მიმღები” რეჟიმში მუშაობის დიაგრამა ზემოთ განხილულ შემთხვევებისთვის.



სურ.10.24. TWI ინტერფეისის “მიმყოლი მიმღები” რეჟიმში მუშაობის დიაგრამა.

“მიმყოლი გადამცემი” რეჟიმი

“მიმყოლი გადამცემი” რეჟიმში მიმყოლი მოწყობილობა ანხორციელებს მონაცემების გადაცემას წამყვანისაკენ, რომელიც ამ კონკრეტულ შემთხვევაში არის მიმღები. მოდულის ამ რეჟიმში გადართვამდე, საჭიროა შევიტანოთ TWAR რეგისტრის უფროს თანრიგებში მოწყობილობის მისამართი და პროგრამის მუშაობის ლოგიკის შესაბამისად, მოხდეს რეგისტრის უმცროსი თანრიგის (TWGCE) 1-ში ან 0-ში დაყენება. შემდეგ საჭიროა TWCR რეგისტრში ჩაიწეროს მნიშვნელობა (სურ.10.25): TWAR და TWCR რეგისტრების ინიციალიზაციის შემდეგ მოდული ელოდება მისამართის პაკეტს, რომელშიც იქნება TWAR რეგისტრში მითითებული მისამართი ან საერთო



სურ.10.25. მიმყოლის TWCR რეგისტრში ჩაწერილი ინიციალიზაციის კოდი

გამოძახება (თუ კი მისი ამოცნობა ნებადართულია). პაკეტში მისამართის შემდეგ არსებული ბიტის მნიშვნელობა განსაზღვრავს იმ რეჟიმს, რომელშიც გადაირთვება მოდული

მოდული. თუ R/W ბიტი შეიცავს “0” (ჩაწერა), TWI მოდული გადაირთვება “მიმყოლი მიმღების” რეჟიმში. წინააღმდეგ შემთხვევაში მოდული გადაირთვება “ მიმყოლი გადამცემის” რეჟიმში. გავიხსენოთ, რომ მოწყობილობას ასევე შეუძლია ავტომატურად გადაირთოს წამყვანი რეჟიმიდან “მიმყოლი მიმღების” რეჟიმში პრიორიტეტის დაკარგვის შემთხვევაში (იხილეთ $B0 მდგომარეობის კოდის აღწერა ცხრილ10.10.-ში).

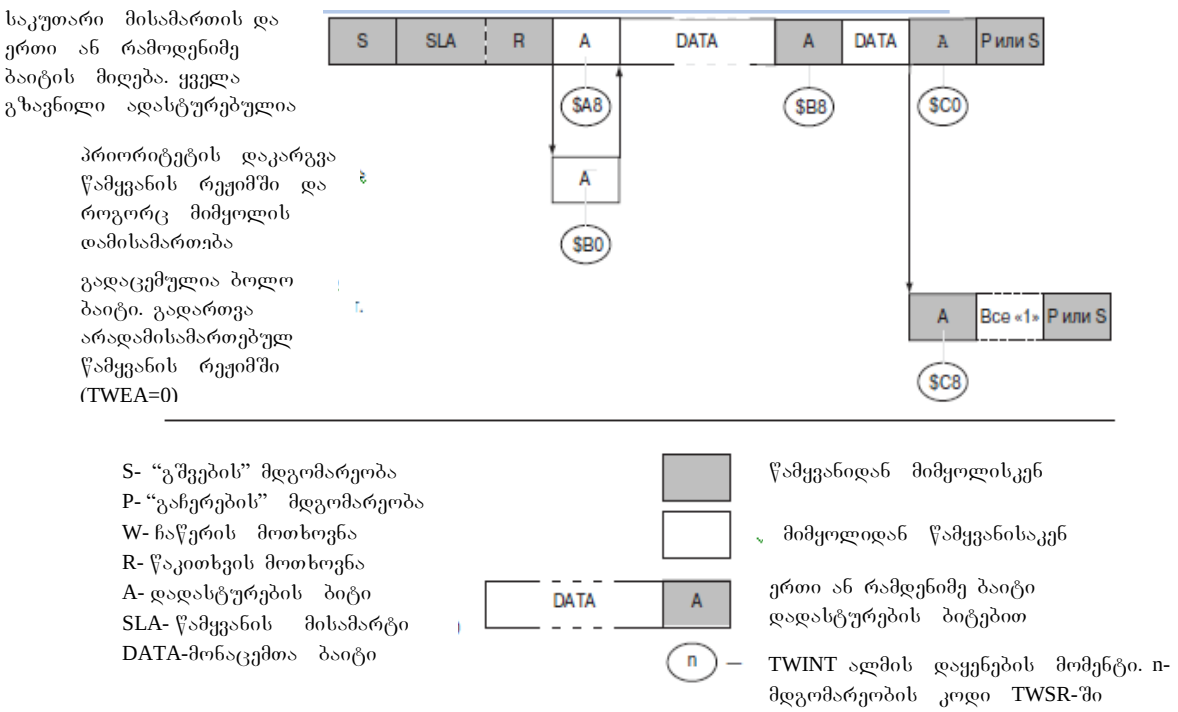
SLA+R პაკეტის მიღების შემდეგ დგება TWINT ალამი, და გადაცემის მდგომარეობა განისაზღვრება მდგომარეობის კოდით (ცხრ.10.10)

ცხრილი 10.10. მდგომარეობის კოდები “ მიმყოლი გადამცემი” რეჟიმისთვის

მდგომარეობის კოდი	TWI სალტის და მოდეულის მდგომარეობა	პროგრამის მოქმედება					TWI მოდულის მიერ შესასრულებელი შემდგომი მოქმედებები
		TWDR ში/და	TWCR რეგისტრში				
			STA	STO	TWINT	TWEA	
$C0	გადაცემული იყო მონაცემთა ბაიტი და მიღებული იყო არდადასტურება (NACK)	არ არის მოქმედება	0	0	1	0	გადაირთოს არადამისამართებული მიმყო ლის რეჟიმში ნებისმიერი გამოძახების გამოცნობა აკრძალულია.
		არ არის მოქმედება	0	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1.
		არ არის მოქმედება	1	0	1	0	გამოძახების გამოცნობა აკრძალულია. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა.
		არ არის მოქმედება	1	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1.. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა

მდგომარეობის კოდი	TWI სალტის და მოდეულის მდგომარეობა	პროგრამის მოქმედება					TWI მოდულის მიერ შესასრულებელი შემდგომი მოქმედებები
		TWDR ში/და	TWCR რეგისტრში				
			STA	STO	TWINT	TWEA	
$C8	გადაცემული იყო მონაცემთა ბაიტი და მიღებული იყო დადასტურება (ACK)	არ არის მოქმედება	0	0	1	0	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია.
		არ არის მოქმედება	0	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1..
		არ არის მოქმედება	1	0	1	0	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებისმიერი გამოძახების გამოცნობა აკრძალულია. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა
		არ არის მოქმედება	1	0	1	1	გადაირთოს არადამისამართებული მიმყოლის რეჟიმში. ნებადართულია საკუთარი მისამართის მქონე SLA გამოცნობა. ნებადართულია საერთო გამოძახების გამოცნობა, თუ TWGCE=1. სალტის განთავისუფლების შემდეგ დაფორმირდება “გაჩერების” მდგომარეობა
$A8	მიღებული იყო SLA+W საკუთარ მისამართთან ერთად და გაიგზავნა დადასტურება (ACK)	ჩაიტვირთოს მონაცემი	X	0	1	0	გადაცემული იქნება მონაცემთა ბოლო ბაიტი და მიღებული უნდა იყოს არდადასტურება (NACK)
		ჩაიტვირთოს მონაცემი	X	0	1	1	გადაცემული იქნება მონაცემთა მომდევნო ბაიტი. მიღებული უნდა იყოს დადასტურება (ACK)
$B0	წამყვანის რეჟიმში პრიორიტეტის დაკარგვა SLA+W გადაცემის დროს. მიღებული იყო SLA+W საკუთარ მისამართთან ერთად და გაიგზავნა დადასტურება (ACK)	ჩაიტვირთოს მონაცემი	X	0	1	0	გადაცემული იქნება მონაცემთა ბოლო ბაიტი და მიღებული უნდა იყოს არდადასტურება (NACK)
		ჩაიტვირთოს მონაცემი	X	0	1	1	გადაცემული იქნება მონაცემთა მომდევნო ბაიტი. მიღებული უნდა იყოს დადასტურება (ACK)
$B8	გადაცემული იყო მონაცემთა ბაიტი და მიღებული იყო დადასტურება (ACK)	ჩაიტვირთოს მონაცემი	X	0	1	0	გადაცემული იქნება მონაცემთა ბოლო ბაიტი და მიღებული უნდა იყოს არდადასტურება (NACK)
		ჩაიტვირთოს მონაცემი	X	0	1	1	გადაცემული იქნება მონაცემთა მომდევნო ბაიტი. მიღებული უნდა იყოს დადასტურება (ACK

მონაცემთა ბოლო ბაიტის გადაცემის დროს აუცილებელია TWEA თანრიგი ჩამოვაგდოთ ნულში. ამის შემდეგ, მოდული გადადის **$C0** ან **$C8** კოდის შესაბამის მდგომარეობაში, იმისდა მიხედვით, თუ რომელ (ACK ან NACK) სიგნალს გადასცემს წამყვანი პასუხად. **$C8** კოდის შესაბამის მდგომარეობაში TWI მოდული გადადის იმ შემთხვევაში, თუ წამყვანმა მოითხოვა დამატებითი მონაცემები, გადასცა (ACK) დადასტურება, მიუხედავად იმისა რომ წამყვანმა გადამცემმა გადასცა ბოლო ბაიტი და ელოდება NACK სიგნალს. 10.26.სურ-ზე ნაჩვენებია TWI ინტერფეისის “მიმყოლ გადამცემ” რეჯიმში მუშაობის დიაგრამა.



სურ.10.26 TWI ინტერფეისის "მიმყოლი გადამცემი" რეჟიმში მუშაობის დიაგრამა

სხვადასხვა რეჟიმის კომბინირება

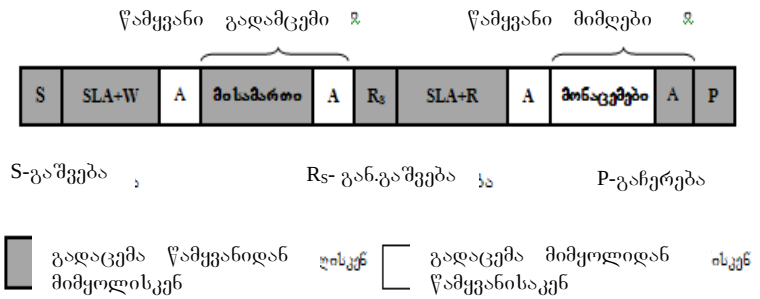
პრაქტიკაში რაიმე ოპერაციის შესრულებისათვის TWI სალტით, თვითეულ მოწყობილობას უხდება რამდენიმე რეჟიმის გამოყენება, ერთმანეთს შორის გადართვის აუცილებლობის შემთხვევაში. მაგალითად, განვიხილოთ მონაცემების წაკითხვის ოპერაცია გარე EEPROM-დან.

ასეთი სახის ყველა ოპერაცია შეიძლება დავყოთ ოთხ ეტაპად:

1) გაცვლის ინიცირება;
2) მისამართის გადაცემა, რომლის მიხედვითაც ხორციელდება მონაცემის წაკითხვა;
3) წაკითხვის შესრულება;
4) გადაცემის დასრულება.

ნათქვამიდან ჩანს, რომ ოპერაციის შესრულების დროს ხორციელდება ინფორმაციის გადაცემა როგორც წამყვანიდან მიმყოლისაკენ, ასევე პირიქით. გაცვლის შესრულების დროს წამყვანი უნდა იმყოფებოდეს " წამყვანი გადამცემი" რეჟიმში, რომ შეატყობინოს მიმყოლს მისამართი, რომლიდანაც აპირებს განახორციელოს წაკითხვა.

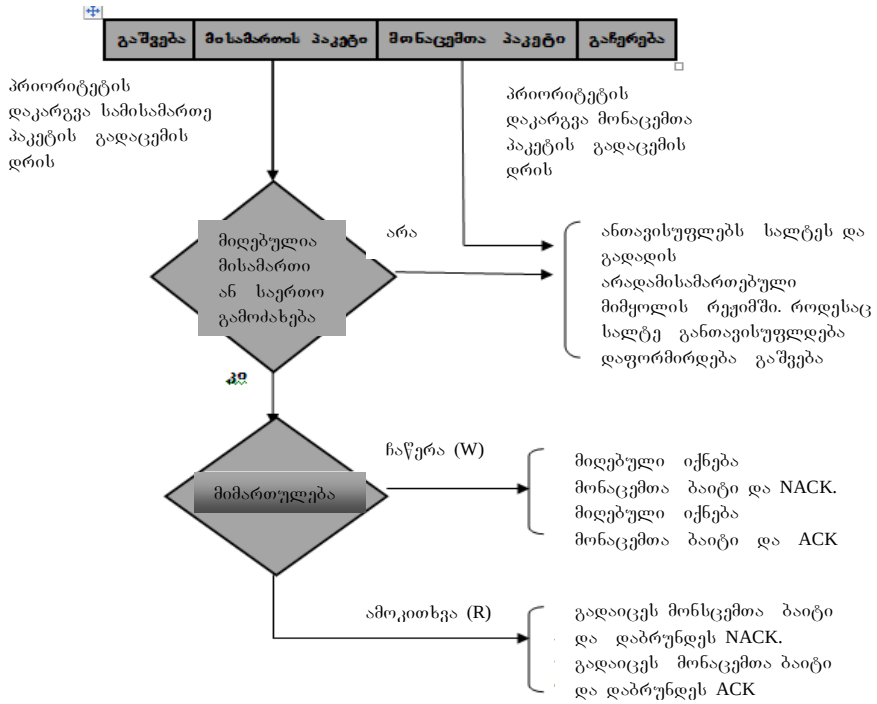
ოპერაციის შემდგომი ეტაპის (მონაცემების წაკითხვა) შესრულებისთვის წამყვანი უნდა გადაერთოს "წამყვანი მიმღების" რეჟიმში. ამასთან, უნდა შეინარჩუნოს კონტროლი სალტეზე ოპერაციის ყველა ეტაპების შესრულების განმავლობაში. ამისთვის, გამოიყენება "გან.გაშვების" მდგომარეობა. განხილულ შემთხვევაში წამყვანი ამ მდგომარეობას აფორმირებს მისამართის გადაცემისა და მონაცემის მიღებას შორის, რომელიც ნაჩვენებია 10.27.სურ.ზე.



სურ. 10.27. პაკეტების გადაცემის მიმდევრობა რეჟიმების ცვლის შემთხვევაში

არბიტრაჟი

იმ შემთხვევაში, როდესაც სალტეზე რამდენიმე წამყვანია შესაძლე- ბელია წარმოიქმნას სიტუაცია, რომლის დროსაც რამოდენიმე წამყვანი ერთდროულად დაიწყებს გაცვლის პროცესს. ასეთ შემთხვევაში TWI სპეციფიკა- ციით წინასწარ განსაზღრულია პროცესის პრიორიტეტების განაწილება (არბიტრაჟი), რომლის შესრულების შედეგად სალტეზე რჩება მხოლოდ ერთი წამყვანი მოწყობილობა. ამ პროცესის პრინციპები განხილული იყო ზევით, მაგრამ აქვე უნდა აღინშნოს რომ, მისი შესრულება შეიძლება განვითარდეს სხვადასხვა სცენარით (სურ.10.28):



სურ.10.28. მრავალ აბონენტიან TWI ინტერფეისებს შორის მონაცემთა გაცვლის არბიტრაჟის ალგორითმი

1. ერთი ან მეტი წამყვანი ანხორციელებს ერთი ტიპის გაცვლას ერთი და იმავე მიმყოლთან. ასეთ შემთხვევაში ვერცერთი მადგანი ვერ შესძლებს ამოიცნოს კონფლიქტური სიტუაცია.

2. ორი ან მეტი წამყვანი ანხორციელებს მიმართვას ერთი და იმავე მიმყოლთან სხვადასხვა მონაცემებით ან განსხვავებული გაცვლის (წაკითხვა/ჩაწერის) ტიპით. ასეთ შემთხვევაში პრიორიტეტების განაწილება განხორციელდება მონაცემთა ბიტის ან მიმართულების ბიტის გადაცემის დროს. წამყვანი, რომელმაც დაკარგა პრიორიტეტი, შეიძლება გადაერთოს დაუმისამართებელი მიმყოლის რეჟიმში ან დაელოდოს სალტის განთავისუფლებას და მოახდინოს "გაშვების" მდგომარეობის ფორმირება მისი დაპყრობისათვის.

3. ორი ან მეტი წამყვანი მიმართავს სხვადასხვა მიმყოლს აღნიშნულ შემთხვევაში პრიორიტეტების განაწილება იწყება მისამართის პაკეტის ბიტების გადაცემის დროს. პრიორიტეტ დაკარგული წამყვანი კი გადაირთვება მიმყოლის რეჟიმში, რომ შეამოწმოს იყო თუ არა მასთან მიმართვა წამყვანის მიერ. თუ ყოფილი წამყვანი აღმოჩნდა დამისამართებული წამყვანის მიერ, მაშინ ის გადაერთვება "მიმყოლი გადამცემის" ან "მიმყოლი მიმღების" რეჟიმში, რომელიც განისაზღვრება მიმართულების ბიტის მნიშვნელობით, ან დაელოდება სალტის განთავისუფლებას და დააფორმირებს "გაშვების" ახალ მდგომარეობას.

თავი 11

შესავალი მიკროკონტროლერების პროგრამირებაში

11.1. საერთო ცნობები

მიკროკონტროლერების დაპროგრამირებაში იგულისხმება ფიზიკური პროცესი, რომლის საშუალებით მიკროპროცესორის მეხსიერებასა და სხვადასხვა რეგისტრში იწერება მისი ფუნქციონირებისთვის საჭირო პროგრამა და მონაცემები, წინასწარ გადასახული ორობით სისტემაში. დაპროგრამირება სრულდება სხვადასხვა ინტერფეისების საშუალებით, სპეციალური ბრძანებე- ბის გამოყენებით.

მომდევნო თავებში განიხილება დაპროგრამების შესრულება მიკრო- კონტროლერ Atmega 128-ის მაგალითზე.

AVR ოჯახის მიკროკონტროლერები მხარს უჭერენ დაპროგრამირების შემდეგ რეჟიმებს:

- მიმდევრობითი დაპროგრამება მაღალი ძაბვის შემთხვევაში;
- მიმდევრობით დაპროგრამება (SPI ინტერფეისით) დაბალი ძაბვის შემთხვევაში;
- პარალელური დაპროგრამება მაღალი ძაბვის შემთხვევაში;
- პროგრამირება JTAG ინტერფეისით

ჩამოთვლილი დაპროგრამირების რეჟიმებიდან მიკროკონტროლერი Atmega 128 მხარს უჭერს ბოლო სამს.

გარდა ამისა, მიკროკონტროლერ Atmega 128-ს გააჩნია თვითპროგრამირების შესაძლებლობა. აღნიშნული ტერმინის ქვეშ მოიაზრება პროგრამის მეხსიერების შემცველობის ცვლილება, თვით მიკროკონტროლერის მიერ.

დაპროგრამირების პროცესის დროს შესაძლებელია შესრულდეს შემდეგი ოპერაციები:

- კრისტალის შემცველობის წაშლა(Chip erase);
- FLASH პროგრამის მეხსიერებაში წაკითხვა/ჩაწერა;
- EEPROM მონაცემის მეხსიერებაში წაკითხვა/ჩაწერა;
- საკონფიგურაციო უჯრედების წაკითხვა/ჩაწერა;
- დაცვის უჯრედების წაკითხვა/ჩაწერა
- იდენტიფიკატორის უჯრედების წაკითხვა;
- მაკალიბრებელი ბაიტის წაკითხვა.

მიკროკონტროლერის ყველა მოდელებში თავდაპირველად პროგრამული და მონაცემების მეხსიერება სუფთაა (ყველა უჯრედებში ჩაწერილია რიცხვი «$FF») და შესაძლებელია მისი დაპროგრამირება.

11.2 პროგრამების და მონაცემების დაცვა

FLASH-მეხსიერების (პროგრამის მეხსიერება) შემცველობა, და ასევე EEPROM (მონაცემის მეხსიერება) შემცველობა შესაძლებელია დაცული იყოს ჩაწერის და/ან წაკითხვისაგან დამცველი LB1 და LB2 (Lock Bits) ბიტების დაპროგრამირების მეშვეობით. ამ

ბიტების მდგომარეობის შესაბამისი შესაძლო დაცული რეჟიმების ვარიანტები მოცემულია 11.1. ცხრილში.

ცხრილი 11.1. დაცვის რეჟიმები

დაცვის ბიტები			აღწერა
რეჟიმის №	LB1	LB2	
1	1	1	მონაცემთა დაცვის კოდი გათიშულია
2	0	1	მომდევნო ჩაწერა FLASH და EEPROM აკრძალულია
3	0	0	აკრძალულია ჩაწერა და წაკითხვა FLASH და EEPROM

მე-2 და მე-3 რეჟიმებში ასევე იკრძალება კონფიგურაციული უჯრედების ცვლილება (იხილეთ ქვევით). ამის გამო დაცვის ჩართვა საჭიროა შესრულდეს, მიკროკონტროლერის მეხსიერების დანარჩენი არის პროგრამირების შემდეგ.

Mega მიკროკონტროლერის ოჯახის წარმომადგენლებს აქვთ ოთხი დამატებითი BLB02, BLB01, BLB12 და BLB11 დაცვის ბიტი. BLB02:BLB01 უჯრედები განსაზღვრავს შეღწევადობის დონეს ჩამტვირთავის სექციიდან გამოყენებით პროგრამის სექციაში, ხოლო BLB12:BLB11 უჯრედები – პირიქით, შეღწევადობის დონეს გამოყენებითი პროგრამის სექციიდან ჩამტვირთავის სექციაში.

ყველა ზევით ჩამოთვლილი დაცვის ბიტები მოთავსებულია ერთ ბაიტში. მასში დამცველი ბიტების განთავსება ნაჩვენებია 11.1.სურათზე.

7	6	5	4	3	2	1	0
-	-	BLB12	BLB11	BLB01	BLB00	LB2	LB1
საწყისი მნიშვნელობა 1	1	1	1	1	1	1	1

სურ.11.1. დაცვის ბაიტის უჯრედები

საწყის (არადაპროგრამირებულ) მდგომარეობაში დაცვის თითოეულ უჯრედში ჩაწერილია ლოგიკური “1”, ხოლო დაპროგრამირების შემდეგ – “0”. უჯრედის წაშლა (მასში ლოგიკური 1 ჩაწერა) შეიძლება შესრულდეს “კრისტალის წაშლის” ბრძანების შესრულების დროს, რომელიც შლის აგრეთვე FLASH და EEPROM მეხსიერების შემცველობას.

შესაძლო დაცვის რეჟიმები, რომლებიც შეესაბამება ამ უჯრედების სხვადასხვა მდგომარეობას, ნაჩვენებია 11.2. და 11.3. ცხრილებში.

11.3. მაკონფიგურირებელი უჯრედები

როგორც დასახელებიდან ჩანს, კონფიგურაციული უჯრედები (Fuse Bits) განსაზღვრავს მიკროკონტროლერის კონფიგურაციის ზოგიერთ პარამეტრებს. კონფიგურაციული უჯრედები განთავსებულია ცალკე სამისამართო სივრცეში, რომლებიც მისაწვდომია მხოლოდ პროგრამირების დროს. ყველა მაკონფიგურირებელი

ცხრილი 11.2. გამოყენებითი პროგრამის სექციის დაცვის რეჟიმები

დაცვის ბიტები			აღწერა
რეჟიმის №	BLB02	BLB01	
1	1	1	არ არის არავითარი შეზღუდვა კოდთან მიმართვისთვის, რომელიც განთავსებულია გამოყენებითი პროგრამის სექციაში
2	1	0	SPM ბრძანებას არ შეუძლია შეასრულოს ჩაწერა იმ მისამართებით, რომლებიც განთავსებულია გამოყენებითი პროგრამის არეში
3	0	0	SPM ბრძანებას არ შეუძლია შეასრულოს ჩაწერა იმ მისამართებით, რომლებიც განთავსებულია გამოყენებითი პროგრამის სექციის ფარგლებში და LPM (ELPM) ბრძანებას, რომლის გამოძახება ხორციელდება ჩამტვირთავის სექციიდან, არ შეუძლია განახორციელოს წაკითხვა გამოყენებითი პროგრამის სექციიდან. თუ წყვეტის ვექტორის ცხრილი განთავსებულია ჩამტვირთავის სექციაში, მაშინ წყვეტა აკრძალულია იმ შემთხვევაში, როდესაც მისი გამოძახება ხორციელდება გამოყენებითი პროგრამის სექციიდან.
4	0	1	LPM (ELPM) ბრძანებას, რომლის გამოძახება ხორციელდება ჩამტვირთავის სექციიდან, არ შეუძლია განახორციელოს წაკითხვა გამოყენებითი პროგრამის სექციიდან, თუ წყვეტის ვექტორის ცხრილი განთავსებულია ჩამტვირთავის სექციაში, წყვეტა აკრძალულია იმ შემთხვევაში თუ მათი გამოძახება ხორციელდება გამოყენებითი პროგრამის სექციიდან.

ცხრილი 11.3. ჩამტვირთავი სექციის დაცვის რეჟიმები

დაცვის ბიტები			აღწერა
რეჟიმის №№	BLB12	BLB 11	
1	1	1	არ არის არავითარი შეზღუდვა ჩამტვირთავის სექციაში განთავსებულ კოდთან წვდომაზე
2	1	0	SPM ბრძანებას არ შეუძლია განახორციელოს ჩაწერა იმ მისამართებზე, რომლებიც განთავსებულნი არიან ჩამტვირთავის სექციის ფარგლებში
3	0	0	SPM ბრძანებას არ შეუძლია განახორციელოს ჩაწერა იმ მისამართებზე, რომლებიც განთავსებულია ჩამტვირთავის სექციის ფარგლებში და LPM (ELPM) ბრძანებას, რომლის გამოძახება ხორციელდება გამოყენებითი პროგრამიდან არ შეუძლია განახორციელოს წაკითხვა ჩამტვირთავის სექციიდან. თუ წყვეტის ვექტორის ცხრილი განთავსებულია გამოყენებითი პროგრამის სექციაში, წყვეტა აკრძალულია იმ შემთხვევაში, როდესაც მათი გამოძახება ხორციელდება ჩამტვირთავის სექციიდან.
4	0	1	LPM (ELPM) ბრძანებას, რომლის გამოძახება ხდება გამოყენებითი პროგრამის სექციიდან, არ შეუძლია განახორციელოს წაკითხვა ჩამტვირთავის სექციიდან. თუ წყვეტის ვექტორის ცხრილი განთავსებულია გამოყენებითი პროგრამის სექციაში, წყვეტა აკრძალულია იმ შემთხვევაში, როდესაც მათი გამოძახება ხორციელდება ჩამტვირთავის სექციიდან.

უჯრედები დაჯგუფებულია რამდენიმე ბაიტად (ერთიდან სამამდე, რომელსაც განსაზღვრავს მოდელი), ხოლო ამ უჯრედების შემადგენლობა დამოკიდებულია მიკროკონტროლერის კონკრეტულ მოდელზე.

მიკროკონტროლერებში ამა თუ იმ ბიტის დანიშნულება განისაზღვრება 11.4.ცხრილით, სადაც მითითებულია ცალკეული ბიტის შემოკლებული დასახელება. “ვარსკლავით” აღნიშნულ სვეტში მოცემულია უთქმელობით საკონფიგურაციო უჯრედების მდგომარეობა.

საკონფიგურაციო უჯრედების შემცველობის შეცვლისათვის გამოიყენება სპეციალური დაპროგრამების ბრძანებები. ბრძანება “ კრისტალის წაშლა” ამ უჯრედების მდგომარეობაზე ზემოქმედებას ვერ ახდენს. შეგახსენებთ რომ LB1,LB0 დაცვის დაპროგრამებული ბიტის შემთხვევაში ხდება კონფიგურაციული ბიტების ბლოკირება. აქედან გამომდინარე მიკროკონტროლერის კონფიგურაცია აუცილებელია განვახორციელოთ დაცვის უჯრედების დაპროგრამებამდე.

ცხრილი 11.4. Atmega 128 მიკროკონტროლერის საკონფიგურაციო უჯრედები

თანრიგი	დასასელება	*	დანიშნულება
უმცროსი საკონფიგურაციო ბაიტი			
7	BODLEVEL	1	განსაზღვრავს BOD სქემის გაშვების ზღვარს
6	BODEN	1	ნებას რთავს/ კრძალავს BOD სქემის ფუნქციონირებას : (0- ნებადართულია, 1- აკრძალულია)
5	SUT1	1	განსაზღვრავს *t*TOUT განულების დაყოვნების ხანგრძლივობას
4	SUT0	0	
3	CKSEL3	0	განსაზღვრავენ სატაქტო გენერატორის მუშაობის რეჟიმს, და ასევე *t*TOUT განულების დაყოვნების ხანგრძლივობას.
2	CKSEL2	0	
1	CKSEL1	0	
0	CKSEL0	1	
უფროსი კონფიგურაციული ბაიტი			
7	OCDEN	1	ნებას რთავს/კრძალვს შიგასქემურ გაწყობას (0- ნებადართულია, 1- აკრძალულია)
6	JTAGEN	0	ნებას რთავს /კრძალავს JTAG ინტერფეისის გამოყენებას(0- ნებადართულია, 1- აკრძალულია)
5	SPIEN	0	ნებას რთვს/კრძალავს პროგრამირებას SPI-ით (0- ნებადართულია, 1- აკრძალულია)
4	CKPOT	1	განსაზღვრავს სატაქტო გენერატორის ფუნქციონირებას, მოქმედება დამოკიდებულია CKSEL უჯრედის დაყენებაზე
3	EESAVE	1	განსაზღვრავს “კრისტალის წაშლის” ბრძანების ზემოქმედებას EEPROM მეხსიერებაზე (0- არ შლის, 1-შლის)
2	BOOTSZ1	0	განსაზღვრავენ ჩამტვირთავის სექციის ზომას
1	BOOTSZ0	0	
0	BOOTRST	1	განსაზღვრავს წყვეტის ვექტორის მდგომარეობას
დამატებითი კონფიგურაციული ბაიტი			
1-7	-	1	-
0	WDTON	1	განსაზღვრავს მოდარაჯე ტაიმერის მუშაობის რეჟიმს (0 - ყოველთვის ჩართულია, 1 - შეიძლება იყოს გამორთული პროგრამულად)

11.4. იდენტიფიკატორი

მიკროკონტროლერ Atmega 128-ს გააჩნია სამი რვა თანრიგიანი უჯრედი, რომელთა შემცველობაც იძლევა საშუალებას მოხდეს მოწყობილობის იდენტიფიცირება. როგორც კონფიგურაციის, ასევე იდენტიფიკაციის უჯრედები განთავსებულია ცალკე სამისამართო სივრცეში, სადაც წვდომა შესაძლებელია მხოლოდ პროგრამირების რეჟიმში. მაგრამ იდენტიფიკატორის უჯრედები, განსხვავებით საკონფიგურაციო უჯრედებისაგან, შეღწევადია მხოლოდ წაკითხვის დროს. იდენტიფიკატორის შემცველობა ნაჩვენებია 11.5.ცხრილში

ცხრილი 11.5. იდენტიფიკატორის უჯრედები

მისამართი	კოდი	აღწერა
$00	$1E	მწარმოებლის კოდი(«Atmel»)
$01	$97	FLASH მეხსიერების* მოცულობის კოდი
$02	$02	მოწყობილობის კოდი
* $97 — 128 კბაიტი		

11.5. მაკალიბრებელი უჯრედი

დამზადების დროს მაკალიბრებელ უჯრედში შეაქვთ მაკალიბრებელი კონსტანტა, რომელიც განკუთნილია შიგა RC გენერატორის ნომინალური სიხშირის რეგულირებისათვის. Atmega 128-ს აქვს ოთხი 8 თანრიგიანი უჯრედი. რომლებიც განთავსებულია იდენტიფიკატორის უჯრედების უფროსი ბაიტების სამისამართო სივრცეში ($000, $001, $002 და $003 მისამართებზე 1, 2, 4, 8 მგჰც შესაბამის სიხშირეებისათვის).

მაკალიბრებელი კონსტანტების შეტანა ხორციელდება აპარატურულად OSCCAL რეგისტრში, როდესაც მიკროკონტროლერი არის განულებული. Atmega 128 გენერატორის ავტომატური დაკალიბრება ხორციელდება მხოლოდ 1 მგჰც სიხშირეზე. იმ შემთხვევაში, როდესაც გამოიყენება RC გენერატორის მუშაობის სხვა რეჟიმი, მაშინ მისი დაკალიბრება სრულდება ხელით: პროგრამატორმა უნდა წაიკითხოს მაკალიბრებელი უჯრედის შემცველობა და შეიტანოს FLASH- პროგრამული მეხსიერების რომელიმე მისამართზე. პროგრამამ უნდა წაიკითხოს მისი მნიშვნელობა პროგრამული მეხსიერებიდან და ჩაწეროს OSCCAL რეგისტრში.

11.6. Atmega 128 მიკროკონტროლერის პროგრამის და მონაცემთა მეხსიერების ორგანიზაცია

მიკროკონტროლერ Atmega 128-ში გამოიყენება პროგრამის მეხსიერების გვერდუ- ლი ორგანიზაცია. დაპროგრამების დროს FLASH - მეხსიერების მთლიანი მოცულობა დაყოფილია ე.წ. “გვერდებად”, რომელთა ზომა და რაოდენობა დამოკიდებულია მიკროკონტროლერის კონკრეტულ მოდელზე. Atmega 128 მიკროკონტროლერისათვის მათი მნიშვნელობები ნაჩვენებია 11.6.ცხრილში.

ცხრილი 11.6. Atmega 128 მიკროპროკონტროლერის პროგრამის მეხსიერების გვერდული ორგანიზაციის პარამეტრები

პარამეტრი	მოცულობა და რაოდენობა
პროგრამის მეხსიერების ზომა (16 თანრიგიანი სიტყვა)	64კ
გვერდზე სიტყვების რაოდენობა	128
გვერდების რაოდენობა	512

მიკროკონტროლერის პროგრამის მეხსიერების პროგრამირების დროს, მონაცემები წინასწარ ჩაიტვირთება გვერდის ბუფერში და ამის შემდეგ უშუალოდ მოხდება მისი შეტანა პროგრამის მეხსიერებაში. გვერდის ყველა უჯრედის "გაკერვა" ხორციელდება ერთდროულად.

ცხრილი 11.7. Atmega 128 მიკროპროცესორის EEPROM-მეხსიერების გვერდული ორგანიზაციის პარამეტრები

პარამეტრი	მოცულობა და რაოდენობა
EEPROM მეხსიერების ზომა[ბაიტი]	4კ
გვერდის ზომა [ბაიტი]	8
გვერდების რაოდენობა	512

საჭიროა აღინიშნოს, რომ EEPROM მეხსიერების გვერდული ორგანიზაცია გამოიყენება მხოლოდ პარალელურ რეჟიმში პროგრამირების დროს. ამ მეხსიერების პროგრამირება ხორციელდება მიმდევრობითი არხით ბაიტ-ბაიტ.

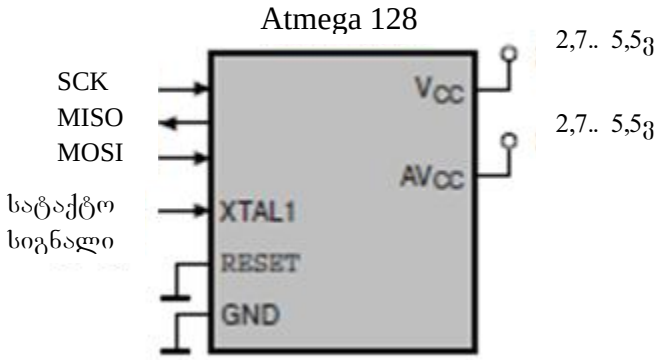
თავი 12

დაპროგრამება მიმდევრობითი არხით

12.1. საერთო ცნობები

ამ რეჟიმში პროგრამის და მონაცემების მეხსიერების პროგრამირება ხორციელდება SPI ინტერფეისით. როგორც წესი, ეს რეჟიმი გამოიყენება მიკროკონტროლერებში უშუალოდ მოწყობილობის პროგრამირების (გადაპროგრამირების) დროს.

მიკროკონტროლერის ჩართვის სქემა პროგრამირების რეჟიმში მიმდევრობითი არხის გამოყენებით ნაჩვენებია 12.1. სურათზე.



შენიშვნა: თუ სატაქტო იმპულსებისათვის გამოიყენება შიგა RC გენერატორი, XTAL1 გამომყვანი რჩება მიერთების გარეშე

სურ.12.1. მიკროკონტროლერის ჩართვა მიმდევრობითი არხით

როგორც სურათიდან ჩანს, პროგრამატორის მიერთებისათვის მიკროკონტროლერთან გამოიყენება იტერფეისის სამი გამტარი: SCK (სატაქტო იმპულსი), MOSI (მონაცემის შეტანა), MISO (მონაცემის გამოტანა). შესაბამისობა ინტერფეისის გამტარებსა და მიკროკონტროლერის შეტანა/გამომტანის პორტების კონტაქტებს შორის ნაჩვენებია 12.1. ცხრილში.

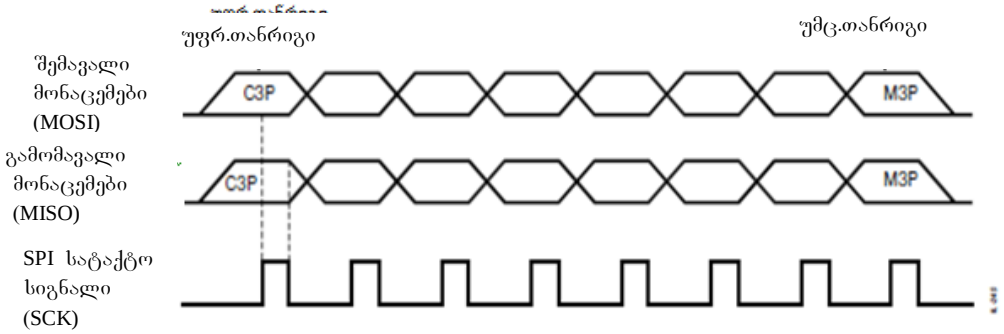
ცხრილი 12.1. გამომყვანები, რომლებიც გამოიყენება მიმდევრობით არხით პროგრამირების დროს

ინტერფეისის გამომყვანების დასახელება		გამომყვანების დასახელებები
SCK	PB1	სატაქტო იმპულსის შესასვლელი
MISO (PDO)	PE1	მონაცემთა გამოსასვლელი
MOSI (PDI)	PE0	მონაცემთა შესასვლელი

ყურადსაღებია ის რომ, გამომყვანები, რომლებიც გამოიყენება პროგრამირების დროს, არ ემთხვევა გამომყვანებს, რომლებიც განკუთვნილია SPI ინტერფეისის საშტატო რეჟიმში მუშაობის დროს.

როგორც მუშა რეჟიმში, ასევე მიმდევრობითი არხით პროგრამირებისათვის შეიძლება გამოვიყენოთ მიკროკონტროლერისთვის დასაშვები ნებისმიერი იმპულსების წყარო, ამავე დროს საჭიროა შესრულდეს შემდეგი პირობა: SCK იმპულსის ხანგრძლივობა როგორც მაღალი, ასევე დაბალი დონისათვის 2-ჯერ ($f_{CK} < 12$ მჰც) ან 3-ჯერ ($f_{CK} \geq 12$ მჰც) მეტი უნდა იყოს მიკროკონტროლერის სატაქტო სიგნალების პერიოდზე.

პროგრამირება ხორციელდება მიკროკონტროლერის MOSI გამომყვანზე 4 ბაიტიანი ბრძანების გადაგზავნით. წაკითხვის ბრძანების შესრულების შედეგი მიიღება მიკროკონტროლერის MISO გამოსასვლელიდან. ბრძანების გადაცემა და შედეგის გაცემა სრულდება უფროსი თანრიგიდან უმცროსისაკენ. ამასთან, შემავალი მონაცემების შეტანა ხორციელდება SCK სიგნალის აღმავალი ფრონტით, ხოლო გამომავალი მონაცემების შეტანა - დაღმავალი ფრონტით. (სურ.12.2)



სურ.12.2. მონაცემების გადაცემა მიმდევრობით არხით პროგრამირების დროს

12.2. პროგრამირების რეჟიმში გადართვა

მიკროკონტროლერის გადართვა მიმდევრობითი არხით პროგრამირების რეჟიმში შემდეგი თანამიმდევრობით სრულდება:

1. მიკროკონტროლერს მიეწოდება კვების ძაბვა, ამასთან SCK და RESET გამოსასვლელებზე უნდა გვქონდეს დაბალი დონის ძაბვა. ზოგიერთ შემთხვევაში (თუ პროგრამატორი არ იძლევა SCK სიგნალის "0"ზე დაყენების გარანტიას კვების მიწოდების დროს) SCK სიგნალის ნულზე დაყენების შემდეგ, აუცილებელია RESET გამომყვანზე მიეწოდოს დადებითი იმპულსი მიკროკონტროლერის სატაქტო სიგნალის არანაკლებ ორი პერიოდის ხანგრძლივობით.
2. დაყოვნება არა ნაკლებ 20 მწმ;
3. MOSI გამოსასვლელზე გაიგზავნოს ბრძანება "პროგრამირების ნებადართვა".

ბრძანების გადაცემის კონტროლის მიზნით ხორციელდება მე-2 ბაიტის ($53) დაბრუნება, მე-3 ბაიტის გაგზავნის დროს. თუ დაბრუნებული მნიშვნელობა განსხვავდება მითითებულისგან, მაშინ აუცილებელია RESET გამოსასვლელს მიეწოდოს დადებითი იმპულსი და კვლავ გაიგზავნოს "პროგრამირების ნებადართვის" ბრძანება. იმისგან დამოუკიდებლად თუ რა მნიშვნელობაა დაბრუნებული, საჭიროა გადაიცეს ბრძანების

ოთხივე ბაიტი. 32 მცდელობის შემდეგ $53 რიცხვითი მნიშვნელობის დაუბრუნებლობა იმის მაუწყებელი, რომ არ არსებობს კავშირი პროგრამატორსა და მიკროსქემას შორის ან მიკროსქემა გაუმართავია.

პროგრამირების დასრულების შემდეგ შესაძლებელია RESET გამოსასვლელს მიეწოდოს მაღალი დონის ძაბვა მიკროკონტროლერის მუშა რეჟიმში გადასაყვანად ან მისი გამორთვა. უკანასკნელ შემთხვევაში აუცილებელია შევასრულოთ შემდეგი მოქმედების თანამიმდევრობა:

1. XTAL1 გამოსასვლელს მიეწოდოს დაბალი დონის ძაბვა, თუ მიკროკონტროლერის ტაქტირება ხორციელდება გარე სქემით;
2. RESET გამოსასვლელზე მიეწოდება მაღალი დონის ძაბვა;
3. უნდა მოხდეს მიკროკონტროლერისათვის კვების ძაბვის გამორთვა.

ბრანებების ფორმატი, რომლებიც გამოიყენება Atmega 128 მიკროკონტროლერისთვის პროგრამირების ამ რეჟიმში მოცემულია 2 დანართში

12.3. FLASH-მეხსიერების პროგრამირების პროცესის მართვა

მიკროკონტროლერის პროგრამების მეხსიერების პროგრამირება ხორციელდება გვერდებად. დასაწყისში გვერდის შემცველობა ბაიტ-ბაიტ შეიტანება ბუფერში ბრძანებით ”FLASH – მეხსიერების გვერდის ჩატვირთვა”. თითოეული ბრძანებით გადაიცემა შესაცვლელი უჯრედის მისამართის უმცროსი თანრიგები (უჯრედის მდებარეობა შიგნით გვერში) და ჩასაწერი მნიშვნელობა. უჯრედის შემცველობა უნდა ჩაიტვირთოს შემდეგი თანამიმდევრობით: თავდაპირველად უმცროსი ბაიტი, ხოლო შემდეგ - უფროსი.

ფაქტიურად FLASH-მეხსიერების გვერდის პროგრამირება ხორციელდება გვერდის ბუფერის ჩატვირთვის შემდეგ, ბრძანებით “ FLASH- მეხსიერებაში გვერდის ჩაწერა”. ამ ბრძანებით გადაიცემა უჯრედის მისამართის უფროსი თანრიგები (გვერდის ნომერი). მომდევნო გვერდის პროგრამირება შეიძლება შესრულდეს მხოლოდ გვერდის ჩაწერის დასრულების შემდეგ.

ჩაწერის დასრულების გასარკვევად შეიძლება გამოვიყენოთ ორი ხერხიდან ერთ-ერთი. პირველი და ყველაზე უნივერსალური - ჩაწერის ბრძანებების გაგზავნის შორის გარკვეული პაუზის შესრულება. მეორე ხერხი მდგომარეობს უჯრედის შემცველობის შემოწმებაში ჩაწერის ბრძანების გაგზავნის შემდეგ: უჯრედში ჩაწერის დამთავრებამდე მისი წაკითხვის შემთხვევაში ბრუნდება მნიშვნელობა “FF”, ხოლო ჩაწერის დამთავრების შემდეგ- ჩაწერილი მნიშვნელობა. უნდა აღინიშნოს, რომ რამდენადაც ყველა გვერდების ჩაწერა ხორციელდება ერთდროულად, ჩაწერის მომენტის დამთავრების განსაზღვრისათვის მეორე ხერხით შესაძლებელია გამოვიყენოთ ნებისმიერი უჯრედი, რომელიც განთავსებულია გვერდზე.

12.4. EEPROM- მეხსიერების პროგრამირების პროცესის მართვა

EEPROM- მეხსიერების პროგრამირება ხორციელდება ბაიტ-ბაიტ “ჩაწერა EEPROM – “ ბრძანებების გადაგზავნის მეშვეობით. ყოველ ბრძანებით გადაიცემა შესაცვლელი უჯრედის მისამართი და ჩასაწერი მნიშვნელობა. მომდევნო პროგრამირება შეიძლება შესრულდეს მხოლოდ მას შემდეგ, როდესაც დასრულდება წინა უჯრედში ჩაწერა. იმისათვის, რომ განისაზღვროს ჩაწერის დასრულების მომენტი შეიძლება გამოვიყენოთ ბრძანებებს შორის პაუზა, რომლის ხანგრძლივობა განისაზღვრება შესაბამისი ცხრილით ან უნდა მოხდეს უჯრედის შემცველობის შემოწმება ჩაწერის ბრძანების შესრულების შემდეგ.

თავი 13

პარალელური პროგრამირება

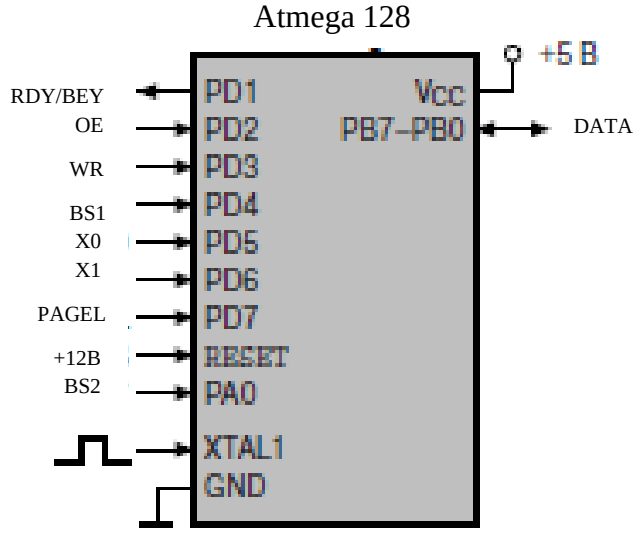
13.1. საერთო ცნებები

როგორც დასახელებიდან ჩანს, აღნიშნულ რეჟიმში პროგრამატორიდან მიკროკონტროლერს ერთდროულად გადაეცემა ბრძანების კოდის ან მონაცემთა ბაიტების ყველა თანრიგები. აღნიშნული რეჟიმის განხორციელების დროს გამოიყენება მიკროკონტროლერის გამომყვანების დიდი რაოდენობა, და ამასთან აუცილებელია მაღალი ძაბვის დამატებითი კვების წყარო (12ვ). ამიტომ, პარალელურ რეჟიმში დაპროგრამების დროს იხმარებიან სპეციალიზირებული პროგრამატორები. ამ რეჟიმის ძირითადი გამოყენებაა მიკროკონტროლერის "გაკერვა" მისი პლატაზე დაყენებამდე მასიური წარმოების პირობებში.

მიკროკონტროლერის ჩართვის სქემა პარალელური დაპროგრამების რეჟიმში ნაჩვენებია 13.1.სურ.-ზე. აღნიშნულ რეჟიმისათვის 13.1. და 13.2. ცხრილებში მითითებულია მიკროსქემის გამომყვანების სიგნალების დანიშნულება. ყურადსაღებია, რომ შემდგომში პარალელური პროგრამირების დროს გამომყვანები, რომლებიც მითითებულია 13.1. ცხრილში, აღნიშნული იქნება იმ სიგნალების სახელწოდებით, რომელიც შეესაბამება გამომყვანებზე სიგნალების მნიშვნელობებს.

ზოგადად აღნიშნულ რეჟიმში დაპროგრამების პროცესი ითვალისწინებს შემდეგი ოპერაციების მრავალჯერად შესრულებას:

- ბრძანების ჩატვირთვა;
- მისამართის ჩატვირთვა;
- მონაცემის ჩატვირთვა;
- ბრძანების შესრულება.



სურ.13.1. მიკროკონტროლერის ჩართვა პროგრამირების პარალელურ რეჟიმისათვის

ცხრილი 13.1. გამოყენებული აღნიშვნები და გამომყვანების ფუნქციები პარალელური პროგრამირების რეჟიმში

სიგნალი	გამომყვანები	შეს./გამ.	დანიშნულება
RDY/BSY	PD1	გამოსასვლელი	მოწყობილობის მდგომარეობა: <<0>>- დაკავებულია(სრულდება წინა ბრძანება) <<1>>- მზადაა მომდევნო ბრძანების შესასრულებლად
$\overline{OE}$	PD2	შესასვლელი	მონაცემთა სალტის მუშაობის რეჟიმის მართვა PB7...PB0: <<0>> - გამოსასვლელი, <<1>>- შესასვლელი
$\overline{WR}$	PD3	შესასვლელი	ჩაწერის სიგნალი (აქტიური დონე - ლოგ: <<0>>).
BSI	PD4	შესასვლელი	ბაიტის ამორჩევა (<<0>>-უმცროსი ბაიტი ,<<1>>-უფროსი ბაიტი.
XA0	PD5	შესასვლელი	განსაზღვრავს მოქმედებას, რომელიც სრულდება XTAL1 გამოსასვლელზე დადებითი იმპულსის დროს (ცხრილი 13.2).
XA1	PD6	შესასვლელი	
PAGEL*	PD7	შესასვლელი	მეხსიერების გვერდის ჩატვირთვის სიგნალი,
BS2*	PA0	შესასვლელი	ბაიტის ამორჩევა (<<0>>-უმცროსი ბაიტი, <<1>>-უფროსი ბაიტი).
DATA	PB7.....PB0	შეს./გამ.	მონაცემის ორ მიმართული სალტე.

ცხრილი 13.2. XA0 და XA1 საგნალების ფუნქციები

XA0	XA1	მოქმედებები, რომელიც სრულდება სატაქტო იმპულსზე.
0	0	მეხსიერების უჯრედის მისამართის ჩატვირთვა (უმცროსი ან უფროსი ბაიტი, BS1 სიგნალის დონის შესაბამისად).
0	1	მონაცემების ჩატვირთვა (უმცროსი ან უფროსი ბაიტი, BS1 სიგნალის დონის შესაბამისად).
1	0	ბრძანების ჩატვირთვა
1	1	არ სრულდება მოქმედება, მოლოდინის რეჟიმი.

13.3. ცხრილში მოცემულია სხვადასხვა ბაზური ოპერაციების შესრულების დროს მიკროკონტროლერის გამომყვანებზე სიგნალების თანამიმდევრული მიწოდება.

ცხრილი13.3. ბაზური ოპერაციები პარალელურ პროგრამირების რეჟიმში

№	ოპერაციის დასახელება	მოქმედება
1	ბრძანების ჩატვირთვა	1.XA,X0 გამომყვანების დაყენება <<10>> მდგომარეობაში (ბრძანების ჩატვირთვა). 2.BSI გამტარს მიეწოდოს ძაბვა ლოგ.<<0>>. 3.DATA სალტეზე გაიცეს ბრძანების კოდი. 4.XTAL1 გამტარს მიეწოდოს დადებითი იმპულსი.
2	მისამართის ჩატვირთვა	1.XA1,XA0 გამომყვანების დაყენება <<00>> მდგომარეობაში (მისამართის ჩატვირთვა). 2.BSI გამტარს მიეწოდოს ძაბვა ლოგ.<<0>> (უმცროსი ბაიტის ჩატვირთვა). 3.DATA სალტეზე გაიცეს მისამართის უმცროსი ბაიტი. 4.XTAL1 გამტარს მიეწოდოს დადებითი იმპულსი.
3	მონაცემთა ჩატვირთვა	1.XA1,XA0 გამომყვანების დაყენება <<01>> მდგომარეობაში (მონაცემის ჩატვირთვა). 2. Mega ოჯახის მიკროკონტროლერებისათვის, BSI გამტარს მიეწოდოს ძაბვა ლოგიკური <<0>> (უმცროსი ბაიტის ჩატვირთვა) ან ლოგიკური <<1>> (უფროსი ბაიტის ჩატვირთვა). 3.DATA სალტეზე გაიცეს მონაცემთა ბაიტის შემცველობა. 4.XTAL1 გამტარს მიეწოდოს დადებითი იმპულსი.
4	მონაცემთა ჩაწერა ბუფერში (მხოლოდ Atmega-სთვის)	1.BSI გამტარს მიეწოდოს ძაბვა ლოგიკური <<1>>. 2.PAGEL გამტარს მიეწოდოს დადებითი იმპულსი.
5	მეხსიერების უჯრედში ჩაწერა	1.BSI გამტარს მიეწოდოს ძაბვა ლოგიკური <<0>> (უმცროსი ბაიტის ჩაწერა) ან ლოგიკური <<1>> (უფროსი ბაიტის ჩაწერა). 2.$\overline{WR}$ გამტარს მიეწოდება უარყოფითი იმპულსი; ამასთან ერთად RDY/$\overline{BSY}$ გამტარზე დგება დაბალი დონის სიგნალი. 3.მოლოდინი, სანამ RDY/$\overline{BSY}$ გამტარზე არ დადგება მაღალი დონის სიგნალი.
	გვერდის ჩაწერა	1.BSI გამტარს მიეწოდოს ძაბვა ლოგიკური <<0>>. 2.$\overline{WR}$ გამტარს მიეწოდება უარყოფითი იმპულსი; ამასთან ერთად RDY/$\overline{BSY}$ გამტარზე დგება დაბალი დონის სიგნალი. 3.მოლოდინი, სანამ RDY/$\overline{BSY}$ გამტარზე არ დადგება მაღალი დონის სიგნალი.

განხილულ რეჟიმში გამოყენებული 9 ბრძანების კოდები მოცემულია 13.4. ცხრილში.

ცხრილი 13.4 პარალელურ რეჟიმში პროგრამირების ბრძანებები

ბრძანების კოდი	აღწერა
1000 0000	“კრისტალის წაშლა”
0100 0000	“კონფიგურაციული უჯრედების ჩაწერა”
0010 0000	“დაცვის უჯრედების ჩაწერა”
0001 0000	“FLASH მეხსიერების ჩაწერა”
0001 0001	“EEPROM მეხსიერებაში ჩაწერა”
0000 1000	“იდენტიფიკატორის წაკითხვა”
0000 0100	“კონფიგურაციის უჯრედების და დაცვის უჯრედების წაკითხვა”
0000 0010	“FLASH მეხსიერების წაკითხვა”
0000 0011	“EEPROM მეხსიერების წაკითხვა”

13.2. პროგრამირების პარალელურ რეჟიმზე გადართვა

მიკროკონტროლერის პროგრამირების დროს პირველი ოპერაცია არის მისი გადაყვანა პროგრამირების რეჟიმში. მიკროკონტროლერ Atmega 128 გადასართავად პროგრამირების რეჟიმში საჭიროა შესრულდეს შემდეგი მოქმედებები:

1. მიკროკონტროლერს უნდა მიეწოდოს კვების ძაბვა;
2. RESET გამომყვანს უნდა მიეწოდოს დაბალი დონის ძაბვა და XTAL1 გამომყვანზე დაფორმირდეს არა ნაკლებ სამი იმპულსისა.
3. PAGEL, XA1, XA0, BS1 გამომყვანებს უნდა მიეწოდოს დაბალი დონის ძაბვა არა უმეტეს 100 ნწამის ხანგრძლივობით;
4. RESET გამომყვანზე მიეწოდოს 11.5.....12.5 ვ. ძაბვა, ხოლო PAGEL, XA1, XA0, BS1 გამომყვანებს მიეწოდოს დაბალი დონის ძაბვა მინიმუმ 100 ნწ ხანგრძლივობით. ამ დროის განმავლობაში აღნიშნულ გამომყვანებზე ნებისმიერი აქტივობის შემთხვევაში მიკროკონტროლერი არ გადავა პროგრამირების რეჟიმში

13.3. კრისტალის წაშლა

“კრისტალის წაშლის” ბრძანება უნდა შესრულდეს მიკროკონტროლერის ყოველი ახალი პროგრამირების წინ. მოცემული ბრძანება მთლიანად შლის FLASH და EEPROM მეხსიერებების შემცველობას, რის შემდეგაც ხორციელდება დაცვის უჯრედების ჩამოგდება (“1”-ის ჩაწერა). კონფიგურაციის უჯრედების მდგომარეობაზე ეს ბრძანება ზემოქმედებას ვერ ახდენს. შესაძლებელია თავიდან ავიცილოთ EEPROM მეხსიერების წაშლა EESAVE კონფიგურაციის უჯრედის დაპროგრამირების საშუალებით.

“კრისტალის წაშლის” ბრძანების შესრულებისთვის აუცილებელია შესრულდეს შემდეგი მოქმედებები:

1. ჩაიტვირთოს ბრძანება “კრისტალის წაშლა” (“1000 0000” კოდი);
2. მიეწოდოს WR გამომყვანს უარყოფითი იმპულსი; RDY/BSY გამომყვანზე დგება დაბალი დონის სიგნალი;
3. დაველოდოთ RDY/BSY გამომყვანზე მაღალი დონის სიგნალის გამოჩენას.

13.4. FLASH მეხსიერების პროგრამირება

FLASH -- მეხსიერებაში ჩაწერა

FLASH მეხსიერებაში ჩაწერა ხორციელდება შემდეგი თანამიმდევრობით (ჩაწერის პროცესის ყოველი ეტაპის რეალიზაცია ნაჩვენებია 13.4. ცხრილში);

1. ჩაიტვირთოს ბრძანება "FLASH მეხსიერების ჩაწერა" ("0001 0000"კოდი).
2. ჩაიტვირთოს მისამართის უმცროსი ბაიტი (უჯრედის განლაგება გვერდის შიგნით).
3. ჩაიტვირთოს მონაცემის უმცროსი ბაიტი.
4. ჩაიტივირთოს მონაცემის უფროსი ბაიტი.
5. განხორციელდეს მონაცემის დამახსოვრება ბუფერში;
6. გამეორდეს 2.....5 ქვეპუნქტები გვერდის ბუფერის შევსებამდე.
7. ჩაიტვირთოს მისამართის უფროსი ბაიტი (გვერდის ნომერი);
8. ჩაიწეროს გვერდი;
9. გამეორდეს 2.....8 ქვეპუნქტები პროგრამის მეხსიერების დანარჩენი გვერდების ჩაწერისათვის.
10. დავასრულოთ პროგრამირება "არ არის ოპერაცია" ბრძანების ჩატვირთვით ("0000 0000" კოდი)

საჭიროა აღინიშნოს, რომ მეხსიერების უჯრედის დამისამართებისთვის გვერდის შიგნით საჭიროა 8 თანრიგზე ნაკლები (გვერდის უდიდესი ზომა 128 სიტყვას შეადგენს). მისამართის უმცროსი ბაიტის დარჩენილი უფროსი თანრიგი გამოიყენება გვერდის დამისამართების დროს "გვერდის ჩაწერის" ბრძანების შესრულების შემთხვვაში.

FLASH -- მეხსიერების წაკითვა

FLASH - მეხსიერების წაკითხვისათვის საჭიროა შესრულდეს შემდეგი მოქმედებები (თითოეული ეტაპის რეალიზაცია ნაჩვენებია 13.3.ცხრილში):

1. ჩაიტვირთოს ბრძანება" FLASH მეხსიერების წაკითხვა" ("0000 0010"კოდი);
2. ჩაიტვირთოს მისამართის უფროსი ბაიტი;
3. ჩაიტვირთოს მისამართის უმცროსი ბაიტი;
4. დაყენდეს OE და BS1 "0"-ს მდგომარეობაში, რის შემდეგაც შესაძლებელია DATA მონაცემთა სალტიდან მეხსიერების უჯრედის უმცროსი ბაიტის შემცველობის წაკითხვა;
5. დაყენდეს BS1 "1"-ში, რის შემდეგაც შესაძლებელია განხორციელდეს DATA მონაცემთა სალტიდან მეხსიერების უჯრედის უფროსი ბაიტის შემცველობის წაკითხვა;
6. მოხდეს OE "1"-ში დაყენება.

13.5. EEPROM მეხსიერების პროგრამირება

EEPROM მეხსიერებაში ჩაწერა

Atmega 128 EEPROM მეხსიერება ორგანიზებულია სტრიქონებად. ჩაწერა სრულდება შემდეგი თანამიმდევრობით:

1. ჩაიტვირთოს ბრძანება "EEPROM მეხსიერებაში ჩაწერა" ("0001 00001"კოდი);

2. ჩაიტვირთოს მისამართის უფროსი ბაიტი;
3. ჩაიტვირთოს მისამართის უმცროსი ბაიტი;
4. ჩაიტვირთოს მონაცემის ბაიტი;
5. მონაცემთა დამახსოვრება ბუფერში;
6. გამეორდეს 3.....5-ე ქვეპუნქტების ბუფერის სრულ შევსებამდე;
7. გვერდის ჩაწერა.

EEPROM - მეხსიერების წაკითხვა

EEPROM - მეხსიერების შემცველობის წაკითხვისთვის აუცილებელია შესრულდეს შემდეგი მოქმედებები (ყოველი ეტაპის რეალიზაცია ნაჩვენებია 13.3. ცხრილში):

1. ჩაიტვირთოს ბრძანება "EEPROM - მეხსიერების წაკითხვა" ("0000 0011" კოდი) ;
2. ჩაიტვირთოს მისამართის უფროსი ბაიტი;
3. ჩაიტვირთოს მისამართის უმცროსი ბაიტი;
4. OE და BS1 "0" –ში დაყენება, რის შემდეგაც შეიძლება განხორციელდეს Data მონაცემთა სალტიდან მეხსიერების უჯრედის შემცველობის წაკითხვა;
5. OE <<1>> -ში დაყენება.

13.6. მიკროკონტროლერის კონფიგურირება

Atmega 128 მიკროკონტროლერს გააჩნია რამოდენიმე მაკონფიგურირებელი ბაიტი, რომელთა პროგრამირება ხორციელდება ქვემოთ ჩამოთვლილი თანმიმდევრობით:

უმცროსი მაკონფიგურირებული ბაიტის პროგრამირება

1. მიკროკონტროლერში ჩაიტვირთოს ბრძანება "მაკონფიგურირებულ უჯრედში ჩაწერა"("0100 0000" კოდი);
2. მიკროკონტროლერში ჩაიტვირთოს მონაცემის უმცროსი ბაიტი;
3. BS1 და BS2 გამომყვანების "0"-ში დაყენება;
4. WR გამომყვანს მიეწოდოს უარყოფითი იმპულსი და განხორციელდეს ლოდინი RDY/BSY გამომყვანზე მაღალი დონის სიგნალის შემოსვლამდე.

უფროსი მაკონფიგურირებული ბაიტის პროგრამირება

1. მიკროკონტროლერში ჩაიტვირთოს ბრძანება "მაკონფიგურირებულ უჯრედში ჩაწერა" ("0100 0000" კოდი);
2. მიკროკონტროლერში ჩაიტვირთოს მონაცემის უმცროსი ბაიტი;
3. BS1-ს და BS2 გამომყვანების "0"-ში დაყენება;
4. WR გამომყვანს მიეწოდოს უარყოფითი იმპულსი და განხორციელდეს ლოდინი RDY/BSY გამომყვანზე მაღალი დონის სიგნალის შემოსვლამდე.
5. BS1 "0"- ში ჩამოგდება.

დამატებითი მაკონფიგურირებელი ბაიტების პროგრამირება

1. მიკროკონტროლერში ჩაიტვირთოს ბრძანება “მაკონფიგურირებელი უჯრედების ჩაწერა” (“0100 0000”კოდი);
2. მიკროკონტროლერში ჩაიტვირთოს მონაცემის უმცროსი ბაიტი BS1 და BS2 “0”-ში დაყენება;
3. WR გამომყვანს მიეწოდოს უარყოფითი იმპულსი და განხორციელდეს ლოდინი RDY/BSY გამომყვანზე მაღალი დონის სიგნალის შემოსვლამდე;
4. BS2-ს “0”-ში ჩამოგდება.

დაცვის უჯრედების პროგრამირება

დაცვის უჯრედების პროგრამირება ისევე ხორციელდება, როგორც მაკონ-ფიგურირებელი უჯრედების:

1. ჩაიტვირთოს ბრძანება “დაცვის უჯრედების ჩაწერა” (“0010 0000” კოდი) ;
2. ჩაიტვირთოს მონაცემების ბაიტი.
3. მოხდეს მონაცემების უმცროსი ბაიტის ჩაწერა

მაკონფიგურირებელი და დაცვის უჯრედების წაკითხვა

ეს ოპერაციები სრულდებიანა შემდეგი თანამიმდევრობით:

1. ჩაიტვირთოს ბრძანება “მაკონფიგურირებელი და დაცვის უჯრედების წაკითხვა” («0000 0100»კოდი) ;
2. OE, BS1 და BS2 დაყენება “0”-ში, რის შემდეგ DATA მონაცემების სალტიდან შეიძლება მაკონფიგურებელი უმცროსი ბაიტის მნიშვნელობის ამოკითხვა;
3. OE, BS1 и BS2 დაყენება “1”-ში, რის შემდეგ DATA მონაცემების სალტიდან შეიძლება მოხდეს მაკონფიგურებელი უფროსი ბაიტის მნიშვნელობის ამოკითხვა;
4. OE, BS1 დაყენება “0”–ში და BS2-ს დაყენება “1”-ში, რის შემდგომაც DATA მონაცემების სალტიდან შესაძლებელია მოხდეს დამატებითი მაკონფიგუ-რირებელი ბაიტის მნიშვნელობის ამოკითხვა;
5. OE-ს “0” –ში დაყენება, BS1-ს დაყენება “1 “-ში, ხოლო BS2-ს “0”-ში, რის შემდეგაც შესაძლებელია მოხდეს DATA მონაცემების სალტიდან დაცვის ბაიტის ამოკითხვა;
6. OE-ს “1” -ში დაყენება.

იდენტიფიკატორის და მაკალიბრებელი კონსტანტების უჯრედების წაკითხვა

იდენტიფიკატორების უჯრედების წაკითვა ხორციელდება შემდეგი თანამიმდევ-რობით:

1. ჩაიტვირთოს ბრძანება “იდენფიკატორების უჯრედების წაკითხვა” (“0000 1000” კოდი);
2. ჩაიტვირთოს მისამართის უმცროსი ბაიტი;

3. OE და BS1-ს დაყენება “0”-ში, რის შემდგომაც შეიძლება განხორციელდეს ამორჩეული იდენტიფიკატორის უჯრედის შემცველობის წაკითხვა DATA სალტიდან.
4. განხორციელდეს OE დაყენება “1” -ში. მაკალიბრებელი კონსტანტების წაკითხვა ხორციელდება ანალოგიურად და იმავე ბრძანებებით:
5. ჩაიტვირთოს ბრძანება “იდენტიფიკატორის უჯრედების წაკითხვა” («0000 1000» კოდი);
6. ჩაიტვირთოს მისამართის უმცროსი ბაიტის ($00…$03);
7. განხორციელდეს OE დაყენება “0”-ში, ხოლო BS1-ს “1”-ში, რის შემდგომაც შესაძლებელია DATA მონაცემთა სალტიდან განხორციელდეს ამორჩეული მაკალიბრებრებელი კონსტანტის წაკითხვა;
8. OE “1”-ში დაყენება.

თავი 14

მიკროკონტროლერების თვითპროგრამირება

14.1. საერთო ცნობები

Mega ოჯახის ყველა მიკროკონტროლერს გააჩნია თვითპროგრამირების შესაძლებლობა ანუ პროგრამის მეხსიერების შემცველობის დამოუკიდებლად შეცვალა. ეს თავისებურება შესაძლებლობას იძლევა, რომ მის საფუძველზე შეიქმნას მეტად მოქნილი სისტემები, რომელთა მუშაობის ალგორითმი შეიცვლება თვით მიკრო-კონტროლერის მიერ რაიმე შიგა პირობის ან გარე მოვლენების გათვალისწინებით.

თვითპროგრამირების მხარდაჭერისათვის მთლიანი პროგრამის მეხსიერების არე ლოგიკურად ორ სექციად არის დაყოფილი – გამოყენებითი პროგრამების სექცია (Application Section) და ჩატვირთვის სექცია (Boot Loader Section). პროგრამის მეხსიერების ცვლილება ხორციელდება ჩამტვირთავი პროგრამით, რომელიც განთავსებულია ერთსახელა სექციაში. ჩამტვირთავ პროგრამას პროგრამის მეხსიერების ახალი შემცველობის ჩატვირთვისათვის და ასევე ძველი შემცველობის ამოტვირთვისათვის შეუძლია გამოიყენოს მონაცემთა გადაცემის ნებისმიერი (USART/UART, SPI, TWI) ინტერფეისი, რომელიც გააჩნია მიკროკონტროლერს თავის შემადგენლობაში. აქვე უნდა აღინიშნოს, რომ ჩამტვირთავს აქვს უნარი შეცვალოს ორივე სექციის შემცველობა. ეს აძლევს მას იმის საშუალებას, რომ მოახდინოს საკუთარი პროგრამების მოდიფიცირება და ასევე განახორციელოს თავისი პროგრამის ამოღება მეხსიერებიდან იმ შემთხვევაში, თუ მასზე საჭიროება აღარ იქნება. მიმართვის (წაკითხვის/ჩაწერის) დონე ყოველი სექციისთვის გაიცემა მომხმარებლის მიერ დაცვის BLB02:BLB01 და BLB12:BLB11 უჯრედების მეშვეობით, რომელიც 11-ე თავის 11.2 ცხრილში იყო აღწერილი.

გადასვლა ჩამტვირთავ პროგრამაზე შეიძლება მოხდეს სხვადასხვა გზით. კერძოდ მისი გამოძახება შეიძლება განხორციელდეს CALL/JMP ბრძანებებით ძირითადი პროგრამიდან. ასევე შეიძლება განხორციელდეს ჩატვირთვის სექციის დასაწყისში განულების ვექტორის გადაადგილებით. ასეთ შემთხვევაში ჩატვირთვის პროგრამის გაშვება ავტომატურად ხორციელდება მიკროკონტროლერის ყოველი განულების შემდეგ. განულების ვექტორის პოზიცია განისაზღვრება BOOTRST კონფიგურაციული უჯრედის მდგომარეობით. თუ მისი შემცველობა არის “1”, მაშინ განულების ვექტორი განთავსებულია პროგრამის მეხსიერების დასაწყისში $0000 მისამართზე. დაპროგრამირებული უჯრედის შემთხვევაში, როდესაც მასში ჩაწერილია “0”, განულების ვექტორი განთავსებულია ჩატვირთვის სექციის დასაწყისში (მისამართები იხილეთ 14.1. ცხრილში).

ჩამტვირთავი და გამოყენებითი პროგრამების სექციის ზომები განისაზღვრება BOOTSZ1:BOOTSZ0 ორი მაკონფიგურირებელი უჯრედების მეშვეობით. მიკროკონტროლერის პროგრამის მეხსიერების შესაძლებელი კონფიგურაციები ნაჩვენებია 14.1. ცხრილში.

ცხრილი 14.1. პროგრამის მეხსიერების კონფიგურაცია

BOOTSZ1	BOOTSZ0	ჩამტვირთავის ზომა [სიტყვებში]	გვერდები	გამოყენებითი პროგრამის სექცია	ჩამტვირთავის სექცია
1	1	512	4	$0000…$FDFF	$FE00…$FFFF
1	0	1024	8	$0000…$FBFF	$FC00…$FFFF
0	1	2048	16	$0000…$F7FF	$F800…$FFFF
0	0	4096	32	$0000... $FFFF	$F000…$FFFF

14.2. ჩამტვირთავის ფუნქციონირება

თვითპროგრამირების პროცესის მართვა

პროგრამის მეხსიერების შემცველობის და დაცვის უჯრედების შეცვლა ხორციელდება SPM (Store Program Memory) ბრძანების მეშვეობით. ამ ბრძანების პარამეტრებია: მეხსიერების მისამართი, რომელიც იტვირთება წინასწარ ინდექსურ Z რეგისტრში, და საჭიროების შემთხვევაში მონაცემები, რომელიც განთავსებული არიან R1:R0 წყვილ რეგისტრებში.

დაპროგრამების პროცესის მართვა და კერძოთ განსაზღვრა იმ ოპერაციების, რომელიც გამოიყენება SPM ბრძანების გამოძახებისისათვის სრულდება SPMCR (Store Program Memory Control Register) შეტანა/გამოტანის რეგისტრის დახმარებით. ამ რეგისტრის ფორმატი ნაჩვენებია 14.1.სურ.-ზე, ხოლო მისი თანრიგების აღწერა ნაჩვენებია 14.3.ცხრილში.

	7	6	5	4	3	2	1	0
	X	X	-	X	BLBSET	PGWRT	PGERS	SPMEN
წაკითხვა (R) ჩაწერა (W)	R/W	R	R	R/W	R/W	R/W	R/W	R/W
	0	0	0	0	0	0	0	0

სურ.14.1. SPMCR რეგისტრის ფორმატი

ცხრილი 14.3. SPMCR რეგისრის თანრიგები

თანრიგი	დასახელება	აღწერა
7	-	არ გამოიყენება
6	-	არ განიხილება
5	-	არ განიხილება
4	-	არ განიხილება
3	BLBSET	**დაცვის ჩატვირთვის უჯრედის ცვლილება.** ამ თანრიგის და SPMEN თანრიგის ერთდროული დაყენებისთვის, ხორციელდება SPM ბრძანების გაშვება 4 მანქანური ციკლის ხანგრძლივობით, რის შემდეგაც ხდება დაცვის უჯრედების ჩატვირთვა R0 რეგისტრის შემცველობის შესაბამისად. BLBSET თანრიგის განულება ხორციელდება აპარატურის მეშვეობით, მას შემდეგ როდესაც მოხდება დაცვის უჯრედების დაყენება ან მითითებული დროის გასვლის შემდეგ.

გაგრძელება

თანრიგი	დასახელება	აღწერა
2	PGWRT	**გვერდის ჩაწერა.** ამ თანრიგის და SPMEN თანრიგის ერთდროული დაყენებისთვის, ხორციელდება SPM ბრძანების გაშვება 4 მანქანური ციკლის განმავლობაში, ის ანხორციელებს პროგრამის მეხსიერების გვერდის ჩაწერას დროებითი ბუფერიდან. გვერდის მისამართი ჩატვირთული უნდა იყოს Z (R31) რეგისტრის უფროს ბაიტში. PGWRT თანრიგის განულება ხორციელდება აპარატურის მეშვეობით გვერდის ჩაწერის შემდეგ ან მითითებული დროის გასვლის შემდეგ.
1	PGERS	**გვერდის წაშლა**. ამ თანრიგის და SPMEN თანრიგის ერთდროული დაყენებისთვის, ხორციელდება SPM ბრძანების გაშვებული 4 მანქანური ციკლის განმავლობაში, გვერდის წაშლის შემდეგ პროგრამის მეხსიერების დროებითი ბუფერიდან. გვერდის მისამართი ჩატვირთული უნდა იყო Z (R31) რეგისტრის უფროს ბაიტში. PGERS თანრიგის განულება ხორციელდება აპარატურული საშუალებით გვერდის წაშლის დამთავრების შემდეგ ან მითითებული დროის გასვლის შემდეგ.
0	SPMEN	SPM ბრძანების შესრულების ნებადართვა. ამ თანრიგის დაყენება იძლება იმის საშუალებას, რომ მოხდეს SPM ბრძანების გაშვების ნებადართვა 4 მანქანური ციკლის განმავლობაში. თუ SPMEN თანრიგის დაყენება მოხდა BLBSET, PGWRT ან PGERS ერთ-ერთ თანრიგთან ერთდროულად სრულდება ოპერაცია, რომელიც განისაზღვრება ამ თანრიგით (ნახეთ თანრიგების აღწერა). თუ ხდება მხოლოდ SPMEN თანრიგის დაყენება, ხორციელდება R1:R0 რეგისტრის შემცველობის შენახვა დროებით Z რეგისტრში. SPMEN თანრიგის განულება ხდება აპარატურული საშუალებით ოპერაციის დასრულების ან მითითებული დროის ამოწურვის შემდეგ.

EEPROM - მეხსიერებაში ჩაწერის დროს SPMCR რეგისტრის ცვლილება შეუძლებელია. აქედან გამომდინარე სანამ SPMCR რეგისტრში ხდება რაღაც ინფორმაციის ჩაწერა, რეკომენდებულია დაველოდოთ EECR რეგისტრის EEWE ალმის განულებას.

პროგრამის მეხსიერების შეცვლა

პროგრამის მეხსიერების შემცველობის შეცვლა ხდება შემდეგი თანამიმდევრობით:

1. გვერდის დროებითი ბუფერის შევსება ახალი მნიშვნელობებით;
2. გვერდის გასუფთავება;
3. ბუფერის შემცველობის გადატანა პროგრამის მეხსიერებაში.

საჭიროა აღინიშნოს, რომ გვერდის გასუფთავება შეიძლება განხორციელდეს, როგორც ბუფერის შევსების შემდეგ, ასევე მისი შევსების დაწყებამდე. მაგრამ იმ შემთხვევაში, როდესაც საჭირო ხდება მხოლოდ გვერდის ნაწილის შეცვლა ზემოთ აღნიშნული მოქმედებათა ჩამონათვალი, გასაგები მიზეზების გამო არის ერთადერთი. ამ დროს უკრედის შემცველობა, რომელიც არ მოითხოვს შეცვლას, შეინახება ბუფერში გვერდის წაშლამდე.

ოპერაციის დასრულების მომენტის განსაზღვრისათვის, შესაძლებელია ან SPMCR რეგისტრის SPMEN ალამის მდგომარეობის შემოწმება მისი ჩამოგდების მოლოდინით, ან გამოვიყენოთ წყვეტა “SPM მზადყოფნა”. ეს წყვეტა გენერირდება ყოველთვის, სანამ SPMEN ალამი ჩამოგდებულია. ამ უკანასკნელ შემთხვევაში წყვეტის ვექტორის ცხრილი განთავსებული უნდა იყოს ჩატვირთვის სექციაში, ხოლო აღნიშნული წყვეტა ნებადართული SPMCR რეგისტრის SPMIE ალმის დაყენებით.

პროგრამის მეხსიერების გვერდის წაშლისათვის საჭიროა Z რეგისტრში შეტანილი იყოს გვერდის მისამართი, SPMCR რეგისტრში ჩაიწეროს “x0000011” მნიშვნელობა და ოთხი მანქანური ციკლის ხანგრძლივობის განმავლობაში შესრულდეს SPM ბრძანება.

ბუფერში ბრძანების სიტყვის ჩაწერისათვის აუცილებელია უჯრედის მისამართი ჩაიტვირთოს Z რეგისტრში, ხოლო ოპერაციის კოდი R1:R0 რეგისტრებში.

ამის შემდეგ აუცილებელია SPMCR რეგისტრში მოხდეს “x0000011” მნიშვნელობის ჩაწერა და ოთხი მანქანური ციკლის ხანგრძლივობის განმავლობაში უნა შესრულდეს SPM ბრძანება. ბუფერის გასუფთავება ხორციელდება ავტომატურად გვერდის ჩაწერის დასრულების შემდეგ ან ხელით SPMCR რეგისტრის RWWSRE თანრიგში ლოგიკური “1”-ის ჩაწერით. უნდა აღვნიშნოთ, რომ ბუფერის ერთი და იმავე მისამართზე ჩაწერა შეუძლებელია გასუფთავების გარეშე.

პროგრამულ მეხსიერებაში ბუფერული მეხსიერების შემცველობის ჩაწერა ხორციელდება ანალოგიურად. Z რეგისტრში შეაქვთ გვერდის მისამართი, SPMCR რეგისტრში ჩაიწერება “x0000101” მნიშვნელობა და ოთხი მანქანური ციკლის განმავლობაში სრულდება SPM ბრძანება.

14.3. ჩამტვირთავის დაცვის უჯრედის შეცვლა

BLB12:BLB11 და BLB02:BLB01 ჩამტვირთავის დაცვის უჯრედების შეცვლა აგრეთვე SPM ბრძანებით ხორციელდება. ამისთვის აუცილებელია R0 რეგისტრში ჩაიტვირთოს მოთხოვნილი მნიშვნელობა 14.2. სურ-ის შესაბამისად.

	7	6	5	4	3	2	1	0
	1	1	BLB12	BLB11	BLB02	BLB01	1	1
წაკითხვა (R) ჩაწერა (W)	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
საწყისი მნიშვნელობა	1	1	1	1	1	1	1	1

სურ14.2. RO რეგისტრის შემცველობა დაცვის უჯრედის მდგომარეობის შეცვლის შემთხვევაში SPM ბრძანების გამოყენებით

ამის შემდეგ SPMCR რეგისტრში უნდა ჩაიწეროს “x0001001” მნიშვნელობა და ოთხი მანქანური ციკლის განმავლობაში შესრულდეს SPM ბრძანება.

14.4. მაკონფიგურირებელი და დაცვის უჯრედების წაკითხვა

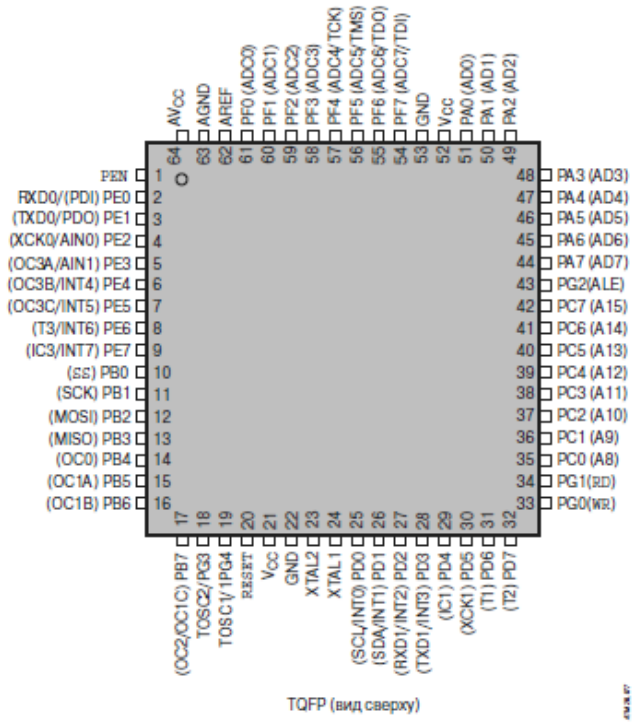
გარდა მიკროკონტროლერის დაპროგრამებისა, ჩამტვირთავს ასევე შეუძლია განახორციელოს მაკონფიგურებელი და დაცვის უჯრედების წაკითხვა. დაცვის ბაიტის წაკითხვისათვის საჭიროა Z რეგისტრში ჩაიტვირთოს $0001 რიცხვი, SPMCR რეგისტრში ჩაიწეროს “x0001001” მნიშვნელობა და შესრულდეს LPM ბრძანება. რის შედეგადაც დაცვის ბაიტი შეტანილი იქნება საერთო დანიშნულების მოცემულ რეგისტრში. რეგისტრის უჯრედების შესაბამისი თანრიგები ნაჩვენებია 11.1 სურ-ზე.

მაკონფიგურებელი ბაიტების წაკითხვა ხორციელდება ანალოგიურად. Z რეგისტრში ჩაიტვირთება ბაიტის მისამართის ($0000 - უმცროსი ბაიტი, $0003- უფროსი ბაიტი, $0002 - დამატებითი ბაიტი), რის შემდეგაც საჭიროა SPMCR რეგისტრში ჩაიწეროს “x0001001” მნიშვნელობა და შესრულდეს LPM ბრძანება. ბრძანების შესრულების შედეგად მაკონფიგურირებელი ბაიტის შემცველობა შეტანილი იქნება საერთო დანიშნულების რეგისტრში.

Atmega 128 მიკროკონტროლერის გამომყვანების აღწერა

ქვემოთ ნაჩვენებია გამომყვანების დასახელება და მითითებულია მათი ფუნქციები (როგორც მთავარი ასევე დამატებითი). გარდა ამისა ცხრილში A-1. ყოველი გამომყვანისთვის მითითებულია მათი ტიპი (შესასვლელი, გამოსასვლელი, შეტანა/გამოტანა, კვების გამომყვანი).

ცხრილში გამოყენებულია შემდეგი აღნიშვნები: I – შესასვლელი, O – გამოსასვლელი, I /O – შეტანა/გამოტანა, P - კვების გამომყვანები.



Atmega128(L) მოდულის გამომყვანების განლაგება

ცხრილი A-1. Atmega 128 მოდულის გამომყვანების აღწერა

აღნიშვნა	გამომყვანების ნომერი	გამომყვანების ტიპები	აღწერა
XTAL1	24	I	ტაქტური გენერატორის შესასვლელი
XTAL2	23	O	ტაქტური გენერატორის გამოსასვლელი
$\overline{\text{RESET}}$	20	I	განულების შესასვლელი. იმ შემთხვევაში ,როდესაც შესასვლელზე ხდება დაბალი დონის 50 ნწმ დაყოვნება, ხორციელდება მოწყობილობის განულება
პორტი A. 8 თანრიგიანი ორ მიმართული შეტანა /გამოტანის პორტი შიგა მომჭიმავი რეზისტორებით			
PA0 (AD0)	51	I/O	A0 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
PA1 (AD1)	50	I/O	A1 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
PA2 (AD2)	49	I/O	A2 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
PA3 (AD3)	48	I/O	A3 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
PA4 (AD4)	47	I/O	A4 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
PA5(AD5)	46	I/O	A5 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
PA6 (AD6)	45	I/O	A6 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
PA7 (AD7)	44	I/O	A7 (მულტიპლექსირებული მის.სალ./ მონ.სალ. გარე ოდმ-თვის)
პორტი B. 8 თანრიგიანი ორ მიმართული შეტანა /გამოტანის პორტი შიგა მომჭიმავი რეზისტორებით			
$\overline{\text{PB0(SS)}}$	10	I/O	B0(PSI სალტზე Slave-მოწყობილობის ამორჩევა)
PB1(SCK)	11	I/O	B1((Master) გამოსასვლელი ან SPI მოდულის (Slave) ტაქტური სიგნალის შესასვლელი)
PB2(MOSI)	12	I/O	B2((Master) გამოსასვლელი ან SPI მოდულის (Slave) მონაცემების შესასვლელი)
PB3(MISO)	13	I/O	B3 (Master) შესასვლელი ან SPI მოდულის Slave მონაცემების გამოსასვლელი)
PB4(OC0)	14	I/O	B4 (T0 ტაიმერ / მთვლელი ს გამოსასვლელი (Compare,PWM რეჟიმები))
PB5(OCIA)	15	I/O	B5((T1ტაიმერ/ მთვლელის A გამოსასვლელი (Compare,PWM რეჟიმებისთვის))
PB6(OCIB)	16	I/O	B6((T1 ტაიმერ/ მთვლელის B გამოსასვლელი (Compare,PWM რეჟიმებისთვის))
PB7(OC2/OCIC)	17	I/O	B7 (T2ტაიმერ/ მთვლელის გამოსასვლელი (რეჟიმები Compare,PWM / T1ტაიმერ მთვლელის C გამოსასვლელი / (Compare,PWM რეჟიმებისთვის))
პორტი c. 8 თანრიგიანი ორ მიმართული შეტანა /გამოტანის პორტი შიგა მომჭიმავი რეზისტორებით			
PC0(A8)	35	I/O	C0 (გარე სამ.ისამართე სალტე)
PC1(A9)	36	I/O	C1(გარე სამისამართე სალტე)
PC2(A10)	37	I/O	C2(გარე სამისამართე სალტე)
PC3(A11)	38	I/O	C3(გარე სამისამართე .სალტე)
PC4(A12)	39	I/O	C4(გარე სამისამართე სალტე)
PC5(A13)	40	I/O	C5(გარე სამისამართე სალტე)
PC6(A14)	41	I/O	C6(გარე სამისამართე სალტე)
PC6(A15)	42	I/O	C7(გარე სამისამართე სალტე)

აღნიშვნა	გამომყვანების ნომერი	გამომყვანებ ის ტიპები	აღწერა
პორტი D. 8 თანრიგიანი ორ მიმართული შეტანა /გამოტანის პორტი შიდა მომჭიმავი რეზისტორებით			
PD0(SCL/INT0)	25	I/O	D0 (TWI მოდულის ტაქტური სიგნალი/გარე წყვეტის შესასვლელი)
PD1(SDA/INT1)	26	I/O	D1(TWI მოდულის მონაცემთა სადენები/გარე წყვეტის შესასვლელი)
PD2(RXDI/INT2)	27	I/O	D2(მე-2 USART შესასვლელი/ გარე წყვეტის შესასვლელი)
PD3(TXDI/INT3)	28	I/O	D3(მე-2 USART გამოსასვლელი/ გარე წყვეტის შესასვლელი)
PD4(ICPI)	29	I/O	D4(T1 ტაიმერ/ მთვლელის დაპყრობის შესასვლელი (Capture რეჟიმი))
PD5(XCKI)	30	I/O	D5 (მე-2 USART გარე ტაქტური შეტანა/გამოტანის სიგნალი (Captur რეჟიმი))
PD6(TI)	31	I/O	D6(T1ტაიმერის/ მთვლელის გარე ტაქტური სიგნალის შესასვლელი)
PD7(T2)	32	I/O	D7(T2ტაიმერ/მთვლელის გარე ტაქტური სიგნალის შესასვლელი)
პორტი E. 8 თანრიგიანი ორ მიმართული შეტანა /გამოტანის პორტი შიგა მომჭიმავი რეზისტორებით			
PE0(RXD0/PDI)	2	I/O	E0(1-ლი USART შესასვლელი/ მიმდევრობითი დაპროგრამების დროს მონაცემთა შესასვლელი)
PE1(TXD0/PDO)	3	I/O	E1(1-ლი USART გამოსასვლელი/ მიმდევრობითი დაპროგრამების დროს მონაცემთა გამოსასვლელი)
PE2(XCK0/AIN0)	4	I/O	E2(1-ლი USART გარე ტაქტური სიგნალის შეტანა/გამოტანა/კომპარატორის დადებითი შესასვლელი)
PE3(OC3A/AIN1)	5	I/O	E3(T3ტაიმერ/ მთვლელის A გამოსასვლელი (Compare,PWM რეჟიმები)/კომპარეტორის უარყოფითი შესასვლელი)
PE4(OC3B/INT4)	6	I/O	E4 (T3 ტაიმერ/ მთვლელი B გამოსასვლელი (Compare,PWM რეჟიმები)/გარე წყვეტის შესასვლელი)
PE5(OC3C/INT5)	7	I/O	E5(T3 ტაიმერის/ მთვლელის C გამოსასვლელი (Compare,PWM რეჟიმები)/გარე წყვეტის შესასვლელი)
PE6(T3/INT6)	8	I/O	E6(T1ტაიმერის/ მთვლელის გარე ტაქტური სიგნალის შესასვლელი /შიგა წყვეტის შესასვლელი)
PE7(ICP3/INT7)	9	I/O	E7(T1ტაიმერ/ მთვლელის დაპყრობის შესასვლელი (Compare,PWM რეჟიმები)/გარე წყვეტის შესასვლელი
პორტი F. 8 თანრიგიანი ორ მიმართული შეტანა /გამოტანის პორტი შიგა მომჭიმავი რეზისტორებით			
PF0(ADC0)	61	I/O	F0(აცგ შესასვლელი)
PF1(ADC1)	60	I/O	F1(აცგ შესასვლელი)
PF2(ADC2)	59	I/O	F2(აცგ შესასვლელი)
PF3(ADC3)	58	I/O	F3(აცგ შესასვლელი)
PF4(ADC4/TCK)	57	I/O	F4(აცგ შესასვლელი,/ JTAG ტაქტური სიგნალი)
PF5(ADC5/TM5)	56	I/O	F5(აცგ შესასვლელი,/JTAG ამორჩევის რეჟიმი)
PF6(ADC6/TDO)	55	I/O	F6(აცგ შესასვლელი,/JTAG მონაცემების გამოსასვლელი)
PF7(ADC7/TDI)	54	I/O	F7(აცგ შესასვლელი/ JTAG მონაცემის შესასვლელი)

აღნიშვნა	გამომყვანების ნომერი	გამომყვანების ტიპები	აღწერა
პორტი G. 8 თანრიგიანი ორ მიმართული შეტანა /გამოტანის პორტი შიდა მომჭიმავი რეზისტორებით			
$\overline{\text{PG0(WR)}}$	33	I/O	G0 (გარე ომმ ჩაწერის სტრობი)
$\overline{\text{PG1(RD)}}$	34	I/O	G1 (გარე ომმ წაკითხვის სტრობი)
PG2	43	I/O	G2 (გარე ომმ მისამართის სტრობი)
PG3	18	I/O	G3(შესასვლელი , T2 ტაიმერ/ მთვლელთან რეზისტორის მიერთებისათვის)
PG4	19	I/O	G4(გამოსასვლელი, T2 ტაიმერ / მთვლელთან რეზისტორის მიერთებისათვის)
$\overline{\text{PEN}}$	1	I	დაპროგრამების ნებადართვა
AREF	62	P	აცგ საყრდენი ძაბვის შესასვლელი
AGND	63	P	საერთო ანალოგური გამომყვანი

ცხრილი A-2. მიმდევრობითი არხით დაპროგრამების რეჟიმის ბრძანებები

ბრძანების დასახელება		ბრძანების ფორმატი				ბრძანების აღწერა
		1-ი ბაიტი	2-ე ბაიტი	3-ე ბაიტი	4-ე ბაიტი	
პროგრამირების ნებადართვა		1010 1100	0101 0011	XXXX XXXX	XXXX XXXX	იძლევა მიკროკონტროლერის პროგრამირების ნებადართვას, მანამ სანამ RESET გამომყვანზე არის დაბალი დონის ძაბვა.
კრისტალის გასუფთავება		1010 1100	100X XXXX	XXXX XXXX	XXXX XXXX	FLAH და EEPROM მეხსიერების შემცველობის გასუფთავება. აუცილებელია ბრძანების გაგზავნის შემდეგ შესრულდეს: 1. პაუზა დაყობნებით tw_{E_ERASE} 2. $\overline{RESET}$ გამომყვანს უნდა მიეწოდოს დადებითი იმპულსი. 3. განხორციელდეს მოლოდინი არა უმეტეს 20 მწმ. 4. კვლავ უნდა გაიგზავნოს "პროგრამირების ნებადართვის" ბრძანება.
FLASH მეხსიერებიდან წაკითხვა		0010 H000	aaaa aaaa	bbbb bbbb	0000 0000	პროგრამული მეხსიერების უმცირესი (H =0) ან უფროსი (H =1) ბაიტის წაკითხვა, განთავსებული a:b მისამართზე.
FLASH მეხსიერებაში ჩაწერა (Ttiny ოჯახისთვის)		0100 H000	xxxx aaaa	bbbb bbbb	iiii iiii	უმცროსი (H =0) ან უფროსი (H =1) (i) ბაიტის ჩაწერა პროგრამების მეხსიერების a:b მისამართზე.
FLASH მეხსიერების გვერდების ჩატვირთვა (Mega ოჯახისთვის)	8k	0100 H000	xxxx xxxx	xxxb bbbb	iiii iiii	უმცროსი (H =0) ან უფროსი (H =1) (i) ბაიტის ჩაწერა პროგრამების მეხსიერების გვერდების ბუფერში b მისამართზე (სიტყვისა და გვერდის მისამართი); პირველად უნდა ჩაიტვირთოს უმცროსი ბაიტი.
	16k			xxbb bbbb		
	32k			xxbb bbbb		
	64k			xxbb bbbb		
	128k			xxbb bbbb		

დანართი 2

გაგრძელება

ბრძანების დასახელება		ბრძანების ფორმატი				ბრძანების აღწერა
		1-ი ბაიტი	2-ე ბაიტი	3-ე ბაიტი	4-ე ბაიტი	
FLASH მეხსიერებაში გვერდის ჩაწერა (Mega ოჯახისთვის)	8k	0100 1100	xxxx xxxx	bbbx xxxx	xxxx xxxx	გვერდის ჩაწერა პროგრამის მეხსიერების a:b მისამართზე.
	16k		xxxa aaaa	bbxx xxxx		
	32k		xxaa aaaa	bbxx xxxx		
	64k		xaaa aaaa	bxxx xxxx		
	128k		aaaa aaaa	bxxx xxxx		
წაკითხვა EEPROM მეხსიერებიდან		1010 0000	xxxx aaaa	bbbb bbbb	0000 0000	EEPROM მეხსიერების a:b მისამართიდან უჯრედის შემცველობის წაკითვა.
ჩაწერა EEPROM მეხსიერებაში		1100 0000	xxxx aaaa	bbbb bbbb	iiii iiii	EEPROM მეხსიერების a:b მისამართზე (i) მნიშვნელობის ჩაწერა.
EEPROM მეხსიერებაში გვერდის ჩაწერა		1100 0001	0000 0000	0000 00bb	iiii iiii	EEPROM მეხსიერების გვერდის ბუფერის b მისამართზე (ბაიტის მისამართი გვერდზე) (i) ბაიტის ჩაწერა .
EEPROM მეხსიერებაში გვერდის ჩაწერა		1100 0010	00xx xxxa	bbbb bb00	xxxx xxxx	EEPROM მეხსიერების a:b მისამართზე გვერდის ჩაწერა.
დაცვის უჯრედის წაკითხვა		0101 1000	xxxx xxxx	xxxx xxxx	xx00 0000	--
დაცვის უჯრედებში ჩაწერა		1010 1100	111x xxxx	xxxx xxxx	-	დაცვის უჯრედების ჩაწერა; უჯრედის პროგრამირებისთვის შესაბამისი (i) უჯრედი უნდა განულდეს.
მაკონფიგურირებელი უჯრედების წაკითხვა (Mega ოჯახის უმცროსი ბაიტის)		0101 0000	0000 0000	xxxx xxxx	0000 0000	მაკონფიგურირებელი უჯრედების წაკითხვა; განულებული თანრიგი ნიშნავს, რომ შესაბამისი მაკონფიგურებელი უჯრედი დაპროგრამირებულია.
უფროსი კონფიგურირებული ბაიტის წაკითხვა (Mega ოჯახის)		0101 0000	0000 1000	xxxx xxxx	0000 0000	იგივე
დამატებითი მაკონფიგურირებელი ბაი- ტის წაკითხვა (Mega ოჯახის)		0101 1000	0000 1000	xxxx xxxx	0000 0000	იგივე

დანართი 2

გაგრძელება

ბრძანების დასახელება	ბრძანების ფორმატი				ბრძანების აღწერა
	1-ი ბაიტი	2-ე ბაიტი	3-ე ბაიტი	4-ე ბაიტი	
მაკონფიგურირებელი უჯრედების ჩაწერა(Mega ოჯახის -უმცროსი ბაიტის)	1010 1100	1010 0000	xxxx xxxx	iiii iiii	მაკონფიგურირებელი უჯრედების ჩაწერა. უჯრედის თანრიგის პროგრამირებისთვის აუცილებელია შესაბამისი (i) თანრიგის განულება.
უფროსი კონფიგურაციული ბაიტის ჩაწერა (Mega ოჯახის)	1010 1100	1010 1000	xxxx xxxx	iiii iiii	იგივე
დამატებითი მაკონფიგურებელი ბაიტის ჩაწერა (Mega ოჯახის)	1010 1100	1010 0100	xxxx xxxx	iiii iiii	იგივე
იდენტიფიკატორის წაკითხვა	0011 0000	xxxx xxxx	0000 00bb	0000 0000	უჯრედის (0) იდენტიფიკატორის წაკითხვა b მისამართიდან (მხოლოდ დაცულ 1 და 2 რეჟიმში ნახეთ ცხრილ 4.2.)
დაკალიბრებული ბაიტის წაკითხვა (ATmega)	0011 1000	xxxx xxxx	0000 00bb	0000 0000	მაკალიბრებელი (0) კონსტანტის წაკითხვა b მისამართიდან

შენიშვნა: ცხრილში გამოყენებული პირობითი აღნიშვნების განმარტება:

a – მისამართის ბაიტის უფროსი თანრიგები;

b – მისამართის ბაიტის უმცროსი თანრიგები;

H – "0" - უმცროსი ბაიტი, "I"-უფროსი ბაიტი;

i – მიკროკონტროლერისთვის გაგზავნილი მონაცემები;

0 - მიკროკონტროლერიდან წაკითხული მონაცემები;

x - თანრიგის მდგომარეობა სულერთია.

ლიტერატურა

1. Фрунзе А.В. Микроконтроллеры? Это же просто! 2004.
2. Евстифеев А.В. Микроконтроллеры AVR семейства Tiny и Mega фирмы ATMEL,2008.
3. Atmega 128. Datasheet.
4. Белов А.В. Создаем устройства на микроконтроллерах. СПБ: Наука и техника,2007.
5. Масла Ю.С. Микроконтроллеры PIC. Архитектура и программирование. Из-во “ДМК пресс”,2009.
6. Редькин П.П. 32/16 - битные микроконтроллеры ARM7 семейства AT 91SAM7 фирмы ATMEL. Руководство пользователя, 2008.

სარჩევი